Integración de grafos de conocimiento educativos con modelos de lenguaje masivos mediante arquitecturas RAG para evaluación personalizada en la enseñanza de programación

Trabajo de Fin de Máster
Máster en Investigación en IA — UNED
Adrián Bueno Junquero — Dir.: José Luis Fernández Vindel
Curso 2025/2026

Índice

1 Declaración de autoría y originalidad

Declaro que este Trabajo de Fin de Máster es obra propia y original, fruto de mi trabajo personal, y que no ha sido presentado con anterioridad, ni en su totalidad ni en parte, para la obtención de ningún otro título. Todas las fuentes consultadas, las ideas ajenas y los materiales de terceros empleados se han citado y referenciado de manera explícita conforme a las normas de integridad académica, sin incurrir en plagio en ninguna de sus formas.

Afirmo asimismo que los resultados, las cifras, las tablas y las conclusiones que se presentan en esta memoria son reales y reproducibles a partir del código, los datos y los artefactos del proyecto, y que no han sido inventados, alterados ni inflados. Los objetivos cuantitativos del anteproyecto se mantienen, allí donde aparecen, como metas e hipótesis a contrastar y nunca como logros consumados, y los hallazgos negativos se reportan con la misma franqueza que los positivos.

Asumo la plena responsabilidad sobre el contenido de este documento y declaro su conformidad con la normativa de la Universidad Nacional de Educación a Distancia (UNED) sobre integridad académica.

Adrián Bueno Junquero · Curso 2025/2026

2 Declaración sobre el uso de herramientas de inteligencia artificial

En la elaboración de este Trabajo de Fin de Máster he utilizado herramientas de inteligencia artificial generativa como apoyo a la redacción y a la organización del texto, partiendo de borradores que he revisado, reescrito con mi propia voz y verificado en su totalidad.

El diseño de la investigación, la construcción y la validación del grafo de conocimiento, el desarrollo de la arquitectura RAG y del *fine tuning*, la ejecución de los experimentos y el análisis estadístico, y todas las cifras, tablas y conclusiones que aquí se presentan son obra propia, reales y reproducibles a partir del código y de los artefactos del proyecto.

Asumo la plena responsabilidad sobre el contenido de este documento y declaro este uso de herramientas de inteligencia artificial conforme a la normativa de la Universidad Nacional de Educación a Distancia (UNED) sobre integridad académica.

Adrián Bueno Junquero · Curso 2025/2026

3 Resumen

La retroalimentación formativa es uno de los factores con mayor incidencia en el aprendizaje (Hattie & Timperley, 2007), pero su provisión personalizada y a escala sigue siendo un reto en la enseñanza de la programación. Los modelos de lenguaje masivos (LLM) ofrecen una vía prometedora para generar feedback automático, aunque arrastran limitaciones bien documentadas, entre ellas las alucinaciones, la falta de fundamentación en un modelo pedagógico explícito y la escasa trazabilidad de sus afirmaciones (Bender et al., 2021). Este Trabajo de Fin de Máster aborda dicho problema mediante el diseño, la implementación y la validación de una arquitectura que integra un Grafo de Conocimiento Educativo (EKG) con un LLM a través del paradigma de Generación Aumentada por Recuperación (RAG), con el objetivo de producir retroalimentación fundamentada, explicable y adaptada al nivel del estudiante.

Mi contribución se articula en torno a tres ejes. En primer lugar, he construido un EKG canónico del dominio de la programación en Python, modelado en OWL 2 RL y serializado en Turtle. Este grafo reúne 157 conceptos propios, 1772 enunciados afirmados que ascienden a 4786 tras el razonamiento OWL-RL, 20 clases y 28 propiedades. El grafo incorpora alineamiento con Wikidata mediante 30 enlaces `skos:exactMatch`, jerarquías `skos:broader`, relaciones N-arias para representar evaluaciones de actividad y reificación con procedencia. Opté por `skos:exactMatch` en lugar de `owl:sameAs` para enlazar con Wikidata sin imponer la fusión lógica de individuos propia del perfil OWL-RL.[1] Su consistencia se verifica con SHACL (cero violaciones sobre el grafo canónico). La inferencia demuestra su valor, pues una consulta de conceptos que devuelve cero resultados sin razonamiento alcanza 157 al activarlo. Esos 157 son los conceptos propios inferidos como `pyedu:Concepto`, donde las 30 entidades de Wikidata, enlazadas con `skos:exactMatch`, no se incorporan a ese tipo porque `skos:exactMatch` no propaga la clase, a diferencia de lo que haría `owl:sameAs`. En segundo lugar, he desarrollado una arquitectura RAG completa que analiza el código del estudiante mediante árboles de sintaxis abstracta (AST) y métricas de complejidad, lo indexa con embeddings (`nomic-embed-text`), recupera subgrafos relevantes combinando similitud semántica y expansión SPARQL, y genera prompts que alimentan un LLM local desplegado con Ollama, lo que garantiza la soberanía de los datos. En tercer lugar, he implementado un módulo de explicabilidad con visualización interactiva del subgrafo recuperado (FastAPI y Cytoscape.js) y marcadores de procedencia.

La validación contrasta cuatro configuraciones: un LLM base (Sistema A), un LLM afinado con QLoRA (Sistema B), un LLM base aumentado con GraphRAG (Sistema C) y un híbrido que combina *fine tuning* y GraphRAG (Sistema D). Los objetivos cuantitativos del anteproyecto (precisión diagnóstica >=85%, alucinaciones <=5%) eran hipótesis que debía contrastar, no resultados garantizados. La evaluación realizada es automática (métricas objetivas y juez LLM) sobre datos sintéticos y una muestra de 50 casos held-out; el panel de docentes humanos previsto queda como trabajo futuro. Los resultados muestran que el fine-tuning mejora la clasificación del error (acierto de categoría de 0,26 en A frente a 0,70 en B) mientras que GraphRAG mejora la identificación del concepto (0,18 en A frente a 0,48 en C) y la trazabilidad (1,72 frente a 2,92 sobre 5). Completada ya la evaluación de las cuatro configuraciones con n=50, el Sistema D híbrido sintetiza ambas vías y confirma su complementariedad, ya que gana o empata en las siete dimensiones medidas (mejor en seis) y se sitúa por delante de B y C, que a su vez superan al modelo base A. En particular, D alcanza 0,76 en acierto de categoría y 0,54 en acierto de concepto, con 4,04 en identificación y 3,16 en trazabilidad sobre 5. Aun así, no doy por cumplido ningún target del anteproyecto, ya que el mejor sistema (D) llega a 0,76 en categoría, lejos del 85% fijado, y la validación carece todavía de panel humano; reporto solo lo que he medido de forma efectiva. Una línea de reconstrucción independiente con alineación por preferencias ORPO,[2] que no debe confundirse con el experimento canónico anterior, matiza además la tesis. La augmentación por recuperación no mejora en abstracto, dado que el RAG por pasajes no supera a la base (3,467 y 3,567 frente a 3,733) y solo la

[1] lo que refleja un uso maduro de los datos enlazados

[2] cuatro variantes Base / v2 / v3 / v4, juez `qwen2.5:32b`, n=10

recuperación semántica y quirúrgica del grafo ontológico (ORPO-v4) la supera (4,233), con dos derrotas y una mejora de recuperación, no de fine-tuning.

3.1 Abstract

Formative feedback is among the strongest levers for learning (Hattie & Timperley, 2007), yet delivering it in a personalised, scalable way remains a challenge in programming education. Large language models (LLMs) offer a promising avenue for automatic feedback generation, but suffer from well-documented limitations: hallucinations, lack of grounding in an explicit pedagogical model, and poor traceability of their claims (Bender et al., 2021). This Master's Thesis tackles this problem through the design, implementation and validation of an architecture that integrates an Educational Knowledge Graph (EKG) with an LLM via Retrieval-Augmented Generation (RAG), aiming to produce grounded, explainable feedback adapted to the learner's level.

The work delivers a canonical EKG for the Python programming domain, modelled in OWL 2 RL, comprising 157 native concepts, 1772 asserted statements rising to 4786 after OWL-RL reasoning, alignment with Wikidata via `skos:exactMatch`, N-ary relations and provenance-aware reification, validated with SHACL (zero violations). A complete RAG pipeline analyses student code through abstract syntax trees, indexes it with embeddings, retrieves relevant subgraphs via semantic similarity and SPARQL expansion, and feeds a locally deployed LLM, ensuring data sovereignty. An explainability module visualises the retrieved subgraph and exposes provenance markers.

The evaluation contrasts four configurations: a base LLM (A), a QLoRA fine-tuned model (B), a Graph-RAG-augmented model (C) and a hybrid combining fine-tuning and GraphRAG (D). The ambitious quantitative targets stated in the project proposal were hypotheses to be tested, not achieved results. The evaluation conducted is automatic, on synthetic data and a held-out sample of 50 cases; a human teacher panel remains future work. The results show that fine-tuning improves error classification (category accuracy 0.26 in A versus 0.70 in B), whereas GraphRAG improves concept identification (0.18 versus 0.48) and traceability (1.72 versus 2.92 out of 5). With the four-system evaluation now completed on n=50, the hybrid System D synthesises both routes and confirms their complementarity: it wins or ties on all seven measured dimensions (best on six), ahead of B and C, which in turn surpass the base model A. System D reaches 0.76 category accuracy and 0.54 concept accuracy, with 4.04 identification and 3.16 traceability out of 5. Even so, no proposal target is claimed as met: the best system (D) attains 0.76 in category, well below the 85% target, and the validation still lacks a human panel; only what was actually measured is reported. A separate reconstruction line using ORPO preference alignment (four variants Base / v2 / v3 / v4, judge `qwen2.5:32b`, n=10), which must not be conflated with the canonical experiment above, further qualifies the thesis: retrieval augmentation does not help in the abstract, since passage-based RAG fails to beat the base (3.467 and 3.567 versus 3.733) and only the surgical semantic retrieval over the ontological graph (ORPO-v4) surpasses it (4.233), with two regressions and an improvement attributable to retrieval, not fine-tuning.

Palabras clave: grafo de conocimiento educativo, generación aumentada por recuperación, modelos de lenguaje masivos, retroalimentación formativa, enseñanza de la programación, OWL 2 RL, explicabilidad, fine-tuning, QLoRA, evaluación personalizada.

Keywords: educational knowledge graph, retrieval-augmented generation, large language models, formative feedback, programming education, OWL 2 RL, explainability, fine-tuning, QLoRA, personalised assessment.

4 Introducción y planteamiento del problema

4.1 La tensión entre escalabilidad e individualización en la enseñanza de la programación

La enseñanza introductoria de la programación arrastra desde hace décadas una tensión estructural que ningún avance metodológico ha conseguido resolver del todo. Esa tensión enfrenta la necesidad de atender a un número creciente de estudiantes con la imposibilidad práctica de ofrecer a cada uno la atención individualizada que el aprendizaje de una competencia tan compleja exige. Aprender a programar no consiste únicamente en memorizar una sintaxis ni en reproducir patrones de código; implica construir modelos mentales correctos sobre la ejecución de los programas, depurar concepciones erróneas profundamente arraigadas y desarrollar una capacidad de razonamiento abstracto que se adquiere mediante la práctica deliberada acompañada de una retroalimentación de calidad. Es esa retroalimentación, ese diálogo formativo entre el aprendiz y un experto capaz de comprender por qué se ha equivocado y de orientar su siguiente paso, lo que resulta más difícil de escalar.

El problema no es nuevo ni anecdótico. Los estudios clásicos sobre las dificultades del alumnado novel ya documentaron hace más de dos décadas la magnitud del reto. El célebre estudio multiinstitucional de McCracken et al. (2001), realizado sobre estudiantes de primer curso de varias universidades, reveló que una proporción considerable del alumnado era incapaz de programar soluciones a problemas relativamente sencillos al finalizar un curso introductorio, con puntuaciones medias muy inferiores a las esperadas por el profesorado. En la misma línea, Watson y Li (2014), tras una revisión sistemática de las tasas de aprobación en asignaturas introductorias de programación a escala internacional, situaron la tasa media de fracaso en torno a un tercio del alumnado, una cifra estable a lo largo del tiempo y de las geografías. Estos datos dibujan un panorama en el que la dificultad intrínseca de la materia se combina con una carencia crónica de acompañamiento individualizado para producir resultados de aprendizaje insatisfactorios.

La raíz operativa de esta carencia es, en buena medida, una cuestión de proporciones. En contextos universitarios masificados, y muy especialmente en modalidades de educación a distancia o en línea como la que caracteriza a la propia Universidad Nacional de Educación a Distancia, no es infrecuente encontrar ratios de en torno a cincuenta estudiantes por cada docente, e incluso superiores en las asignaturas de mayor matrícula. Esta proporción de aproximadamente 50:1 convierte en materialmente inviable cualquier aspiración de revisar artesanalmente, una por una, las soluciones que cada estudiante entrega a lo largo del curso. Puedo poner cifras concretas a esta inviabilidad. Una corrección formativa de calidad de una única actividad de programación —entendiendo por tal no la mera comprobación de si el código produce la salida esperada, sino la lectura comprensiva del código, la identificación del error conceptual que subyace al fallo, la valoración de la calidad de la solución y la redacción de una orientación pedagógica útil— requiere del orden de quince a veinte minutos de trabajo experto. Si se multiplica esa horquilla por las decenas de estudiantes a cargo de un docente y por la docena o más de actividades que componen un curso, el resultado es una carga de trabajo que excede con holgura el tiempo razonablemente disponible. La aritmética es implacable, pues solo caben tres salidas. O se reduce el número de actividades, o se reduce la profundidad de la corrección, o se renuncia a la individualización. En la práctica, las tres renuncias coexisten en grados diversos, y todas ellas degradan la experiencia formativa.

Esta es la paradoja que vertebra el presente trabajo y que llamaré, a lo largo de la memoria, la paradoja entre escalabilidad e individualización, y que la Figura 1 sintetiza de un vistazo. Las dos propiedades que más valoramos en un sistema educativo (que llegue a muchos y que atienda bien a cada uno) se encuentran en oposición directa cuando el recurso escaso es el tiempo de un experto humano. Aumentar la escala suele implicar diluir la individualización; preservar la individualización suele implicar limitar la escala. La aspiración legítima de cualquier intervención tecnológica en este ámbito es intentar relajar esa oposición y ofrecer una retroalimentación que se aproxime a la calidad de la individualizada sin

incurrir en su coste, para que la escala deje de pagarse con la renuncia a la atención. No se trata de sustituir al docente, una pretensión que ni es deseable ni, como argumentaré, técnicamente sensata, sino de amplificar su alcance descargándolo de la parte más mecánica del diagnóstico para que pueda concentrar su juicio donde resulta insustituible.

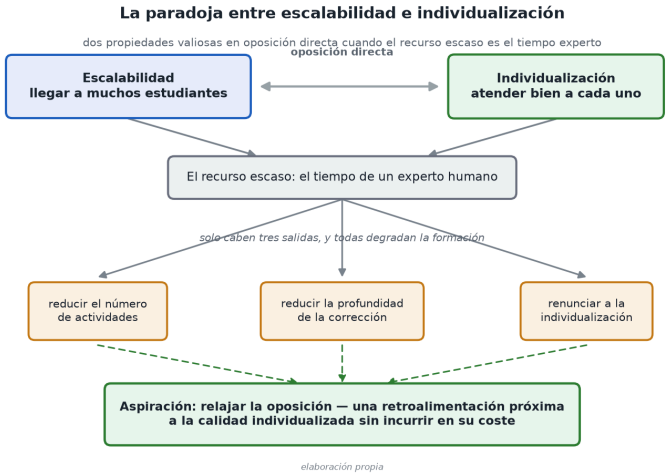

Figura 1. La paradoja entre escalabilidad e individualización: dos propiedades en oposición directa bajo el recurso escaso del tiempo experto, las tres salidas forzadas que degradan la formación y la aspiración de relajar esa oposición. *Fuente: elaboración propia.*

4.2 Los límites del feedback automático tradicional

La respuesta históricamente dominante a esta paradoja ha sido la automatización de la corrección. Desde los primeros sistemas de evaluación automática hasta las plataformas contemporáneas de jueces en línea, la comunidad ha invertido un esfuerzo considerable en delegar en la máquina la verificación de las soluciones del alumnado. Estos sistemas resuelven con eficacia el problema de la escala, porque pueden ejecutar miles de entregas en segundos y devolver un veredicto inmediato. El problema reside en la naturaleza de ese veredicto. La revisión sistemática de Keuning, Jeuring y Heeren (2019) sobre el estado del arte de la retroalimentación automática en la enseñanza de la programación ofrece una radiografía especialmente reveladora de esta limitación. Tras analizar un corpus amplio de herramientas y sistemas descritos en la literatura, los autores constataron que la inmensa mayoría (cerca de un ochenta y nueve por ciento de los sistemas estudiados) se limitaba a ofrecer feedback de carácter diagnóstico —información sobre qué está mal o si la solución pasa o no las pruebas—, pero sin avanzar hacia formas de retroalimentación más ricas que orienten al estudiante sobre cómo corregir el problema o, más ambiciosamente, sobre los principios conceptuales que debería revisar para no volver a cometerlo.

Esta constatación conecta con un marco teórico bien establecido sobre qué hace que la retroalimentación sea formativamente eficaz. Hattie y Timperley (2007), en su influyente modelo sobre el poder del feedback, distinguieron varios niveles según las preguntas a las que responde, que son dónde voy (cuáles son las metas), cómo voy (el diagnóstico de la situación actual) y hacia dónde seguir (los pasos que permiten progresar). Su síntesis de la evidencia empírica mostró que la retroalimentación que se queda en el nivel puramente diagnóstico, e incluso la que se limita a corregir la tarea concreta sin abordar el proceso ni la autorregulación del aprendizaje, tiene un efecto mucho más débil sobre el aprendizaje que aquella que ayuda al estudiante a entender el porqué de su error y a transferir esa comprensión a situaciones nuevas. A la luz de este marco, el dato de Keuning et al. (2019) adquiere todo su peso. Si casi nueve de cada diez sistemas automáticos se detienen en el diagnóstico, entonces la automatización tradicional ha resuelto la escala justamente en la dimensión del feedback que menos contribuye al aprendizaje, dejando sin atender las dimensiones que más lo favorecen.

A esta limitación de fondo se suman carencias técnicas bien conocidas de los enfoques basados en pruebas y en análisis estático predefinido. Los sistemas de pruebas unitarias informan de que una solución falla, pero rara vez de por qué falla en términos comprensibles para un aprendiz; un caso de prueba que no pasa puede deberse a un error conceptual profundo o a un descuido trivial, y el sistema trata ambos por igual. Los analizadores estáticos basados en reglas detectan patrones predefinidos, pero son frágiles ante la enorme variabilidad de las soluciones incorrectas que produce el alumnado novel, cuyos errores no siempre encajan en las categorías anticipadas por el diseñador de las reglas. Trabajos como el de Marwan, Gao y otros (2020) sobre el efecto de las explicaciones en la retroalimentación de programación han mostrado que añadir explicaciones a las pistas mejora la percepción de utilidad y el aprendizaje del estudiante, lo que refuerza la idea de que el déficit explicativo de los sistemas tradicionales no es un detalle menor, sino el núcleo del problema. La automatización clásica nos ha dado escala sin calidad pedagógica suficiente, y deja intacta (y en cierto modo agravada) la cara más importante de la paradoja.

A modo de síntesis, las magnitudes que se han venido invocando en este planteamiento del problema pueden recogerse de forma compacta para fijar el orden de los números en juego.

Magnitudes que cuantifican el problema (cifras del propio capítulo).

Magnitud	Valor referido en el texto
Ratio estudiantes por docente (educación a distancia)	≈ 50:1
Tiempo de corrección formativa por actividad	15–20 min
Tasa media de fracaso en programación introductoria	≈ un tercio del alumnado
Sistemas automáticos limitados a feedback diagnóstico	≈ 89 %

Fuente: elaboración propia a partir de las cifras citadas en este capítulo.

4.3 La irrupción de los modelos de lenguaje masivos y sus dos obstáculos

La aparición y rápida maduración de los modelos de lenguaje masivos ha cambiado de manera sustancial el horizonte de lo posible. A diferencia de los sistemas de reglas o de pruebas, estos modelos exhiben una notable capacidad para leer código en lenguaje natural y formal, comprender su intención, identificar fallos y redactar explicaciones fluidas y aparentemente adaptadas al nivel del interlocutor. Los modelos especializados en código, en particular, han demostrado competencias considerables en tareas de generación y comprensión de programas; el trabajo de Chen et al. (2021) sobre modelos entrenados en código, que dio lugar a herramientas hoy ampliamente difundidas, evidenció hasta qué punto estos sistemas pueden razonar sobre fragmentos de programa. Esta capacidad sugiere, a primera vista, una solución elegante a la paradoja. Un modelo de lenguaje podría generar, a coste marginal casi nulo y de forma instantánea, esa retroalimentación rica, explicativa y orientada al proceso que los sistemas tradicionales no alcanzan y que el docente humano no puede producir a escala. La promesa es tentadora y, en parte, real. Pero apoyarse ingenuamente en ella sería imprudente, porque los modelos de lenguaje arrastran consigo dos obstáculos serios que, en un contexto educativo, resultan especialmente problemáticos.

El primer obstáculo son las alucinaciones. Con este término se designa la propensión de los modelos a generar afirmaciones fluidas, gramaticalmente impecables y persuasivas que, sin embargo, son factualmente incorrectas. El modelo no distingue, en su funcionamiento interno, entre lo que sabe y lo que inventa, porque produce la continuación más plausible dada su distribución de probabilidad, y esa plausibilidad estadística no garantiza veracidad. La revisión de Zhang et al. (2023) sobre el fenómeno de la alucinación en los modelos de lenguaje sistematiza sus tipos y causas, y la literatura empírica que recoge sitúa las tasas de alucinación en rangos que, según la tarea y el modelo, se mueven con frecuencia en el entorno del ocho al quince por ciento de las respuestas. Trasladado al aula, este margen es difícilmente aceptable. Una retroalimentación que diagnostica un error inexistente, que atribuye al estudiante una concepción equivocada que no tiene, o que afirma con aplomo que una función se

comporta de un modo que no se corresponde con la semántica real del lenguaje, no solo es inútil, sino activamente dañina, porque introduce ruido en el modelo mental del aprendiz y erosiona su confianza en el sistema y, peor aún, en sus propias soluciones correctas. En un dominio en construcción como el del estudiante novel, que carece todavía del criterio necesario para detectar que el sistema se equivoca, una alucinación segura de sí misma puede consolidar un error en lugar de corregirlo. Las advertencias de Bender et al. (2021) sobre los riesgos de tratar a estos modelos como si comprendieran lo que dicen, cuando en realidad operan sobre regularidades estadísticas de forma, son aquí plenamente pertinentes.

El segundo obstáculo, más sutil pero igualmente decisivo, es la falta de trazabilidad. Aun cuando un modelo de lenguaje acierte en su diagnóstico, lo hace de un modo opaco, pues produce una afirmación sin poder justificar de dónde procede, en qué conocimiento se apoya ni por qué cadena de razonamiento ha llegado a ella. No hay, en su salida, marcadores de procedencia que permitan al docente verificar la validez de la retroalimentación, ni al estudiante remontarse a los conceptos y principios que fundamentan la corrección. Esta opacidad es un problema grave en un contexto educativo por al menos dos razones. La primera es de responsabilidad y confianza, porque un docente no puede avalar profesionalmente, ni delegar la evaluación de su alumnado en, un sistema cuyas conclusiones no puede auditar; la transparencia no es un lujo, sino una condición de aceptabilidad. La segunda es propiamente pedagógica, pues buena parte del valor formativo de la retroalimentación reside en su capacidad de conectar el error concreto con el cuerpo de conocimiento que el estudiante debe dominar, de modo que pueda comprender el principio general que ha vulnerado. Una respuesta sin trazabilidad rompe ese vínculo; ofrece una corrección sin un mapa del territorio conceptual al que pertenece. Como han señalado Gašević y colaboradores (2022) al reflexionar sobre la incorporación responsable de la inteligencia artificial a la analítica del aprendizaje, la explicabilidad y la fundamentación de las decisiones algorítmicas son requisitos ineludibles cuando esas decisiones afectan a la trayectoria educativa de las personas.

Estos dos obstáculos comparten una raíz común, como recoge la Figura 2, pues el modelo de lenguaje, por su propia naturaleza, no tiene un anclaje en una representación explícita y verificable del conocimiento del dominio. Su saber está difuso en sus parámetros, mezclado e inseparable, sin posibilidad de ser inspeccionado ni de ser invocado como fundamento. De ahí que, si se quiere aprovechar la indudable capacidad expresiva y comprensiva de estos modelos sin heredar su fragilidad factual y su opacidad, sea necesario dotarlos de ese anclaje desde el exterior. Y es aquí donde el presente trabajo sitúa su propuesta.

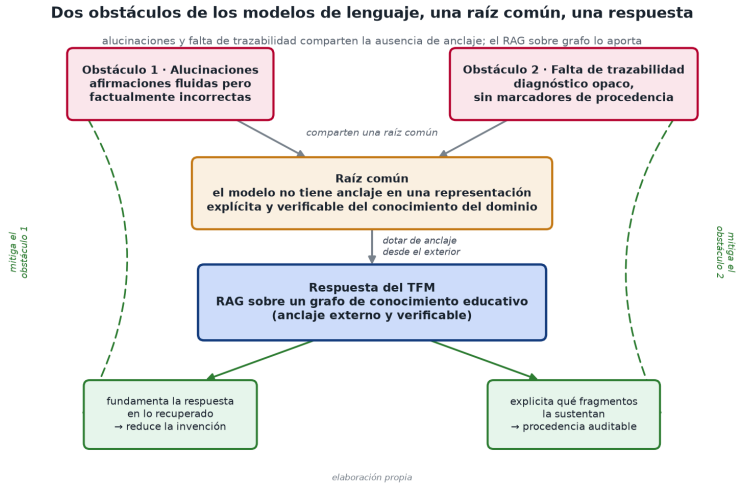

Figura 2. Los dos obstáculos de los modelos de lenguaje (alucinaciones y falta de trazabilidad) comparten la ausencia de anclaje en una representación explícita y verificable; el RAG sobre un grafo de conocimiento aporta ese anclaje externo y mitiga ambos con un efecto doble. *Fuente: elaboración propia.*

4.4 La propuesta de grafos de conocimiento educativos integrados mediante RAG

La hipótesis de trabajo que articula este Trabajo de Fin de Máster es que los dos obstáculos descritos pueden mitigarse simultáneamente integrando el modelo de lenguaje con un grafo de conocimiento educativo del dominio mediante una arquitectura de generación aumentada por recuperación. La generación aumentada por recuperación, formulada por Lewis et al. (2020), es un paradigma en el que el modelo de lenguaje, en lugar de generar su respuesta apoyándose exclusivamente en su conocimiento paramétrico interno, recupera primero información pertinente de una fuente externa de conocimiento y condiciona su generación a esa información recuperada. El efecto buscado es doble. Por un lado, al fundamentar la respuesta en contenido recuperado de una fuente fiable, se reduce el espacio para la invención y, con ello, la incidencia de alucinaciones. Por otro, al hacer explícito qué fragmentos de conocimiento han sustentado la respuesta, se abre la puerta a la trazabilidad, ya que la corrección deja de ser un dictamen opaco para convertirse en una afirmación anclada en fuentes identificables.

La originalidad de la propuesta no reside en aplicar el paradigma genérico de la recuperación aumentada, hoy ya extendido, sino en la naturaleza de la fuente de conocimiento que se recupera. En lugar de un corpus de documentos no estructurados sobre los que se calculan similitudes léxicas o semánticas, se propone emplear un grafo de conocimiento educativo —una representación formal, explícita y estructurada del dominio de la programación—, en la que los conceptos, sus propiedades y, sobre todo, las relaciones que los vinculan (de prerrequisito, de generalización, de composición, de error típico asociado) quedan modelados de manera que una máquina pueda razonar sobre ellos. Los grafos de conocimiento, cuya fundamentación y panorama sistematizan Hogan et al. (2021), ofrecen ventajas decisivas frente a los corpus textuales planos. Permiten recuperar no fragmentos aislados, sino subgrafos coherentes que capturan el entramado conceptual relevante para un error concreto; permiten realizar inferencias lógicas que enriquecen el conocimiento explícitamente afirmado; y, de manera crucial para nuestro propósito, proporcionan una procedencia nítida, donde cada afirmación recuperada es una arista identificable del grafo, con su semántica formal asociada, lo que dota a la trazabilidad de una base sólida. La línea de trabajo sobre la combinación de grafos y recuperación aumentada, que han explorado recientemente autores como Edge et al. (2024) en su propuesta de recuperación aumentada basada en grafos, o las revisiones sobre grafos de conocimiento aplicados a la educación de Abu-Salih y Alotaibi (2024) y de Qu et al. (2024), confirma que se trata de una dirección fértil y todavía en plena exploración.

En el plano concreto, la arquitectura que se diseña, implementa y valida en este trabajo combina varios componentes, como resume la Figura 3. Un analizador de código basado en el árbol de sintaxis abstracta, complementado con métricas de complejidad, extrae una representación estructural de la solución del estudiante. Un sistema de recuperación localiza, por similitud semántica sobre representaciones vectoriales y por expansión mediante consultas formales al grafo, el subgrafo de conceptos y relaciones pertinente al caso. Ese subgrafo se inyecta como contexto fundamentado en el modelo de lenguaje, que genera la retroalimentación condicionada a él. Y un módulo de explicabilidad visualiza el subgrafo recuperado y marca la procedencia de las afirmaciones, cerrando el círculo de la trazabilidad. Un rasgo deliberado y, entiendo, relevante del diseño es su despliegue local, ya que tanto el grafo como el modelo de lenguaje operan sobre infraestructura propia, sin enviar el código del alumnado a servicios de terceros, lo que responde a exigencias de privacidad y soberanía de los datos educativos que abordaré con detalle en el capítulo dedicado a las consideraciones éticas. Anticipo también una salvedad metodológica que se desarrollará en su momento. Algunas de las tecnologías concretas finalmente empleadas difieren de las anticipadas en el anteproyecto, y esas decisiones de implementación se justifican más adelante en lugar de silenciarse.

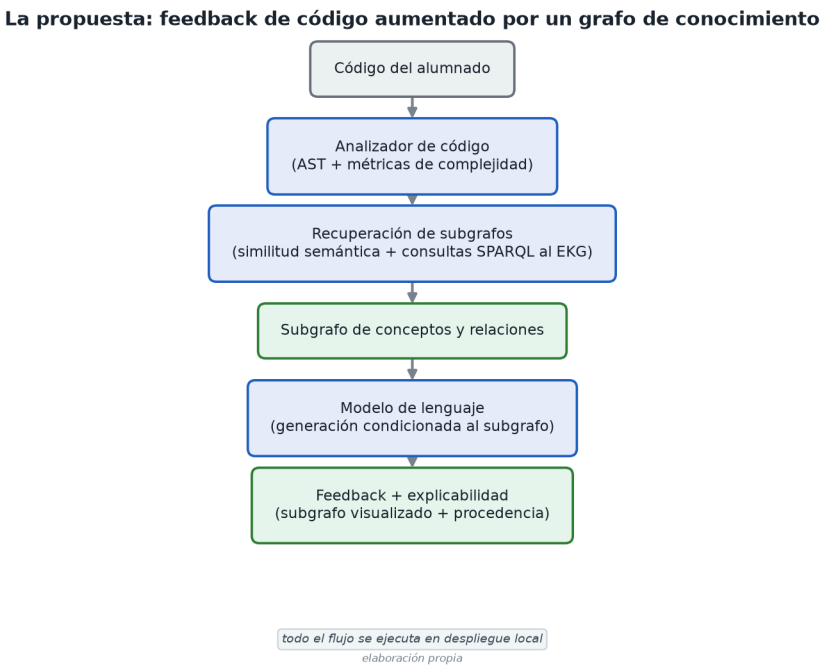

Figura 3. Pipeline propuesto: feedback de código aumentado por el grafo (despliegue local). *Fuente: elaboración propia.*

4.5 Relevancia, aportación y carácter de los resultados

La relevancia de este trabajo se sitúa en una intersección que, pese al dinamismo de las tres áreas que la componen, permanece inexplorada. Por un lado está la línea consolidada de la retroalimentación automática en la enseñanza de la programación, que ha resuelto la escala pero no la calidad pedagógica. Por otro, la pujante línea de los modelos de lenguaje aplicados a la educación, que aporta una calidad expresiva inédita lastrada por las alucinaciones y la opacidad. Y, en tercer lugar, la madura tradición de la web semántica y los grafos de conocimiento, con sus estándares de representación y razonamiento formal. Cada una de estas áreas ha avanzado en buena medida por separado. La integración específica de un grafo de conocimiento educativo, formalmente modelado y semánticamente razonado, con un modelo de lenguaje masivo mediante recuperación aumentada, y orientada de manera deliberada a la evaluación formativa y trazable de la programación, constituye un territorio que la literatura ha rozado pero apenas transitado. Es en esa intersección donde el presente trabajo aspira a aportar.

La aportación es a la vez metodológica y artefactual. En el plano del artefacto, se construye un grafo de conocimiento educativo del dominio de la programación en Python, formalizado con los estándares de la web semántica, dotado de inferencia lógica, validado en cuanto a su integridad estructural y enriquecido con vínculos a fuentes de conocimiento abiertas; y se implementa sobre él la arquitectura de recuperación aumentada completa, con su módulo de explicabilidad y su interfaz de visualización. En el plano metodológico, se diseña y ejecuta una evaluación comparativa entre varios sistemas[3] con el fin de discernir qué aporta cada estrategia a las distintas dimensiones de la calidad del feedback: la precisión del diagnóstico, la incidencia de alucinaciones, la utilidad formativa y la trazabilidad. Esta comparación permite, más allá de validar la propuesta global, comprender el comportamiento diferencial de cada técnica y razonar sobre su complementariedad.

Debo, en este punto inaugural y como cuestión de integridad científica que recorrerá toda la memoria, precisar con claridad el carácter de los resultados que se reportarán. El anteproyecto que dio origen a este trabajo fijó una serie de objetivos cuantitativos ambiciosos: una precisión diagnóstica superior al

[3]desde el modelo de lenguaje base hasta configuraciones que combinan *fine tuning* específico y recuperación aumentada sobre el grafo

ochenta y cinco por ciento, una tasa de alucinaciones igual o inferior al cinco por ciento, una utilidad formativa de al menos cuatro sobre cinco y una trazabilidad completa. Deben leerse en su justo término. Son expectativas e hipótesis a contrastar, metas que orientaron el diseño, no logros que el trabajo dé por consumados. Los resultados efectivamente obtenidos son más modestos que esas aspiraciones y se presentan sin inflarlos ni fingir que las metas se han alcanzado. Es más, la validación realizada hasta la fecha es de naturaleza automática[4] y se ha llevado a cabo sobre datos sintéticos y muestras reducidas. El panel de docentes humanos, el análisis de concordancia entre evaluadores y el volumen amplio de entregas reales que el anteproyecto contemplaba no se han ejecutado todavía y constituyen, declaradamente, trabajo futuro. Asumir esta distinción entre lo aspirado y lo conseguido, entre la validación preliminar disponible y la validación completa pendiente, no debilita el trabajo, sino que lo sitúa donde realmente está, como una primera exploración fundamentada y reproducible de una intersección prometedora, sobre cuya base podrá levantarse una validación más exigente. El resto de la memoria está dedicado a desarrollar esta propuesta, a fundamentarla teóricamente, a describir su construcción e implementación, y a reportar y discutir, con el rigor y la franqueza que la investigación exige, lo que se ha conseguido y lo que queda por hacer.

Para que esta salvedad quede explícita desde el inicio, conviene tabular las metas cuantitativas que el anteproyecto fijó, recordando que se trata de expectativas que orientaron el diseño y no de logros que el trabajo dé por consumados.

Metas cuantitativas fijadas en el anteproyecto (expectativas a contrastar, no resultados).

Dimensión de calidad	Meta del anteproyecto
Precisión diagnóstica	> 85 %
Tasa de alucinaciones	≤ 5 %
Utilidad formativa	≥ 4 sobre 5
Trazabilidad	Completa

Fuente: elaboración propia a partir de las metas enumeradas en este capítulo.

4.6 Estructura de la memoria

El resto del documento se organiza siguiendo el esquema clásico de introducción, materiales y métodos, resultados y discusión, adaptado a la naturaleza mixta —de construcción de un artefacto y de validación empírica— de esta investigación, de modo que el lector pueda recorrerlo con una hoja de ruta clara.

Tras esta introducción, el capítulo de estado del arte revisa los antecedentes en retroalimentación automática, en modelos de lenguaje aplicados a la educación y en grafos de conocimiento. El bloque de marco teórico desarrolla a continuación los fundamentos de la web semántica, de los grafos de conocimiento, de los modelos de lenguaje y la generación aumentada por recuperación, y de la evaluación educativa. Sobre esa base, el capítulo de objetivos e hipótesis fija el objetivo general, los objetivos específicos y las hipótesis contrastables, y el capítulo de metodología describe el diseño de investigación y sus fases.

El núcleo de desarrollo documenta primero la construcción del grafo de conocimiento educativo, su inferencia, su validación con SHACL, su explotación mediante SPARQL y su enlazado con Wikidata; y después la arquitectura de generación aumentada por recuperación en sus componentes —el analizador de código, la recuperación de subgrafos, el *fine tuning*, los sistemas comparados y el módulo de explicabilidad—. El bloque de resultados y discusión presenta el diseño experimental, reporta los resultados —incluidos los hallazgos negativos—, analiza los compromisos de diseño y discute el alcance de lo obtenido.

[4] combinando métricas objetivas con un modelo de lenguaje actuando como juez

Cierran la memoria los capítulos dedicados a las implicaciones éticas, de género y de Objetivos de Desarrollo Sostenible, a las limitaciones y el trabajo futuro, y a las conclusiones, seguidos de las referencias bibliográficas y de los anexos, que recogen las figuras, las capturas de las herramientas, los fragmentos de código, los materiales de reproducibilidad y el protocolo de anotación humana.

5 Estado del arte

El trabajo que aquí presento se sitúa en la confluencia de tres tradiciones de investigación que, hasta fechas muy recientes, han evolucionado de manera relativamente independiente. Estas son la representación estructurada del conocimiento educativo mediante grafos, la generación de texto aumentada por recuperación con modelos de lenguaje masivos y el estudio del feedback automático en la enseñanza de la programación. Cada una de estas líneas aporta un cuerpo de evidencia sólido, pero también arrastra limitaciones bien documentadas que, examinadas en conjunto, dibujan con nitidez el hueco que este Trabajo de Fin de Máster pretende ocupar. En este capítulo reviso de forma crítica el estado de la cuestión en cada una de las tres líneas, no como una mera enumeración bibliográfica, sino con la intención de mostrar dónde se agotan las propuestas existentes y por qué su intersección sigue siendo un territorio escasamente explorado. La revisión concluye con la formulación explícita de las tres brechas que motivan la arquitectura propuesta y con el posicionamiento del presente trabajo respecto de ellas.

Advierto desde el principio que el objetivo de esta revisión no es agotar la literatura de cada campo —tarea inabarcable y, en gran medida, redundante con manuales y revisiones sistemáticas ya disponibles —, sino seleccionar aquellos hitos que permiten justificar las decisiones de diseño tomadas más adelante. Por ello, las tres secciones que siguen están organizadas en torno a una pregunta común, que indaga qué ofrece esta línea de trabajo a un sistema de evaluación formativa fundamentado, explicable y adaptado al nivel del estudiante, y qué le falta todavía para alcanzar ese objetivo. Esta lente pedagógica, más que una clasificación técnica exhaustiva, es la que da coherencia al capítulo.

Antes de entrar en cada una, la siguiente tabla anticipa, para las tres líneas, qué aporta cada una a un sistema de evaluación formativa fundamentado, qué limitación documenta la literatura y cómo la aborda este trabajo.

Tabla. Síntesis de las tres líneas de trabajo relacionado: qué aporta cada una, su limitación documentada y cómo la aborda este TFM.

Línea de investigación	Qué aporta (enfoque)	Limitación documentada	Cómo la aborda este TFM
Grafos de conocimiento educativos	Representación estructurada, razonable y trazable del dominio (RDF, RDFS, OWL, SPARQL, SKOS); semántica explícita, razonamiento e integración (Hogan et al., 2021).	Menos del 12 % de los trabajos los emplean para evaluación formativa; predominan la recomendación, la navegación y el modelado de prerrequisitos (Abu-Salih y Alotaibi, 2024).	Grafo canónico propio de 157 conceptos (skos:broader, relaciones N-arias y reificación con procedencia; SHACL e inferencia OWL 2 RL), concebido para el diagnóstico.
Recuperación aumentada y GraphRAG	RAG ancla las respuestas en fuentes externas y reduce las alucinaciones (≈ 47 %, Lewis et al., 2020); GraphRAG habilita el razonamiento multi-salto sobre grafos (Edge et al., 2024).	Desarrollada mayoritariamente en dominios de propósito general; las aplicaciones educativas con finalidad evaluativa son escasas (Gupta et al., 2024).	Arquitectura GraphRAG que recupera subgrafos por similitud semántica y expansión mediante SPARQL e inyecta ese conocimiento estructurado en el contexto del modelo.
Feedback automático en programación	Taxonomía del feedback (Keuning et al., 2019) y marco del feedback eficaz sobre proceso y autorregulación (Hattie y Timperley, 2007); los modelos de lenguaje añaden fluidez y personalización (Du y Daniel, 2024).	Las herramientas se quedan en la corrección superficial; los modelos de lenguaje generan feedback plausible pero incorrecto y comprometen la fiabilidad y la trazabilidad (Du y Daniel, 2024).	Combina el grafo educativo con GraphRAG para dotar al feedback de anclaje conceptual y trazabilidad, reconciliando fluidez y fiabilidad.

5.1 Grafos de conocimiento educativos

La aplicación de grafos de conocimiento al ámbito educativo se apoya en una intuición que cualquier docente reconocerá de inmediato, pues el dominio que se enseña no es un conjunto plano de hechos aislados, sino una red de conceptos interrelacionados en la que unos saberes presuponen otros, ciertos errores remiten a malentendidos sobre nociones previas y el progreso del estudiante puede describirse como un recorrido por esa red. Formalizar esa red mediante las tecnologías de la Web Semántica —RDF como modelo de datos (W3C, 2014a), RDFS y OWL para la definición de esquemas y la inferencia (W3C, 2014b, 2012), SPARQL como lenguaje de consulta (W3C, 2013) y SKOS para la organización de vocabularios jerárquicos (W3C, 2009)— promete varias ventajas frente a representaciones menos estructuradas, como la posibilidad de razonar sobre relaciones implícitas, la interoperabilidad con recursos externos a través de enlaces de datos y la trazabilidad de cada afirmación hasta su fuente. Hogan et al. (2021), en su exhaustiva sistematización del concepto de grafo de conocimiento, subrayan que el valor diferencial de estas estructuras no reside en su capacidad de almacenamiento, sino en la combinación de semántica explícita, razonamiento automatizado y capacidad de integración, tres propiedades que resultan especialmente atractivas para un dominio formativo en el que la justificación del conocimiento importa tanto como el conocimiento mismo.

La revisión sistemática más reciente y completa sobre grafos de conocimiento educativos es la de Abu-Salih y Alotaibi (2024), que constituye un punto de referencia obligado para este trabajo. Tras analizar un volumen considerable de estudios publicados en la última década, los autores documentan una expansión notable del campo en términos de número de publicaciones, pero también revelan un

desequilibrio llamativo en los usos a los que se destinan estos grafos. La inmensa mayoría de las aplicaciones se concentra en cuatro tareas. Estas son la recomendación de recursos, la búsqueda y navegación de contenidos, la construcción de itinerarios de aprendizaje y el modelado de prerrequisitos. Por el contrario, los autores constatan que menos del doce por ciento de los trabajos revisados emplean el grafo de conocimiento con fines de evaluación formativa, esto es, para diagnosticar el estado de comprensión del estudiante y devolverle información orientada a la mejora. La cifra resulta reveladora, porque el grafo se utiliza profusamente para organizar y servir contenido, pero apenas se explota su potencial diagnóstico, que es justamente donde su semántica explícita podría aportar más valor. La brecha entre lo que la tecnología permite y lo que la comunidad ha investigado es, en este punto, muy amplia.

Una de las dificultades recurrentes que señala la literatura tiene que ver con la interoperabilidad y la estandarización de estos grafos. Existen estándares de metadatos educativos consolidados —como IEEE LOM (Learning Object Metadata)— pensados para describir objetos de aprendizaje de forma uniforme y facilitar su reutilización entre plataformas. Integrar esos estándares con representaciones basadas en grafos no siempre es trivial. Qu et al. (2024) abordan esta cuestión proponiendo mecanismos para alinear esquemas de grafos de conocimiento educativos con vocabularios estandarizados como IEEE LOM, con el objetivo de que los grafos resultantes no queden aislados en silos institucionales sino que puedan conectarse con ecosistemas más amplios de recursos. Su trabajo ilustra una tensión que atraviesa todo el campo. La riqueza expresiva de un grafo construido a medida para un dominio concreto choca a menudo con la necesidad de interoperar mediante estándares más genéricos, y encontrar el equilibrio entre ambas exigencias es una decisión de diseño delicada. Para un grafo orientado a la evaluación, además, esta tensión se agudiza, porque la granularidad conceptual que requiere un diagnóstico fino rara vez está contemplada en los estándares de metadatos de objetos de aprendizaje, concebidos para describir recursos y no estados cognitivos.

Sitúo estas consideraciones en el contexto más amplio de la analítica del aprendizaje. Gašević et al. (2022) advierten que muchos sistemas analíticos en educación adolecen de una desconexión entre los datos que recogen y los modelos teóricos del aprendizaje que deberían guiar su interpretación; en otras palabras, se acumulan datos sin un marco conceptual que les confiera sentido pedagógico. Un grafo de conocimiento educativo bien diseñado puede actuar como ese marco conceptual explícito, anclando los datos observados sobre el estudiante a una ontología del dominio que hace transparentes los supuestos pedagógicos del sistema. Esta es una de las razones por las que en este trabajo se ha optado por construir un grafo canónico propio, con un esquema cuidadosamente diseñado, en lugar de reutilizar acríticamente representaciones genéricas. El grafo resultante —157 conceptos propios, con relaciones jerárquicas mediante skos:broader, relaciones N-arias para representar evaluaciones de actividades y mecanismos de reificación que registran la procedencia de cada afirmación— busca ofrecer la granularidad diagnóstica que la literatura echa en falta, al tiempo que mantiene enlaces a recursos externos como Wikidata mediante skos:exactMatch para no quedar encerrado en un silo. Se ha optado deliberadamente por skos:exactMatch en lugar de owl:sameAs para no imponer la fusión lógica de individuos propia del perfil OWL 2 RL,[5] una elección que refleja una madurez en el uso de los datos enlazados. La inferencia bajo el perfil OWL 2 RL permite, además, que consultas que sin razonamiento no devuelven concepto alguno recuperen las nociones enlazadas externamente, e ilustra de forma tangible el valor añadido de la semántica explícita.

En síntesis, la línea de los grafos de conocimiento educativos aporta a este trabajo el sustento para una representación rica, razonable y trazable del dominio, pero la literatura deja claro que el uso de estos grafos con fines de evaluación formativa está apenas iniciado y que su explotación diagnóstica fina, especialmente en combinación con generación de lenguaje natural, constituye un terreno mayoritariamente inexplorado.

[5] que identificaría el concepto propio con la entidad de Wikidata y propagaría su tipo

5.2 Recuperación aumentada y GraphRAG

La segunda línea que confluye en este trabajo es la de los modelos de lenguaje masivos y, más concretamente, las arquitecturas que los aumentan con mecanismos de recuperación de información externa. El punto de partida ineludible es la propuesta seminal de Lewis et al. (2020), que introdujo el paradigma de la Retrieval-Augmented Generation (RAG). La idea fundamental es elegante. El conocimiento paramétrico almacenado en los pesos del modelo durante el entrenamiento es opaco, difícil de actualizar y propenso a la fabricación de hechos plausibles pero falsos. En lugar de confiar exclusivamente en él, se acopla al modelo generativo un componente de recuperación que, ante una consulta, localiza fragmentos relevantes en una base documental y los inyecta en el contexto a partir del cual el modelo redacta su respuesta. De este modo, la respuesta queda anclada a fuentes externas verificables. Lewis et al. (2020) demostraron empíricamente que este enfoque reduce de forma sustancial la tendencia de los modelos a alucinar, llegando a reportar una disminución cercana al cuarenta y siete por ciento en la incidencia de alucinaciones en tareas intensivas en conocimiento. Para un sistema de evaluación educativa, en el que un diagnóstico incorrecto o una explicación inventada pueden inducir a error al estudiante y minar la confianza en la herramienta, esta capacidad de fundamentar las respuestas en una fuente externa resulta especialmente valiosa.

El problema de las alucinaciones que RAG viene a mitigar ha sido extensamente analizado en la literatura crítica sobre modelos de lenguaje. Bender et al. (2021), en su conocida advertencia sobre los riesgos de los modelos lingüísticos de gran tamaño, recuerdan que estos sistemas son, en esencia, máquinas de modelar la distribución estadística de secuencias de palabras, sin comprensión genuina ni acceso a un referente externo que les permita distinguir lo verdadero de lo simplemente verosímil. Zhang et al. (2023) y Zhu et al. (2023) profundizan en la caracterización y el origen de las alucinaciones, mostrando que estas no son un defecto accidental susceptible de corrección trivial, sino una propiedad emergente del propio mecanismo de generación probabilística. Tal literatura justifica que cualquier aplicación de modelos de lenguaje en un contexto de alto riesgo pedagógico no pueda descansar únicamente en el conocimiento paramétrico del modelo, sino que deba incorporar mecanismos de anclaje y verificación. RAG es, hoy por hoy, la familia de técnicas más madura para ese anclaje, y de ahí su centralidad en el diseño aquí propuesto.

El RAG clásico de Lewis et al. (2020), sin embargo, opera principalmente sobre colecciones de documentos no estructurados, recuperando fragmentos de texto por similitud semántica. Este enfoque es muy eficaz para responder preguntas cuya respuesta se encuentra localizada en un pasaje concreto, pero exhibe limitaciones notables cuando la respuesta requiere conectar información dispersa en múltiples fuentes o razonar sobre relaciones entre entidades, lo que se conoce como razonamiento multi-salto (multi-hop). Edge et al. (2024) abordan esta limitación con su propuesta de GraphRAG. Esta arquitectura reemplaza o complementa la recuperación sobre texto plano con una recuperación que explota una estructura de grafo construida a partir del corpus. Al organizar el conocimiento como un grafo de entidades y relaciones, GraphRAG permite responder a preguntas globales o que exigen integrar información de varios nodos conectados, superando el carácter local de la recuperación por fragmentos. Este desplazamiento del paradigma (de recuperar pasajes a recuperar subgrafos) es de la mayor relevancia para el presente trabajo, porque sugiere una vía natural para hacer dialogar la primera y la segunda línea de investigación, ya que, si el conocimiento del dominio ya está representado como un grafo educativo, la recuperación puede operar directamente sobre esa estructura y extraer el subgrafo de conceptos pertinentes para diagnosticar un error concreto.

La intersección entre grafos de conocimiento y modelos de lenguaje ha sido objeto de una atención creciente. Gupta et al. (2024) ofrecen una panorámica de las distintas formas en que ambos pueden combinarse, distinguiendo enfoques en los que el grafo enriquece al modelo en tiempo de inferencia, otros en los que el modelo contribuye a construir o completar el grafo, y arquitecturas verdaderamente híbridas en las que ambos componentes se realimentan. Su revisión deja patente que, pese al evidente interés del campo, la mayor parte de los trabajos se han desarrollado en dominios de propósito general

(respuesta a preguntas factuales, búsqueda empresarial, asistentes conversacionales) y que las aplicaciones a dominios educativos con finalidad evaluativa son comparativamente escasas. Existe, así, un terreno fértil y poco roturado en la conjunción de GraphRAG con grafos de conocimiento educativos orientados al diagnóstico.

En cuanto a la realización técnica de estas arquitecturas, dejo constancia de una decisión de implementación que el lector encontrará desarrollada más adelante y que conviene anticipar aquí. El anteproyecto contemplaba una pila tecnológica basada en el modelo de embeddings all-MiniLM-L6-v2, el índice vectorial FAISS y la base de datos de grafos Neo4j. Durante el desarrollo, sin embargo, la implementación real se decantó por el modelo de embeddings nomic-embed-text, por la biblioteca rdflib (RDFLib Team, s.f.) en combinación con el almacén GraphDB (Ontotext, s.f.) para la gestión del grafo RDF, y por la ejecución de modelos de lenguaje en local mediante Ollama. La divergencia no responde a un capricho, sino a una conjunción de razones prácticas. La adopción de un grafo plenamente conforme con los estándares de la Web Semántica (RDF, OWL, SPARQL, SHACL) aconsejaba un almacén RDF nativo con capacidad de razonamiento e validación de restricciones (W3C, 2017) antes que un grafo de propiedades como el de Neo4j; la disponibilidad de nomic-embed-text a través de la misma infraestructura de Ollama simplificaba el despliegue íntegramente local; y este despliegue local respondía además a exigencias de soberanía de los datos que se discuten en el capítulo dedicado a las consideraciones éticas. Prefiero dejar constancia de la decisión tal como se produjo y no presentar la arquitectura final como si hubiera sido la planificada desde el inicio.

La siguiente tabla resume esa divergencia entre la pila tecnológica que planificaba el anteproyecto y la que finalmente se implementó.

Tabla. Divergencia entre la pila tecnológica del anteproyecto y la de la implementación final.

Componente	Anteproyecto	Implementación final
Modelo de embeddings	all-MiniLM-L6-v2	nomic-embed-text
Índice vectorial	FAISS	—
Almacén del grafo	Neo4j (grafo de propiedades)	rdflib + GraphDB (Ontotext), almacén RDF nativo
Ejecución del modelo de lenguaje	—	Ollama (despliegue en local)

La línea del RAG y GraphRAG aporta el mecanismo central para fundamentar las respuestas del modelo de lenguaje en una fuente externa verificable, mitigar las alucinaciones y habilitar el razonamiento sobre estructuras relacionales. Su combinación con un grafo educativo de granularidad diagnóstica es, como muestra la literatura revisada, una posibilidad prometedora y todavía poco explorada.

5.3 Feedback automático en la enseñanza de la programación

La tercera línea que vertebra este trabajo es el estudio del feedback automático en la enseñanza de la programación, un campo con una larga trayectoria y una literatura abundante. El punto de referencia fundamental para ordenar este territorio es la revisión sistemática de Keuning et al. (2019), que analiza un amplio conjunto de herramientas de generación automática de feedback para ejercicios de programación a lo largo de varias décadas. Su contribución más influyente es una taxonomía del tipo de feedback que ofrecen estas herramientas, que va desde el más superficial —indicar simplemente si una solución es correcta o incorrecta, o señalar errores de sintaxis— hasta el más sofisticado, que incluye pistas sobre cómo proceder, identificación del error conceptual subyacente y orientación sobre los pasos siguientes. El hallazgo más significativo de Keuning et al. (2019) es de naturaleza diagnóstica respecto al propio campo. La inmensa mayoría de las herramientas existentes se concentran en los niveles inferiores de la taxonomía y ofrecen feedback sobre corrección y errores de bajo nivel, mientras que el feedback de tipo formativo y conceptual —el que realmente ayuda al estudiante a comprender por qué se ha equivocado y cómo subsanar la laguna en su comprensión— está mucho menos representado. En otras palabras,

abundan los sistemas que dicen al estudiante que algo está mal, pero escasean los que le explican por qué y le ayudan a aprender de ello.

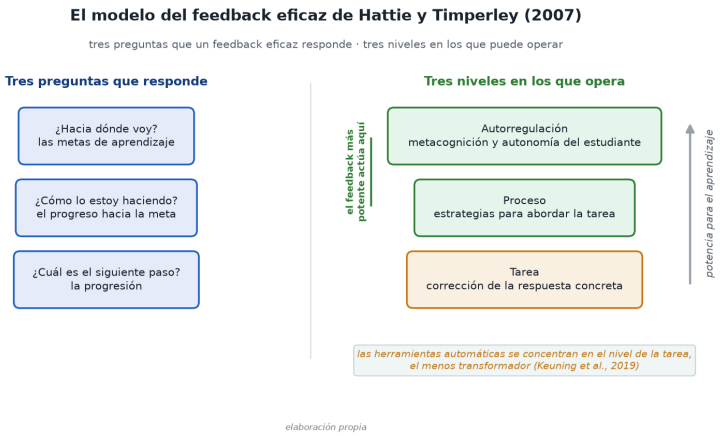

Figura 4. El modelo del feedback eficaz de Hattie y Timperley (2007): tres preguntas que el feedback responde y tres niveles en los que opera.

Este diagnóstico de Keuning et al. (2019) cobra todo su sentido cuando se contrasta con la teoría del feedback educativo. El marco de referencia obligado es el de Hattie y Timperley (2007), cuyo modelo distingue tres preguntas que un feedback eficaz debe responder (¿hacia dónde voy?, ¿cómo lo estoy haciendo? y ¿cuál es el siguiente paso?) y tres niveles en los que puede operar, que son la tarea, el proceso y la autorregulación, según sintetiza la Figura 4. Su revisión, una de las más citadas en toda la investigación educativa, establece que el feedback más potente para el aprendizaje no es el que se limita a corregir la tarea concreta, sino el que actúa sobre el proceso y la autorregulación del estudiante, ayudándole a desarrollar estrategias transferibles. Confrontar la taxonomía empírica de Keuning et al. (2019) con el marco normativo de Hattie y Timperley (2007) revela con claridad la brecha del campo, donde las herramientas automáticas existentes se quedan mayoritariamente en el nivel de la tarea, justamente el menos transformador desde el punto de vista del aprendizaje. El reto, en suma, no es generar más feedback, sino generar feedback de mayor calidad pedagógica.

La investigación específica sobre las dificultades de los estudiantes de programación aporta el sustrato empírico que explica por qué este reto es tan exigente. El célebre estudio multinacional de McCracken et al. (2001) documentó que un porcentaje alarmante de estudiantes que habían superado un primer curso de programación era incapaz de resolver problemas de programación elementales, evidenciando una desconexión profunda entre la apariencia de progreso y la comprensión real. Watson y Li (2014) analizaron las tasas de fracaso en cursos introductorios de programación a escala internacional, confirmando que se trata de un problema persistente y generalizado. Esta dificultad estructural de la materia subraya la importancia de un feedback que no se limite a la superficie sintáctica del código, sino que alcance los malentendidos conceptuales que subyacen a los errores. Marwan et al. (2020) aportan a este respecto una evidencia matizada. En su estudio sobre el efecto de añadir explicaciones a las pistas automáticas en entornos de programación, encontraron que la mera presencia de feedback no garantiza el aprendizaje. La forma en que se formula la explicación condiciona de manera decisiva su utilidad. Este hallazgo es una advertencia pertinente para cualquier sistema que, como el aquí propuesto, aspire a generar explicaciones automáticas, porque la calidad y la pertinencia pedagógica de la explicación importan tanto o más que su mera existencia.

La irrupción de los modelos de lenguaje masivos ha transformado radicalmente el panorama de este campo en los últimos años. Modelos capaces de comprender y generar código, como los descritos por Chen et al. (2021) a propósito de la evaluación de modelos entrenados sobre código, han abierto la posibilidad de generar feedback en lenguaje natural de una fluidez y una aparente pertinencia inéditas. Du y Daniel (2024) revisan el uso de modelos de lenguaje masivos para la generación de feedback en

educación en programación, sistematizando las oportunidades y los riesgos de este nuevo paradigma. Entre las oportunidades destacan la capacidad de producir explicaciones personalizadas, adaptadas al código concreto del estudiante y formuladas en un registro accesible. Entre los riesgos, Du y Daniel (2024) señalan con insistencia la propensión de estos modelos a generar feedback plausible pero incorrecto, a diagnosticar errores que no existen o a pasar por alto los que sí están presentes, y la dificultad de verificar y fundamentar las afirmaciones que el modelo produce. Su revisión confirma, en suma, que los modelos de lenguaje resuelven el problema de la fluidez y la personalización que aquejaba a las herramientas anteriores, pero introducen un problema nuevo y grave de fiabilidad y trazabilidad, que es el que las arquitecturas RAG y GraphRAG examinadas en la sección precedente están en condiciones de mitigar.

Figura 5. Confluencia de las tres líneas de investigación: cada una aporta una pieza y su intersección define la contribución del trabajo.

Este es el punto en el que las tres líneas confluyen de manera natural, tal como ilustra la Figura 5. El feedback automático en programación necesita superar el nivel de la corrección superficial y alcanzar el diagnóstico conceptual (la lección de Keuning et al., 2019, y de Hattie y Timperley, 2007); los modelos de lenguaje aportan la fluidez necesaria para ese salto pero comprometen la fiabilidad (la advertencia de Du y Daniel, 2024); y la combinación de un grafo de conocimiento educativo con una arquitectura GraphRAG ofrece el anclaje conceptual y la trazabilidad que permitirían reconciliar la fluidez con la fiabilidad. El presente trabajo se ubica deliberadamente en esa confluencia.

5.4 Brechas identificadas y posicionamiento del trabajo

Tres brechas en la literatura y cómo las aborda este trabajo

Brecha identificada	Evidencia en la literatura	Cómo la aborda este TFM
Infrautilización de grafos de conocimiento en educación	menos del 12 % de los trabajos los emplean (Abu-Salih y Alotaibi, 2024)	EKG de Python con 157 conceptos y taxonomía formal RDF/OWL
Escasa aplicación de GraphRAG a la evaluación educativa	GraphRAG es reciente y se aplica sobre todo a QA general (Edge et al., 2024; Lewis et al., 2020)	recuperación de subgrafos por SPARQL como contexto del LLM
Tensión entre fluidez y fiabilidad/trazabilidad del feedback	los LLM son fluidos pero poco fiables; el feedback debe ser trazable (Keuning et al., 2019; Hattie y Timperley, 2007)	feedback anclado al grafo, con procedencia explícita

elaboración propia

Figura 6. Tres brechas de la literatura y cómo las aborda el trabajo.

La revisión precedente permite formular con precisión tres brechas que, en su conjunto, justifican la pertinencia de este Trabajo de Fin de Máster y delimitan su contribución. La Figura 6 las sintetiza y anticipa el modo en que el trabajo aborda cada una.

La primera brecha es la infrautilización de los grafos de conocimiento educativos con fines de evaluación formativa. Como documentan Abu-Salih y Alotaibi (2024), menos del doce por ciento de los trabajos sobre grafos de conocimiento educativos los emplean para diagnosticar el estado de comprensión del estudiante, concentrándose la inmensa mayoría en tareas de recomendación, navegación y organización

de contenidos. La granularidad conceptual y la capacidad de razonamiento que ofrecen estas estructuras, que las harían especialmente idóneas para un diagnóstico fino, permanecen en buena medida sin explotar en el contexto evaluativo. Este trabajo responde a esa brecha construyendo un grafo de conocimiento canónico orientado específicamente al diagnóstico —157 conceptos, jerarquía mediante skos:broader, relaciones N-arias y reificación con procedencia, conforme con SHACL y con inferencia bajo OWL 2 RL—, concebido desde el inicio como instrumento de evaluación y no como simple repositorio de contenidos.

La segunda brecha es la escasa aplicación de las arquitecturas GraphRAG, y en general de la conjunción entre grafos de conocimiento y modelos de lenguaje, al dominio educativo con finalidad evaluativa. Si bien RAG (Lewis et al., 2020) ha demostrado su capacidad para reducir las alucinaciones de forma sustancial y GraphRAG (Edge et al., 2024) ha extendido esa capacidad al razonamiento multi-salto sobre estructuras relacionales, la panorámica de Gupta et al. (2024) evidencia que estas técnicas se han desarrollado mayoritariamente en dominios de propósito general, quedando las aplicaciones educativas evaluativas comparativamente desatendidas. Este trabajo aborda esa brecha diseñando una arquitectura GraphRAG que recupera subgrafos del grafo educativo por similitud semántica y expansión mediante SPARQL, e inyecta ese conocimiento estructurado en el contexto del modelo de lenguaje para fundamentar y trazar el diagnóstico.

La tercera brecha es la tensión, hasta ahora no resuelta de forma satisfactoria, entre la fluidez del feedback que generan los modelos de lenguaje y la fiabilidad y trazabilidad que exige el contexto educativo. Las herramientas tradicionales de feedback en programación se quedan en la corrección superficial (Keuning et al., 2019) cuando la teoría del feedback reclama actuar sobre el proceso y la comprensión (Hattie y Timperley, 2007); los modelos de lenguaje aportan la fluidez para ese salto pero, como advierten Du y Daniel (2024), comprometen gravemente la fiabilidad. Este trabajo intenta reconciliar ambos extremos, pues la augmentación mediante el grafo busca preservar la fluidez y la personalización del feedback generado por el modelo, al tiempo que le confiere el anclaje conceptual y los marcadores de procedencia que lo hacen verificable y, por tanto, pedagógicamente fiable.

El posicionamiento del trabajo respecto de estas tres brechas se concreta en su hipótesis central, H1, según la cual la integración de un grafo de conocimiento educativo con un modelo de lenguaje mediante una arquitectura RAG mejora la calidad pedagógica del feedback frente al modelo sin augmentación, hipótesis que se descompone en las sub-hipótesis relativas a la precisión diagnóstica, la reducción de alucinaciones, la utilidad formativa y la trazabilidad. Subrayo que las metas cuantitativas que el anteproyecto fijó para estas dimensiones constituyen hipótesis ambiciosas a contrastar y no resultados que se den por alcanzados; los resultados efectivamente obtenidos, más modestos y matizados, se reportan en los capítulos correspondientes. La evidencia recogida sobre un conjunto de cincuenta casos held-out sugiere, de hecho, una conclusión interesante y no trivial, ya que el modelo afinado mediante QLoRA mejora la clasificación del error y la explicación técnica, mientras que la augmentación GraphRAG mejora la identificación del concepto, la trazabilidad y la calidad pedagógica, lo que apunta a una complementariedad entre ambas técnicas que motiva la exploración de un sistema híbrido. Esa complementariedad se confirmó al incorporar el Sistema D, que combina ambas técnicas. En la evaluación ya completada sobre los cincuenta casos held-out, el Sistema D resultó el mejor, pues gana o empata en las siete dimensiones consideradas. Insisto, no obstante, en que ninguna de las metas cuantitativas del anteproyecto llega a alcanzarse (el mejor sistema se queda en 0,76 en la métrica de categoría) y en que la evaluación carece todavía de panel humano, que queda como trabajo futuro. Esta observación ilustra que el espacio de diseño abierto por la confluencia de las tres líneas revisadas es rico y que el presente trabajo, lejos de cerrarlo, aspira a contribuir a roturarlo con rigor.

6 Objetivos e hipótesis de investigación

En este capítulo formulo con precisión qué pretendo lograr en el presente Trabajo de Fin de Máster y bajo qué supuestos teóricos se sostiene la propuesta. Antes de entrar en los detalles debo hacer una advertencia metodológica que vertebra todo el documento y que considero una cuestión de integridad científica ineludible. Los objetivos y las hipótesis que aquí enuncio fueron concebidos en el anteproyecto como expectativas ambiciosas y como criterios de éxito *a priori*, no como conclusiones alcanzadas. Las cifras umbral que aparecerán a lo largo del capítulo constituyen metas declaradas y, en el caso de las hipótesis, predicciones contrastables.[6] No describen resultados obtenidos. La distinción es deliberada y se mantiene de forma estricta. Este capítulo enuncia el horizonte que la investigación se propuso, mientras que el capítulo de resultados reportará lo efectivamente medido, que es más modesto y que en ningún caso se infla para fingir que las metas se cumplieron. Quien lea estos objetivos debe entenderlos, por tanto, como un contrato de propósito y no como un acta de logros.

6.1 Objetivo general

El objetivo general de este trabajo es **diseñar, implementar y validar una arquitectura que integre un grafo de conocimiento educativo (Educational Knowledge Graph, EKG) con modelos de lenguaje masivos mediante una aproximación de generación aumentada por recuperación (Retrieval-Augmented Generation, RAG), con el fin de producir retroalimentación formativa en la enseñanza de la programación que sea, simultáneamente, fundamentada, explicable y adaptada al nivel del estudiante**. Cada uno de esos tres adjetivos comporta una exigencia concreta sobre el sistema. Que la retroalimentación sea *fundamentada* significa que las afirmaciones diagnósticas que el sistema emite sobre el código de un aprendiz no deben surgir de la mera fluidez generativa del modelo de lenguaje, sino que han de poder anclarse en conocimiento curricular estructurado y verificable; el grafo de conocimiento actúa aquí como sustrato semántico que delimita y respalda lo que el modelo puede legítimamente afirmar. Que sea *explicable* implica que el sistema no se limite a entregar un veredicto, sino que exponga el camino de razonamiento —qué conceptos del currículo y qué relaciones entre ellos se han movilizado para llegar a ese diagnóstico—, de modo que tanto el estudiante como el docente puedan auditar la procedencia de cada afirmación. Y que sea *adaptada al nivel* exige que la retroalimentación se ajuste a la posición del aprendiz dentro de la secuencia de aprendizaje, reconociendo qué conceptos previos domina o aún no domina, en lugar de ofrecer una corrección uniforme e indiferente al recorrido formativo.

Esta formulación no es caprichosa. Recoge una tensión bien documentada en la literatura sobre retroalimentación educativa y sobre modelos de lenguaje aplicados a la enseñanza. Por un lado, la investigación clásica sobre *feedback* sostiene que la retroalimentación eficaz no es la que únicamente señala el error, sino la que sitúa al aprendiz respecto a una meta, le indica el estado de su progreso y le orienta sobre el paso siguiente, articulando las preguntas «¿hacia dónde voy?», «¿cómo voy?» y «¿qué sigue?» (Hattie & Timperley, 2007). Por otro lado, los modelos de lenguaje masivos, pese a su notable competencia generativa, son propensos a producir afirmaciones plausibles pero infundadas (el fenómeno de las alucinaciones) y han sido caracterizados como sistemas que recombinan formas lingüísticas sin un anclaje verificable en el significado (Bender et al., 2021). La generación aumentada por recuperación se propuso como vía para mitigar esa fragilidad, condicionando la generación a evidencia recuperada de una fuente externa (Lewis et al., 2020); y los grafos de conocimiento ofrecen una representación de esa evidencia que es a la vez estructurada, navegable y dotada de semántica formal (Hogan et al., 2021). El objetivo general combina, en consecuencia, dos tradiciones —la del diseño de retroalimentación formativa y la de la integración de conocimiento simbólico con generación neuronal— en el dominio específico de la enseñanza de la programación, donde la detección automática de errores y la generación de pistas son problemas con una historia investigadora propia (Keuning et al., 2019; Marwan et al., 2020).

[6] una precisión diagnóstica del orden del 85 %, una tasa de alucinaciones por debajo del 5 %, una utilidad formativa percibida igual o superior a 4,0 sobre 5, o una trazabilidad del 100 %

Del objetivo general se derivan cinco objetivos específicos, que estructuran las tres fases metodológicas del trabajo (construcción del EKG, diseño de la arquitectura RAG y validación experimental) y que se detallan a continuación junto con las métricas de éxito que se establecieron para cada uno. La Figura 7 resume esa correspondencia entre objetivos, metas declaradas y evidencia disponible. Insisto en que dichas métricas se enuncian como criterios de evaluación propuestos, no como resultados conseguidos.

6.2 Pregunta principal de investigación y subpreguntas

Los objetivos que acabo de anunciar no flotan en el vacío, sino que responden a una única pregunta que vertebra toda la investigación y que enuncio aquí en los mismos términos en que la formula el artículo derivado de este trabajo, de modo que memoria y *paper* compartan una sola estructura interrogativa y no dos relatos divergentes. La pregunta principal de investigación, que designo como RQ, se formula así.

¿Cuándo, y bajo qué tipo de evaluación, aumentar un modelo de lenguaje masivo (LLM) con un grafo de conocimiento educativo o con fine tuning eficiente en parámetros mejora realmente la retroalimentación formativa en la enseñanza de la programación?

El matiz importa. No me pregunto si la augmentación ayuda en abstracto, sino *cuándo* lo hace y *bajo qué instrumento de medida* puede afirmarse que lo hace, porque, como se verá, la respuesta depende tanto de la condición experimental como de la lente evaluadora que se emplee. Esta pregunta principal se descompone en cuatro subpreguntas, que designo como SQ y que la operacionalizan; cada una se corresponde, respectivamente, con las preguntas PI1 a PI4 ya etiquetadas en el artículo, de manera que no existan dos numeraciones ni dos formulaciones distintas entre ambos documentos. Son las siguientes.

- **SQ1.** ¿Es posible construir un grafo de conocimiento educativo validado con SHACL (cero violaciones) y cerrado bajo razonamiento OWL 2 RL, que cumpla objetivos estructurales explícitos (≥ 150 conceptos) y enlace de forma verificable con una base de conocimiento abierta como Wikidata? Esta subpregunta interroga directamente al objetivo O1, pues es la construcción y validación del EKG la que debe darle respuesta.
- **SQ2.** ¿Mejoran el anclaje en el grafo (GraphRAG), el fine tuning con QLoRA y su combinación híbrida la calidad objetiva de la retroalimentación frente a un LLM base, y se transfiere esa ventaja obtenida en distribución al código real de estudiantes (fuera de distribución)? Esta subpregunta recae sobre los objetivos O2 y O3, ya que enfrenta las distintas configuraciones de la arquitectura y mide su efecto.
- **SQ3.** ¿Constituye un juez LLM un instrumento fiable y válido para puntuar la calidad de esta retroalimentación cuando se contrasta su validez de criterio con la de un panel humano? Esta subpregunta, junto con la siguiente, desborda el alcance estrictamente comparativo del objetivo O3 y abre la reflexión metodológica sobre los propios instrumentos de medida.
- **SQ4.** ¿Es un panel humano promediado un instrumento fiable y discriminante de la calidad de la retroalimentación, incluso cuando la concordancia entre evaluadores individuales es baja?

Conviene advertir que SQ3 y SQ4 reclaman evidencia (el juez LLM contrastado con un panel humano y el análisis de concordancia entre evaluadores) que, según declaro en el objetivo O3 y reitero en el capítulo de resultados, este trabajo aborda solo de forma preliminar; la validación humana a gran escala que esas subpreguntas presuponen permanece como trabajo futuro. Enunciarlas aquí no es un exceso, sino la manera de fijar el horizonte interrogativo completo al que la investigación aspira y de preservar la coherencia con el artículo. El objetivo O5, dedicado al análisis de los compromisos de diseño, no agota una subpregunta propia, sino que atraviesa de forma transversal la RQ al precisar el «cuándo» y el «a qué coste» de las mejoras que SQ2 examina.

6.3 Objetivos específicos

Objetivos del TFM, metas declaradas y evidencia

Objetivo	Meta declarada (anteproyecto)	Evidencia / estado real
Construir el EKG en RDF/OWL	≥150 conceptos, ≥400 relaciones, SHACL sin violaciones	157 conceptos · 174 prerrequisitos · 30 skos:exactMatch · SHACL conforme
Integrar el EKG con un LLM mediante RAG	arquitectura GraphRAG funcional	ast+radon · nomic-embed-text · SPARQL · Ollama (implementado)
Generar feedback trazable y explicable	trazabilidad ~100 %	subgrafo + procedencia (FastAPI + Cytoscape.js)
Evaluar precisión y alucinaciones	precisión ~85 %, alucinaciones ≤5 %, utilidad ≥4,0/5	n=50 held-out; los targets no se alcanzan (se reporta lo real)
Comparar cuatro sistemas (A/B/C/D)	diseño factorial afinado × grafo	D ≥ {B,C} > A (acierto de categoría 0,76 vs 0,26)

elaboración propia

Figura 7. Objetivos, metas declaradas y evidencia. *Fuente: elaboración propia.*

6.3.1 O1. Construir un grafo de conocimiento educativo del dominio de la programación

El primer objetivo consiste en construir un grafo de conocimiento educativo que modele el dominio curricular de la programación introductoria con una cobertura suficiente para sustentar el razonamiento diagnóstico posterior. El EKG debe representar los conceptos del dominio —desde nociones elementales como variable, asignación o tipo de dato hasta construcciones de mayor nivel como funciones, recursión o estructuras de datos—, las relaciones pedagógicas entre ellos (en particular las dependencias de prerrequisito y las relaciones de generalización o especialización) y los errores típicos asociados a cada concepto, de manera que el grafo describa tanto el saber como las formas características en que ese saber se aprende mal. La construcción se apoya en los estándares de la Web Semántica: RDF como modelo de datos (W3C, 2014a), RDFS y OWL 2 para la capa de esquema y la semántica de inferencia (W3C, 2014b, 2012), SKOS para la organización conceptual jerárquica (W3C, 2009) y SHACL para la validación de la integridad estructural del grafo (W3C, 2017).

La métrica de éxito declarada para este objetivo, fijada en el anteproyecto, era alcanzar al menos 150 conceptos y al menos 400 relaciones, junto con la exigencia de que el grafo superara la validación SHACL sin violaciones y de que la inferencia OWL aportara valor efectivo respecto al grafo afirmado. Se trata de un umbral cuantitativo de cobertura mínima, porque un grafo demasiado escueto no podría dar soporte a un diagnóstico fino, mientras que la conformidad SHACL garantiza que la estructura es coherente y que las restricciones de modelado se respetan. La exigencia de que la inferencia aporte valor responde a la idea de que un grafo de conocimiento no es solo un repositorio de aserciones explícitas, sino una base sobre la que un razonador puede derivar conocimiento implícito y enlazar con recursos externos.

6.3.2 O2. Diseñar e implementar la arquitectura RAG sobre el grafo

El segundo objetivo es diseñar e implementar la arquitectura de generación aumentada por recuperación que conecta el código del estudiante con el grafo de conocimiento y con el modelo de lenguaje. Esta arquitectura se concibe como una tubería de varios componentes encadenados. En primer lugar, un **analizador de código** que, a partir del árbol de sintaxis abstracta (AST) del programa entregado por el aprendiz, extraiga las construcciones empleadas y caracterice su complejidad, de modo que el sistema disponga de una representación estructural y no meramente textual del código. En segundo lugar, un **indexador de embeddings** que represente conceptos y fragmentos en un espacio vectorial para permitir la recuperación por similitud semántica. En tercer lugar, un **motor de recuperación de subgrafos** que, partiendo de los conceptos detectados, seleccione la porción relevante del EKG y la expanda mediante consultas estructuradas para incorporar prerrequisitos, errores asociados y conceptos vecinos. En cuarto lugar, un **generador de prompts** que componga la entrada al modelo integrando el código, el subgrafo recuperado y las instrucciones de la tarea. Y finalmente, la **integración con un modelo de lenguaje** que produzca la retroalimentación. Esta cadena materializa el patrón RAG (Lewis et al., 2020) en su variante orientada a grafos, en la línea de los trabajos recientes sobre GraphRAG que aprovechan la estructura relacional del conocimiento para guiar la recuperación (Edge et al., 2024; Qu et al., 2024). La siguiente tabla recoge las etapas encadenadas de esa tubería y la función que cada componente desempeña en el procesamiento.

Etapa de la tubería	Función en el procesamiento
Analizador de código	Extrae del árbol de sintaxis abstracta (AST) del programa las construcciones empleadas y caracteriza su complejidad, aportando una representación estructural y no meramente textual del código.
Indexador de embeddings	Representa conceptos y fragmentos en un espacio vectorial para permitir la recuperación por similitud semántica.
Motor de recuperación de subgrafos	Selecciona la porción relevante del EKG a partir de los conceptos detectados y la expande mediante consultas estructuradas, incorporando prerrequisitos, errores asociados y conceptos vecinos.
Generador de prompts	Compone la entrada al modelo integrando el código, el subgrafo recuperado y las instrucciones de la tarea.
Integración con el modelo de lenguaje	Produce la retroalimentación a partir del prompt compuesto.

La métrica de éxito declarada para este objetivo es de naturaleza funcional, pues exige que la tubería completa opere de extremo a extremo sobre entregas reales de código, que la recuperación de subgrafos devuelva conocimiento pertinente al error analizado y que el sistema funcione íntegramente con despliegue local, sin dependencia de servicios externos, por las razones de soberanía y privacidad de los datos del estudiante que se desarrollan en el capítulo de consideraciones éticas. Anticipo aquí una decisión de implementación que justifico en detalle más adelante. El anteproyecto contemplaba el uso de la biblioteca de embeddings all-MiniLM-L6-v2 junto con FAISS para la indexación vectorial y Neo4j como almacén de grafo, mientras que la implementación efectiva adoptó nomic-embed-text para los embeddings y la combinación de rdflib con GraphDB para la gestión del grafo (Ontotext, s.f.). Este giro no es una desviación arbitraria, sino el resultado de priorizar la coherencia con los estándares RDF/OWL del primer objetivo y la operatividad del despliegue local; lo señalo ya en este punto porque dar cuenta de las decisiones de diseño forma parte del compromiso del trabajo.

6.3.3 O3. Validar experimentalmente la calidad pedagógica de la retroalimentación

El tercer objetivo es someter la propuesta a una validación experimental que permita contrastar si la integración de conocimiento mejora efectivamente la retroalimentación. El diseño previsto es de tipo comparativo, enfrentando una condición de modelo de lenguaje sin augmentación frente a las condiciones que incorporan el grafo y el *fine tuning*, y evaluando la salida de cada sistema mediante una rúbrica de cinco dimensiones que captura los aspectos de la calidad pedagógica que la literatura considera relevantes: la corrección del diagnóstico, la identificación del error, la pertinencia conceptual, la trazabilidad de las afirmaciones y la utilidad formativa de las indicaciones. La evaluación se realiza sobre datos sintéticos generados mediante plantillas, con una separación estricta entre los casos de entrenamiento y los casos de prueba (los denominados *held-out*) para evitar fugas de información que contaminarían la medición.

La métrica de éxito declarada para este objetivo retoma los umbrales ambiciosos del anteproyecto, que se ofrecen aquí como aspiraciones evaluables: una precisión diagnóstica del orden del 85 % o superior, una proporción de alucinaciones del 5 % o inferior, una utilidad formativa media de al menos 4,0 sobre 5 y una trazabilidad cercana a la totalidad de las afirmaciones. Repito, porque es esencial, que estos valores son el listón que se quiso medir, no la altura que se saltó. Hay además una limitación de alcance que debe declararse desde ya con toda claridad y que matiza el sentido de este objetivo. La validación efectivamente ejecutada es **automática**, basada en métricas objetivas y en un juez constituido por un modelo de lenguaje de mayor tamaño actuando como evaluador, y se aplica sobre datos **sintéticos**. El panel de tres a cinco docentes humanos que el anteproyecto preveía, el cálculo del coeficiente de correlación intraclase para medir la concordancia entre jueces y el conjunto de trescientas entregas

reales de estudiantes **no se han ejecutado en este trabajo** y constituyen trabajo futuro explícito. Presentar la evaluación automática como si equivaliera a la validación humana planificada sería una distorsión que este trabajo rechaza.

6.3.4 O4. Desarrollar un módulo de explicabilidad y trazabilidad

El cuarto objetivo es dotar al sistema de un módulo de explicabilidad que haga visible y auditable el razonamiento subyacente a cada retroalimentación. La explicabilidad no se entiende aquí como un añadido cosmético, sino como una propiedad constitutiva de un sistema educativo responsable, porque si la retroalimentación ha de servir para que el estudiante aprenda y para que el docente confíe en la herramienta, es imprescindible que ambos puedan inspeccionar de dónde procede cada afirmación. El módulo debe ofrecer, por una parte, la **visualización del subgrafo recuperado**, mostrando gráficamente qué conceptos y qué relaciones del EKG se han movilizado para producir el diagnóstico, con codificación visual que facilite la lectura; y, por otra parte, **marcadores de procedencia** que vinculen cada afirmación de la retroalimentación con su soporte en el grafo. Esta orientación enlaza con la noción de procedencia propia de la Web Semántica y con la idea de que los grafos de conocimiento permiten una transparencia que los modelos neuronales por sí solos no garantizan (Hogan et al., 2021; Abu-Salih & Alotaibi, 2024).

La métrica de éxito declarada para este objetivo es que el sistema sea capaz de exhibir, para cada caso de retroalimentación, el subgrafo de evidencia y los marcadores de procedencia correspondientes, de modo que un docente pueda verificar la fundamentación de las afirmaciones del sistema. El ideal de trazabilidad completa (que toda afirmación diagnóstica esté respaldada por un elemento identificable del grafo) se enuncia como aspiración de diseño y como una de las dimensiones que la evaluación del objetivo anterior pretende medir, sin presuponer que se alcance en su totalidad.

6.3.5 O5. Analizar los compromisos de diseño del sistema

El quinto objetivo es realizar un análisis de los compromisos (*trade-offs*) que gobiernan el comportamiento del sistema, con el fin de comprender cómo afectan al resultado las decisiones de configuración. Tres ejes de variación resultan especialmente relevantes. El primero es el **tamaño y la naturaleza del modelo de lenguaje**, que enfrenta el coste computacional con la calidad de la generación. El segundo es la **profundidad de la recuperación** sobre el grafo, es decir, cuánto subgrafo se incorpora al contexto, donde una recuperación demasiado escueta empobrece la fundamentación, mientras que una demasiado amplia diluye la señal y satura la ventana de contexto. El tercero es la **cuantización** del modelo y, en términos más generales, el papel del *fine tuning* eficiente en parámetros, que permite especializar un modelo con un coste de memoria reducido. Este último eje conecta con las técnicas de adaptación de bajo rango y de cuantización a cuatro bits que la literatura reciente ha consolidado (Hu et al., 2021; Dettmers et al., 2023), y que en este trabajo se emplean para el *fine tuning* de uno de los sistemas comparados.

La métrica de éxito declarada para este objetivo es de carácter analítico más que de umbral, pues se pretende caracterizar de forma razonada cómo cada eje de configuración incide en la calidad pedagógica y en el coste, y extraer recomendaciones de diseño fundamentadas en la evidencia recogida. El éxito no se mide aquí por superar un número, sino por la solidez del análisis comparativo y por su utilidad para orientar despliegues futuros del sistema en contextos educativos reales.

6.4 Hipótesis de investigación

Sobre la base de estos objetivos, el trabajo articula una hipótesis principal y cuatro sub-hipótesis que la operacionalizan. Antes de enunciarlas insisto en que se trata de **hipótesis contrastables**, pues cada una predice una dirección de efecto y va asociada a un umbral cuantitativo que funciona como expectativa a evaluar, no como hecho establecido. La estructura experimental descrita en el objetivo O3 (la comparación entre sistemas y la rúbrica de cinco dimensiones) es el dispositivo que permite poner a prueba estas predicciones. Que las hipótesis sean refutables es una virtud y no un defecto, porque el valor

científico del trabajo no depende de que todos los umbrales se confirmen, sino de que la confrontación con los datos se realice con rigor y de que sus resultados, sean cuales fueren, se reporten sin retoques.

6.4.1 Hipótesis principal H1

La hipótesis principal sostiene que **la integración de un grafo de conocimiento educativo con un modelo de lenguaje masivo mediante una arquitectura RAG mejora la calidad pedagógica de la retroalimentación generada, en comparación con el mismo modelo de lenguaje operando sin augmentación**. La intuición teórica que la sustenta es doble. Por un lado, condicionar la generación a evidencia recuperada de una fuente externa reduce la dependencia de la memoria paramétrica del modelo y, con ella, la propensión a producir afirmaciones infundadas. Es el mecanismo que la RAG fue diseñada para mitigar (Lewis et al., 2020), y adquiere especial relevancia a la luz de las limitaciones de fundamentación de los modelos de lenguaje (Bender et al., 2021). Por otro lado, el conocimiento que se inyecta no es texto arbitrario, sino un subgrafo curricular estructurado que codifica explícitamente prerrequisitos, errores típicos y relaciones conceptuales; esa estructura aporta al modelo una representación del dominio que la generación libre no posee y que se alinea con lo que la literatura considera retroalimentación formativa de calidad, orientada a la meta, al progreso y al paso siguiente (Hattie & Timperley, 2007). La hipótesis principal se descompone en cuatro sub-hipótesis, cada una referida a una dimensión medible de la calidad, según ilustra la Figura 8. La siguiente tabla sintetiza esas cuatro sub--hipótesis, la dimensión que cada una operacionaliza, la dirección del efecto que predice respecto al modelo sin augmentación y el umbral declarado como expectativa a contrastar.

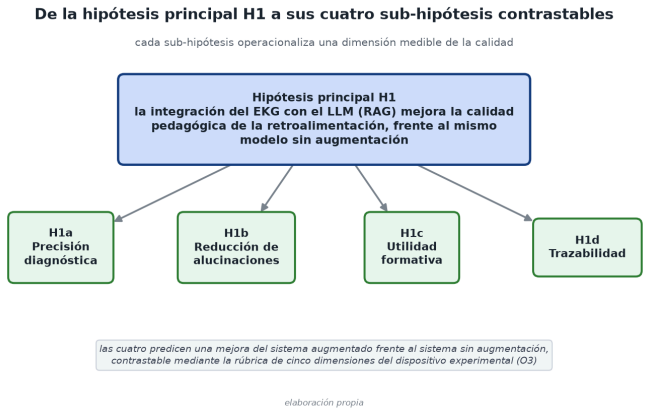

Figura 8. De la hipótesis principal H1 a sus cuatro sub-hipótesis contrastables. *Fuente: elaboración propia.*

Sub-hipótesis	Dimensión de calidad	Efecto predicho (sistema augmentado frente al sistema sin augmentación)	Umbral declarado como expectativa
H1a	Precisión diagnóstica	Identifica el error y el concepto implicado con mayor precisión	≈ 85 %
H1b	Reducción de alucinaciones	Produce menos afirmaciones infundadas o alucinaciones	≤ 5 %
H1c	Utilidad formativa	Ofrece una retroalimentación más útil desde el punto de vista formativo	≥ 4,0 sobre 5
H1d	Trazabilidad	Vincula un mayor número de afirmaciones a un elemento identificable del grafo	≈ 100 %

6.4.2 H1a. Precisión diagnóstica

La sub-hipótesis H1a predice que el sistema augmentado con el grafo **identifica el error y el concepto implicado con mayor precisión** que el modelo sin augmentación. El fundamento teórico es que el diagnóstico correcto de un error de programación no es solo una cuestión de reconocimiento sintáctico, sino de ubicar el fallo en el mapa conceptual del dominio. Distinguir, por ejemplo, si un bucle mal terminado obedece a una confusión sobre la condición de parada o sobre el alcance de una variable exige conocimiento estructurado sobre cómo se relacionan esos conceptos. La detección automática de errores y la clasificación de fallos en programación tienen una tradición investigadora consolidada (Keuning et al., 2019; Watson & Li, 2014; McCracken et al., 2001) que motiva esperar que un sustrato conceptual explícito mejore la precisión del diagnóstico. El umbral declarado como expectativa es una precisión del orden del 85 %; se enuncia como objetivo a contrastar, y la comparación entre condiciones es la que determinará en qué medida la augmentación produce el efecto predicho y si el umbral se aproxima.

6.4.3 H1b. Reducción de alucinaciones

La sub-hipótesis H1b predice que el sistema augmentado **produce menos afirmaciones infundadas o alucinaciones** que el modelo sin augmentación. El fundamento teórico es directo, porque si la generación se condiciona a un subgrafo de evidencia verificable, el modelo dispone de un anclaje que acota el espacio de afirmaciones legítimas y reduce el margen para inventar detalles plausibles pero falsos, que es el riesgo que la RAG busca contener (Lewis et al., 2020) y que la crítica a los modelos de lenguaje ha señalado como una de sus debilidades estructurales (Bender et al., 2021). En un contexto educativo, una alucinación no es un fallo benigno, ya que una afirmación diagnóstica incorrecta puede inducir al estudiante a un aprendizaje erróneo, lo que confiere a esta dimensión una importancia pedagógica de primer orden. El umbral declarado como expectativa es una tasa de alucinaciones igual o inferior al 5 %; de nuevo, se trata de una meta evaluable y no de un resultado asumido.

6.4.4 H1c. Utilidad formativa

La sub-hipótesis H1c predice que la retroalimentación del sistema augmentado **resulta más útil desde el punto de vista formativo** que la del modelo sin augmentación. El fundamento teórico se asienta en la teoría de la retroalimentación eficaz, según la cual una corrección útil no se limita a señalar que algo está mal, sino que ayuda al aprendiz a comprender por qué y a saber cómo avanzar, articulando la retroalimentación de proceso y de autorregulación que la investigación identifica como la más potente (Hattie & Timperley, 2007). El conocimiento estructurado del grafo (que incorpora prerrequisitos y conceptos vecinos) habilita una retroalimentación que sitúa el error en su contexto de aprendizaje y que puede sugerir el paso siguiente, en consonancia con el papel del conocimiento del dominio en la analítica

del aprendizaje y en el diseño de pistas (Gašević et al., 2022; Marwan et al., 2020). El umbral declarado como expectativa es una utilidad formativa media igual o superior a 4,0 sobre 5. Aquí debe reiterarse con especial énfasis la limitación de alcance ya señalada, porque la valoración de la utilidad formativa que este trabajo aporta procede de una evaluación automática con juez LLM sobre datos sintéticos, no del panel de docentes humanos previsto; por tanto, el contraste de H1c que se ofrece es preliminar y la validación con evaluadores humanos queda como trabajo futuro indispensable.

6.4.5 H1d. Trazabilidad

La sub-hipótesis H1d predice que el sistema augmentado **ofrece una trazabilidad superior** de sus afirmaciones, en el sentido de que un mayor número de las afirmaciones diagnósticas puede vincularse a un elemento identificable del grafo de conocimiento. El fundamento teórico es que la procedencia explícita es una propiedad que el conocimiento simbólico estructurado proporciona de forma natural y que la generación neuronal libre no garantiza, pues cuando la retroalimentación se construye sobre un subgrafo recuperado, cada afirmación puede en principio remitirse a los nodos y aristas que la sustentan, lo que hace el sistema auditable (Hogan et al., 2021; Abu-Salih & Alotaibi, 2024). Esta dimensión está íntimamente ligada al objetivo O4 de explicabilidad y a la confianza que docentes y estudiantes pueden depositar en la herramienta. El umbral declarado como expectativa es una trazabilidad próxima al 100 %; se enuncia como ideal de diseño y como predicción a verificar, entendiendo que la medición efectiva determinará en qué grado el sistema augmentado se acerca a él y, sobre todo, cuánto mejora respecto a la condición sin augmentación.

6.5 Síntesis y carácter contrastable de las hipótesis

En conjunto, los cinco objetivos específicos y las cuatro sub-hipótesis configuran un programa de investigación coherente que recorre las tres fases metodológicas del trabajo, donde O1 corresponde a la construcción del grafo, O2 al diseño de la arquitectura, y O3, O4 y O5 a la validación, la explicabilidad y el análisis de compromisos; la Figura 9 sintetiza esa correspondencia entre las tres fases y los cinco objetivos. Las sub-hipótesis H1a a H1d operacionalizan la hipótesis principal en cuatro dimensiones medibles (precisión, alucinaciones, utilidad y trazabilidad) que se corresponden directamente con las dimensiones de la rúbrica de evaluación, de manera que el dispositivo experimental no es un añadido sino el instrumento diseñado para someter a prueba cada predicción.

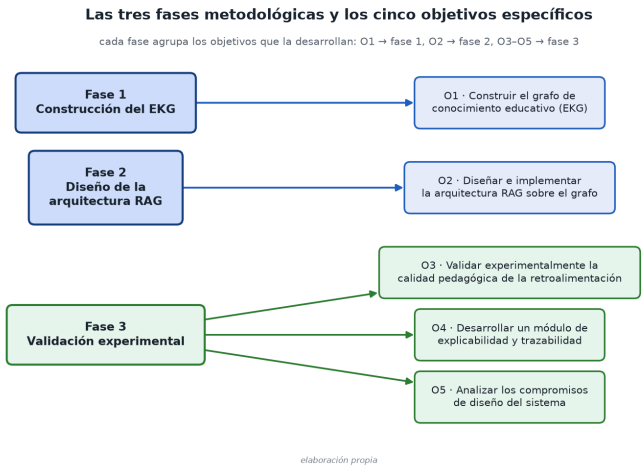

Figura 9. Las tres fases metodológicas y los cinco objetivos específicos. *Fuente: elaboración propia.*

Quiero cerrar el capítulo reafirmando el compromiso de integridad que lo abre. Todos los umbrales enunciados —la precisión del orden del 85 %, las alucinaciones por debajo del 5 %, la utilidad de al menos 4,0 sobre 5 y la trazabilidad cercana al 100 %— son expectativas a evaluar, no logros declarados. Las hipótesis H1a a H1d son genuinamente contrastables, pues predicen una dirección de efecto y pueden ser confirmadas, matizadas o refutadas por los datos. La evaluación que este trabajo ha podido ejecutar es automática y se apoya en datos sintéticos y en un juez constituido por un modelo de lenguaje, con

un número reducido de casos de prueba; la validación con un panel de docentes humanos, el análisis de concordancia entre jueces y el uso de entregas reales de estudiantes a gran escala permanecen como trabajo futuro que el propio diseño reclama. Esta delimitación del alcance no debilita la propuesta, sino que la sitúa en su justo lugar —una arquitectura diseñada, implementada y sometida a una primera validación rigurosa—, cuyos resultados (reportados sin inflar) se exponen en los capítulos siguientes.

7 Marco teórico: Web Semántica y estándares

La arquitectura que se propone en este Trabajo de Fin de Máster descansa sobre un grafo de conocimiento educativo (EKG) que, lejos de ser una base de datos convencional, se construye empleando el conjunto de estándares que conforman la denominada Web Semántica. Comprender estos estándares no es un mero requisito formal. La elección de RDF como modelo de datos, de OWL 2 RL como perfil de razonamiento y de SHACL como mecanismo de validación condiciona de manera directa qué inferencias pueden derivarse del grafo, qué garantías de consistencia ofrece y cómo se conecta con recursos externos como Wikidata. Dedico este capítulo a presentar de forma razonada los fundamentos de la Web Semántica y los estándares del World Wide Web Consortium (W3C) que sustentan la fase de construcción del EKG, tratando de ir más allá de la descripción técnica para explicitar por qué cada pieza resulta pertinente al problema de la evaluación formativa en programación.

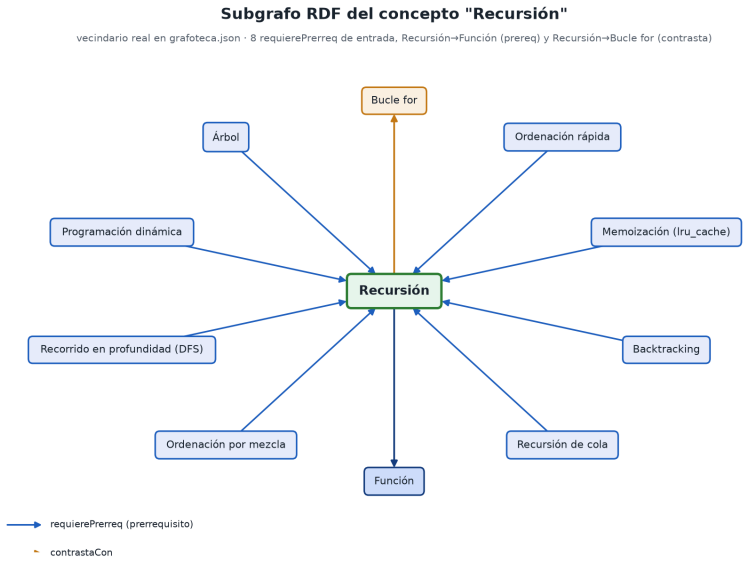

Figura 10. Subgrafo RDF del concepto «Recursión»: prerrequisitos, contrastes, error y tema. *Fuente: elaboración propia a partir del grafo canónico `ekg-python-150.ttl`.*

7.1 De la Web de documentos a la Web de datos

La World Wide Web nació como un espacio de documentos enlazados pensados para el consumo humano. Las páginas se conectan entre sí mediante hiperenlaces, pero el significado de su contenido permanece, en la práctica, opaco para las máquinas. Un programa puede recuperar una página y mostrarla, pero no puede razonar de manera fiable sobre las entidades y relaciones que esa página describe. La visión de la Web Semántica, articulada a lo largo de las dos décadas siguientes por el W3C, propone complementar esa Web de documentos con una Web de datos en la que la información se publique de forma estructurada, con una semántica explícita y procesable por agentes automáticos. El objetivo no es sustituir el contenido textual, sino dotarlo de una capa de significado que permita integrar, consultar e inferir conocimiento a través de fuentes heterogéneas.

Este giro tiene implicaciones profundas para un dominio como la educación en programación. Conceptos como bucle, recursión, complejidad algorítmica o variable no son etiquetas aisladas, sino nodos de una red de dependencias conceptuales, prerrequisitos pedagógicos y relaciones de especialización. Representar ese conocimiento en un formato que las máquinas puedan recorrer y sobre el que puedan razonar abre la posibilidad de que un sistema de evaluación automática no se limite a clasificar superficialmente un error, sino que lo sitúe dentro de la estructura conceptual de la materia, identifique los prerrequisitos implicados y trace la procedencia de cada afirmación que produce. La Web Semántica aporta el sustrato formal para ello, y en este trabajo se ha optado por adoptar sus estándares en lugar de construir un modelo de grafo ad hoc, con el fin de aprovechar la interoperabilidad, la reutilización de vocabularios consolidados y la capacidad de razonamiento que ofrecen.

7.2 RDF 1.1: el modelo de datos por tripletas

El Resource Description Framework (RDF) constituye el modelo de datos fundamental sobre el que se apoya toda la pila tecnológica de la Web Semántica. En su versión 1.1, RDF representa la información mediante afirmaciones elementales denominadas tripletas, compuestas por un sujeto, un predicado y un objeto (W3C, 2014a). Cada tripleta expresa un hecho atómico (por ejemplo, que un determinado concepto tiene como prerrequisito otro concepto) y el conjunto de todas las tripletas conforma un grafo dirigido y etiquetado en el que los nodos son recursos o valores y las aristas son las propiedades que los relacionan. Esta simplicidad estructural es, paradójicamente, la fuente de su potencia, ya que cualquier dominio de conocimiento puede descomponerse en afirmaciones de este tipo, y los grafos resultantes pueden fusionarse de manera natural simplemente uniendo sus conjuntos de tripletas. La Figura 10 ilustra este modelo sobre un fragmento real del EKG, el subgrafo del concepto «Recursión», donde cada arista es un predicado que enlaza el concepto con sus prerrequisitos, sus contrastes y el error típico asociado.

Los recursos en RDF se identifican mediante IRI (Internationalized Resource Identifiers), lo que garantiza que una entidad pueda ser referida de forma unívoca y global.[7] En el EKG desarrollado en este trabajo he definido dos espacios de nombres propios para organizar estas identidades. El primero es el prefijo `pyedu:`, reservado para los elementos del esquema (las clases y propiedades que definen la ontología del dominio), y el prefijo `pyr:`, destinado a las instancias concretas (los conceptos, errores y actividades particulares). Esta separación entre el nivel terminológico y el nivel asercional, habitual en la práctica de la ingeniería ontológica, facilita el mantenimiento del grafo y clarifica la distinción entre la definición de los tipos y los datos individuales.

RDF 1.1 distingue tres clases de términos que pueden ocupar las posiciones de una tripleta. Los IRI nombran recursos; los literales representan valores de datos como cadenas, números o fechas, y pueden ir acompañados de un tipo de dato o de una etiqueta de idioma; y los nodos en blanco (blank nodes) representan recursos cuya identidad no se hace explícita mediante un IRI y resultan especialmente útiles para modelar estructuras auxiliares. Esta última figura adquiere relevancia cuando se necesita representar relaciones que no encajan de forma natural en el patrón binario de la tripleta. En el EKG construido, por ejemplo, ha sido necesario modelar relaciones N-arias y reificar ciertas afirmaciones para anotarlas con información de procedencia. La relación de evaluación de una actividad vincula simultáneamente a un estudiante, una actividad, un concepto evaluado y un resultado, y no puede expresarse con una única tripleta binaria. Se ha materializado como una clase intermedia, `EvaluacionActividad`, de la que se han creado diez instancias en el grafo canónico. Del mismo modo, la reificación permite tratar una afirmación como sujeto de otras afirmaciones, de modo que pueda decirse de quién procede una determinada relación o en qué fuente se sustenta, capacidad esencial para el módulo de explicabilidad y trazabilidad que persigue el objetivo O4 del proyecto.

RDF, por sí solo, no impone restricciones sobre qué tripletas son admisibles ni atribuye semántica especial a los predicados más allá de la que el propio grafo declara. Es un modelo deliberadamente minimalista, una base sobre la que las capas superiores (RDFS, OWL, SHACL) añaden expresividad semántica y capacidad de validación. Esta arquitectura en capas es una de las decisiones de diseño más acertadas de la Web Semántica, ya que permite adoptar progresivamente la complejidad que cada aplicación necesita sin verse obligada a asumir de golpe todo el aparato formal.

7.3 Serialización en Turtle

Un grafo RDF es una estructura abstracta, independiente de cualquier representación textual concreta. Para almacenarlo, transmitirlo o editarlo se requiere una sintaxis de serialización, y RDF admite varias, como RDF/XML, JSON-LD, N-Triples o Turtle, entre otras. En este trabajo se ha optado por Turtle (Terse RDF Triple Language) como formato principal de serialización del EKG, decisión que merece justificarse.

[7] Un IRI generaliza al URI clásico al admitir caracteres Unicode más allá del repertorio ASCII; en la práctica del EKG empleo IRI cortos abreviados con prefijos como `pyedu:` y `pyr:`.

Turtle ofrece un equilibrio notable entre legibilidad humana y concisión, pues permite declarar prefijos para abreviar los IRI, agrupar varias tripletas que comparten sujeto mediante el separador punto y coma, y enumerar objetos de un mismo predicado con la coma, lo que reduce drásticamente la verbosidad frente a alternativas como RDF/XML. Para un grafo que ha de ser inspeccionado, revisado y corregido manualmente durante su construcción, esta legibilidad resulta determinante.

La elección de Turtle se alinea además con la práctica habitual en la comunidad de la Web Semántica para la edición de ontologías y conjuntos de datos de tamaño moderado. El EKG canónico, con sus 157 conceptos propios y un total de 1772 enunciados afirmados antes de cualquier inferencia, se mantiene en un volumen perfectamente manejable en este formato, lo que facilita tanto el control de versiones del esquema como la auditoría de las afirmaciones por parte del autor y del director del trabajo.

7.4 RDFS: vocabulario y jerarquías básicas

Sobre el modelo de datos de RDF, el RDF Schema (RDFS) introduce un vocabulario que permite describir clases de recursos y propiedades, así como las relaciones jerárquicas entre ellos (W3C, 2014b). RDFS aporta construcciones como `rdfs:Class` para declarar clases, `rdf:type` para afirmar que un recurso es instancia de una clase, `rdfs:subClassOf` para establecer jerarquías de especialización entre clases, `rdfs:subPropertyOf` para análogas jerarquías entre propiedades, y `rdfs:domain` y `rdfs:range` para indicar el tipo de los sujetos y objetos que una propiedad relaciona. Con estos elementos, RDFS habilita un primer nivel de razonamiento, pues si se afirma que un concepto es de tipo `ConceptoAvanzado` y que `ConceptoAvanzado` es subclase de `Concepto`, un razonador puede inferir que dicho concepto es también de tipo `Concepto`, aunque esa tripleta no se haya afirmado explícitamente.

Esta capacidad inferencial básica es ya valiosa para el EKG, pues permite mantener un esquema económico y derivar automáticamente las consecuencias de las jerarquías declaradas. No obstante, RDFS tiene una expresividad limitada, porque no permite, por ejemplo, declarar que dos propiedades son inversas la una de la otra, que una propiedad es transitiva, o que dos individuos identificados con IRI distintos representan en realidad la misma entidad. Para estas necesidades, frecuentes al integrar conocimiento educativo con fuentes externas y al modelar relaciones de prerrequisito o equivalencia, es necesario recurrir a OWL.

7.5 OWL 2: ontologías y perfiles

El Web Ontology Language en su versión 2 (OWL 2) extiende sustancialmente la expresividad de RDFS, proporcionando un lenguaje rico para la construcción de ontologías formales con una semántica fundamentada en la lógica descriptiva (W3C, 2012). OWL 2 permite definir restricciones sobre las propiedades, declarar características como la transitividad, la simetría o la funcionalidad de una propiedad, expresar equivalencias y disyunciones entre clases, e introducir axiomas sobre la cardinalidad. Una construcción de particular importancia para este trabajo es `owl:sameAs`, que afirma que dos IRI denotan el mismo individuo, lo que permite conectar entidades del EKG con sus equivalentes en grafos de conocimiento externos.

La gran expresividad de OWL 2 tiene un coste, ya que el razonamiento sobre la lógica descriptiva completa que subyace al lenguaje (conocida como OWL 2 DL) es computacionalmente costoso, y en su forma más general indecidible. Consciente de este compromiso entre expresividad y eficiencia, el W3C definió en OWL 2 tres perfiles, subconjuntos sintácticos del lenguaje diseñados para garantizar buenas propiedades computacionales en escenarios de uso característicos (W3C, 2012). El perfil OWL 2 EL está orientado a ontologías con un gran número de clases y propiedades y garantiza razonamiento en tiempo polinómico. Resulta idóneo para grandes terminologías como las del ámbito biomédico. El perfil OWL 2 QL está concebido para la respuesta eficiente a consultas sobre grandes volúmenes de instancias, lo que permite reescribir el razonamiento como consultas sobre bases de datos relacionales. Y el perfil OWL 2 RL, que es el adoptado en este trabajo, está diseñado para implementaciones escalables basadas en reglas, de modo que el razonamiento puede realizarse mediante un motor de reglas que aplica de

forma iterativa un conjunto fijo de implicaciones hasta alcanzar un punto fijo. La Figura 11 sitúa estos tres perfiles según el compromiso entre expresividad y garantías de cómputo, y señala por qué el EKG adopta RL.

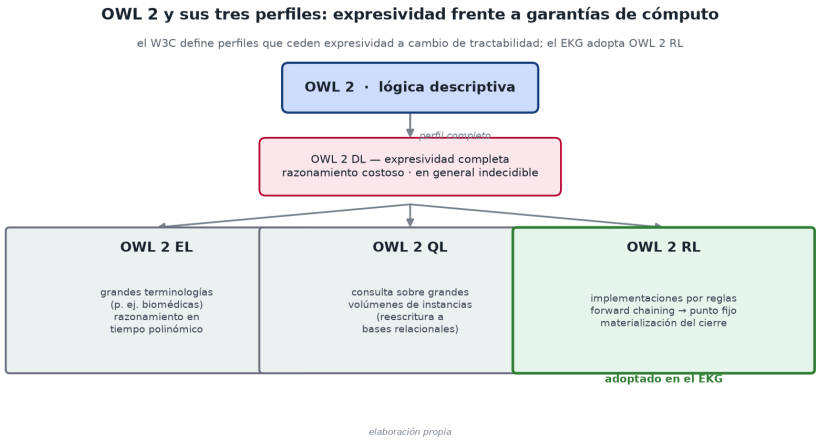

Figura 11. Los tres perfiles de OWL 2 (EL, QL, RL) y por qué el EKG adopta el perfil RL. *Fuente: elaboración propia.*

7.5.1 OWL 2 RL como perfil de razonamiento del EKG

La decisión de emplear el perfil OWL 2 RL para el EKG responde a un análisis explícito de las necesidades del proyecto frente a las garantías que cada perfil ofrece. OWL 2 RL permite materializar las inferencias mediante un razonamiento hacia adelante (forward chaining) que aplica reglas de manera determinista y completa para el fragmento del lenguaje que cubre, con un coste computacional tratable y predecible. Esto encaja con un escenario en el que el grafo se construye una vez, se materializa el cierre inferencial y, a partir de ese momento, las consultas SPARQL operan sobre el grafo ya enriquecido sin necesidad de razonar en tiempo de consulta. Para una aplicación que ha de recuperar subgrafos con baja latencia dentro de una arquitectura RAG, esta materialización previa resulta especialmente conveniente.

El efecto de la inferencia bajo OWL 2 RL sobre el EKG es cuantitativamente notable. El grafo canónico parte de 1772 enunciados afirmados de manera explícita; tras aplicar el razonamiento OWL-RL, el número de enunciados asciende a 4786, lo que refleja la materialización de todas las consecuencias lógicas derivadas de las jerarquías de clases y propiedades y de las características declaradas sobre las propiedades. Más ilustrativo todavía resulta el comportamiento de una consulta concreta sobre los conceptos del dominio. Sin inferencia, una determinada consulta que recupera los conceptos del dominio a través de las jerarquías declaradas devuelve cero resultados, porque las tripletas necesarias para satisfacerla no están afirmadas explícitamente; con la inferencia activada, la misma consulta devuelve 157 conceptos, que corresponden a los 157 conceptos propios del EKG. Las 30 entidades de Wikidata con las que se enlaza el grafo no engrosan esta cifra, ya que, al vincularse mediante `skos:exactMatch` (y no mediante `owl:sameAs`), el razonador no propaga el tipo `pyedu:Concepto` a dichas entidades, de modo que estas no se infieren como conceptos del dominio. Estos 30 enlaces `skos:exactMatch` hacia Wikidata, junto con las 19 relaciones `skos:broader` que organizan la jerarquía temática, constituyen el principal mecanismo de enlazado del grafo con conocimiento externo, y es el razonamiento bajo OWL 2 RL el que articula su integración a efectos de consulta.

El esquema del EKG comprende 20 clases, 21 propiedades de objeto y 7 propiedades de datos, una dimensión moderada pero suficiente para articular el dominio de la enseñanza de la programación en Python con un nivel de detalle adecuado a los objetivos pedagógicos. Esta contención deliberada en el tamaño del esquema, lejos de ser una limitación, facilita que el razonamiento OWL 2 RL se mantenga eficiente y que el grafo resultante sea auditable.

7.6 SPARQL 1.1: consulta y manipulación del grafo

Disponer de un grafo RDF, por rico que sea, carece de utilidad práctica sin un lenguaje que permita interrogarlo. Ese lenguaje es SPARQL, cuya versión 1.1 es el estándar del W3C para la consulta y manipulación de grafos RDF (W3C, 2013). SPARQL se basa en el emparejamiento de patrones de grafo, donde una consulta describe un patrón de tripletas con variables, y el motor de consultas busca todas las formas en que ese patrón puede instanciarse contra el grafo, devolviendo las asignaciones de variables que lo satisfacen. SPARQL 1.1 ofrece cuatro formas de consulta —`SELECT` para obtener tablas de resultados, `CONSTRUCT` para generar nuevos grafos RDF, `ASK` para comprobaciones booleanas y `DESCRIBE` para obtener una descripción de un recurso— así como un conjunto de operadores que incluye filtros, uniones opcionales, agregaciones, subconsultas y rutas de propiedad (property paths), estas últimas particularmente útiles para recorrer cadenas de relaciones de longitud variable, como las jerarquías de prerrequisitos.

En la arquitectura propuesta, SPARQL ocupa un lugar central en el motor de recuperación de subgrafos, que constituye uno de los componentes del objetivo O2. Tras una primera fase de recuperación basada en la similitud semántica entre el código del estudiante y los conceptos del grafo mediante embeddings, el sistema expande el conjunto inicial de nodos recuperados mediante consultas SPARQL que recorren las relaciones del EKG, incorporando prerrequisitos, conceptos relacionados y errores asociados. Esta expansión guiada por la estructura del grafo es lo que distingue a un enfoque GraphRAG de una recuperación puramente vectorial. La consulta SPARQL aporta el conocimiento estructural que el modelo de embeddings, por sí solo, no captura. SPARQL 1.1 incorpora además capacidades de actualización (SPARQL Update) que permiten insertar y eliminar tripletas, empleadas durante la construcción y el mantenimiento del grafo.

7.7 SHACL: validación de la conformidad del grafo

Una característica que distingue a RDF de los modelos de datos relacionales es la denominada hipótesis del mundo abierto, donde la ausencia de una afirmación no se interpreta como su negación, sino como desconocimiento. RDF, RDFS y OWL no imponen, por diseño, un esquema cerrado que restrinja qué datos son válidos; OWL, en particular, está orientado a la inferencia, no a la validación, y sus axiomas se interpretan como restricciones lógicas a partir de las cuales se derivan consecuencias, no como reglas de integridad que un dato pueda violar. Esta semántica, adecuada para integrar conocimiento heterogéneo, resulta sin embargo insuficiente cuando se necesita garantizar que un grafo cumple determinadas condiciones de calidad estructural antes de utilizarlo en producción.

Para cubrir esta necesidad, el W3C estandarizó SHACL (Shapes Constraint Language), un lenguaje para describir y validar restricciones sobre grafos RDF (W3C, 2017). SHACL permite definir formas (shapes) que especifican condiciones que deben cumplir los nodos de un grafo, como los tipos de datos esperados, las cardinalidades mínimas y máximas, los patrones que deben satisfacer los valores o las clases a las que deben pertenecer los objetos de una propiedad, entre muchas otras. A diferencia de OWL, SHACL opera bajo una hipótesis de mundo cerrado a efectos de validación, de modo que comprueba que el grafo se ajusta a las formas declaradas y, cuando no lo hace, produce un informe de validación que enumera las violaciones encontradas, indicando el nodo, la propiedad y la restricción incumplida. Esta capacidad de diagnóstico explícito es complementaria del razonamiento OWL, pues mientras OWL deriva nuevo conocimiento, SHACL verifica la integridad del existente.

En el desarrollo del EKG, SHACL ha cumplido la función de garantía de calidad sobre el grafo canónico. Se han definido formas que codifican las restricciones estructurales esperadas del dominio, y la validación del grafo canónico arroja un resultado CONFORME, sin violaciones, lo que confirma que el grafo respeta las condiciones de integridad declaradas. A fin de verificar que la validación es efectiva y no meramente trivial, se ha construido además un ejemplo deliberadamente inválido, sobre el que SHACL detecta correctamente seis violaciones. Esta prueba complementaria tiene un peso metodológico claro, porque demuestra que las formas definidas tienen poder discriminativo real y que el resultado conforme

del grafo canónico es significativo y no consecuencia de unas restricciones vacías. La combinación de razonamiento OWL 2 RL para enriquecer el grafo y validación SHACL para certificar su integridad proporciona, en conjunto, una base de conocimiento fiable sobre la que apoyar la generación de feedback. Esta división de papeles entre OWL, que deriva conocimiento nuevo bajo la hipótesis de mundo abierto, y SHACL, que comprueba la integridad del existente bajo la de mundo cerrado, se resume en la Figura 12.

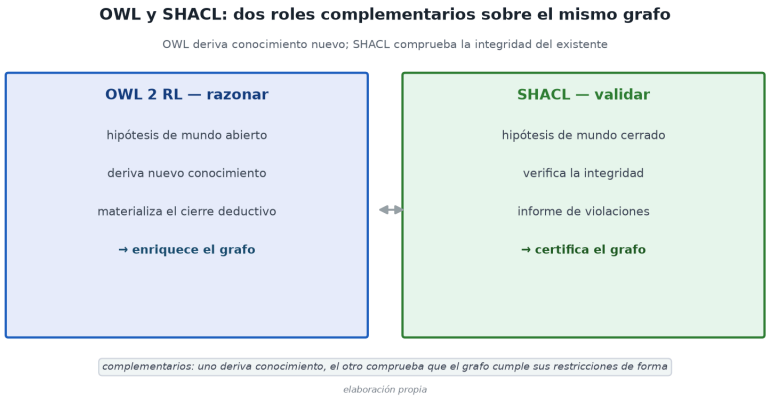

Figura 12. Roles complementarios de OWL (razonar, mundo abierto) y SHACL (validar, mundo cerrado).
Fuente: elaboración propia.

7.8 Linked Data y SKOS: enlazado con conocimiento externo

Los estándares anteriores definen cómo representar, consultar, razonar y validar grafos RDF, pero la verdadera promesa de la Web Semántica se materializa cuando esos grafos se publican y conectan entre sí siguiendo los principios de los Linked Data (datos enlazados). Estos principios, formulados originalmente en el seno del W3C, establecen que las entidades deben identificarse mediante IRI desreferenciables, que al acceder a una IRI debe obtenerse información útil expresada en estándares de la Web Semántica, y que esa información debe incluir enlaces a otras entidades para que sea posible descubrir más conocimiento navegando por el grafo global. El resultado es una nube de datos enlazados en la que conjuntos de datos publicados de forma independiente se entretejen en un espacio de información común.

Para el EKG, los principios de Linked Data se concretan en su conexión con Wikidata, uno de los grafos de conocimiento abiertos de mayor cobertura. A través de las 30 relaciones `skos:exactMatch` mencionadas anteriormente, conceptos del dominio definidos localmente se vinculan con sus contrapartes en Wikidata, lo que permite, gracias al razonamiento OWL 2 RL, incorporar conocimiento externo al espacio de consulta del sistema y enriquecer el contexto disponible para la generación de feedback. Se ha optado deliberadamente por `skos:exactMatch` en lugar de `owl:sameAs` para no imponer la fusión lógica de individuos propia del perfil OWL 2 RL[8]. Esta elección preserva la autonomía del concepto del dominio frente a la entidad externa y refleja una madurez en el uso de los datos enlazados. Este enlazado amplía la cobertura conceptual del grafo y, además, lo inscribe en la red más amplia de los datos enlazados, lo que abre la puerta a futuras integraciones con otros recursos.

En la dimensión propiamente terminológica, el EKG hace uso de SKOS (Simple Knowledge Organization System), el estándar del W3C para representar sistemas de organización del conocimiento como tesauros, taxonomías y esquemas de clasificación (W3C, 2009). SKOS proporciona un vocabulario ligero para expresar relaciones semánticas entre conceptos, entre las que destaca `skos:broader`, que indica que un concepto es más general que otro. En el EKG se han establecido 19 relaciones `skos:broader` que articulan una jerarquía temática de los conceptos del dominio, complementando las jerarquías de

[8] dicha fusión identificaría el concepto local con la entidad de Wikidata como un mismo individuo y propagaría tipos y propiedades entre ambos

clases definidas con RDFS y OWL. El recurso a SKOS para la organización temática, en lugar de modelar toda la estructura mediante subclases OWL, responde a una distinción conceptual. `skos:broader` expresa una relación de generalidad temática entre conceptos tratados como individuos, mientras que `rdfs:subClassOf` expresa una relación de subsunción entre clases. Mantener separadas ambas perspectivas evita confusiones semánticas habituales y se ajusta a las recomendaciones de buenas prácticas en la modelización de conocimiento.

7.9 Razonadores y materialización de inferencias

El razonamiento es el componente que convierte un grafo RDF de un mero repositorio de afirmaciones en una base de conocimiento capaz de derivar consecuencias. Un razonador es un programa que, dado un grafo y la semántica de los estándares aplicables (RDFS, un perfil de OWL), computa las tripletas que se siguen lógicamente de las afirmadas. Existen dos estrategias principales para incorporar el razonamiento en una aplicación. La primera es el razonamiento en tiempo de consulta, en el que las inferencias se calculan bajo demanda al ejecutar cada consulta; ofrece flexibilidad pero penaliza la latencia. La segunda es la materialización, en la que el cierre inferencial se computa una sola vez y las tripletas derivadas se almacenan junto a las afirmadas, de modo que las consultas posteriores operan sobre un grafo ya enriquecido sin coste adicional de razonamiento.

En este trabajo se ha optado por la materialización bajo el perfil OWL 2 RL, estrategia coherente con las características del problema. El EKG es un grafo de construcción mayoritariamente estática, pues se elabora, se valida y se materializa, y a partir de ese momento se consulta de forma repetida desde el motor de recuperación. La materialización previa garantiza que las consultas SPARQL que sustentan la recuperación de subgrafos accedan directamente a todo el conocimiento, tanto el afirmado como el inferido, sin introducir latencia de razonamiento en el camino crítico de la inferencia formativa. El propio perfil OWL 2 RL, diseñado para implementaciones basadas en reglas mediante razonamiento hacia adelante, se presta de manera natural a esta materialización (W3C, 2012). Como se ha cuantificado anteriormente, el efecto de este proceso es el paso de 1772 a 4786 enunciados y la habilitación de consultas que, sin inferencia, no devolverían resultado alguno.

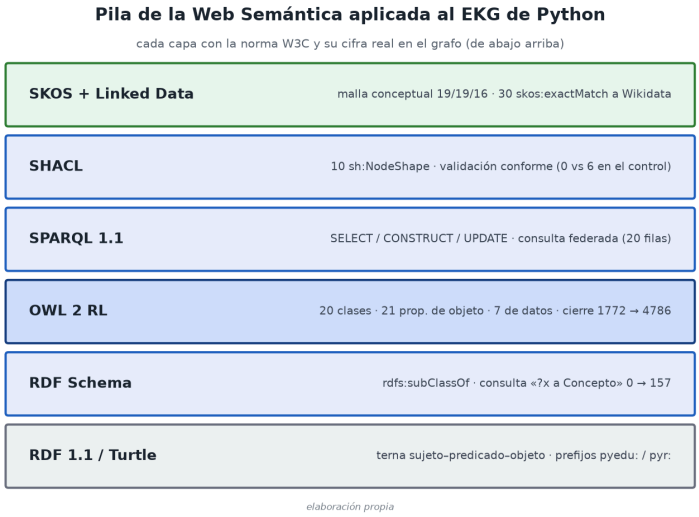

Figura 13. Pila de la Web Semántica aplicada al EKG (cifras reales por capa). *Fuente: elaboración propia.*

Desde el punto de vista de la implementación, el ecosistema de la Web Semántica ofrece diversas herramientas de razonamiento y almacenamiento de grafos. En la realización de este trabajo se han empleado la biblioteca rdflib del ecosistema Python (RDFLib Team, s.f.) para la manipulación programática de los grafos y la aplicación del razonamiento OWL-RL, así como el almacén de tripletas GraphDB de Ontotext como repositorio con capacidades de consulta SPARQL y razonamiento (Ontotext, s.f.). La elección concreta de estas herramientas se apartó de lo previsto en el anteproyecto inicial; las implicaciones de

esta y otras decisiones de implementación se discuten en el capítulo correspondiente a la arquitectura del sistema. Lo relevante en este punto es que el conjunto de estándares descrito —RDF 1.1 como modelo de datos, serializado en Turtle; RDFS y OWL 2 RL para la estructura y el razonamiento; SPARQL 1.1 para la consulta; SHACL para la validación; y SKOS junto a los principios de Linked Data para el enlazado externo— conforma un marco coherente y maduro, sintetizado por capas en la Figura 13, que proporciona al EKG una semántica formal, una capacidad inferencial controlada y unas garantías de integridad verificables, sobre las que se edifica el resto de la arquitectura propuesta.

8 Marco teórico: grafos de conocimiento

En este capítulo establezco los fundamentos conceptuales sobre los que se asienta el primer pilar de la arquitectura que propongo en este Trabajo de Fin de Máster, que es el grafo de conocimiento educativo (EKG) que actúa como sustrato semántico para la recuperación aumentada. Antes de describir cómo construí dicho grafo en el capítulo dedicado a la implementación, quiero fijar con precisión qué entiendo por grafo de conocimiento, qué modelos de datos compiten para representarlo, cómo se construye y se mantiene, por qué resulta razonable apoyarse en vocabularios ya existentes en lugar de reinventarlos, y qué papel desempeña el grafo como base operativa de un sistema que pretende generar retroalimentación formativa fundamentada y explicable. A lo largo del capítulo argumento, además, la decisión de modelado más relevante de esta fase, que es la adopción del estándar RDF/OWL del W3C frente a la alternativa de las bases de datos de grafos de propiedades como Neo4j. Esta elección no es meramente tecnológica. Condiciona de raíz las capacidades de inferencia, validación y enlazado que la propia hipótesis del trabajo necesita para ser puesta a prueba.

8.1 Definición y caracterización

La noción de grafo de conocimiento se ha popularizado desde que las grandes compañías tecnológicas la adoptaron como denominación para sus infraestructuras de datos interconectados. Aun así, ha sufrido durante años de una cierta imprecisión terminológica. Más allá de los tratados de referencia que han sistematizado sus fundamentos, técnicas y aplicaciones (Kejriwal et al., 2021), el trabajo de síntesis más influyente y citado de los últimos años, el extenso survey de Hogan et al. (2021), propone una definición que adopto en esta memoria por su equilibrio entre generalidad y operatividad. Según esa definición, un grafo de conocimiento es «un grafo de datos cuyo objetivo es acumular e integrar conocimiento sobre el mundo, cuyos nodos representan entidades de interés y cuyas aristas representan relaciones potencialmente diversas entre dichas entidades». Lo determinante de esta definición no es tanto la estructura de grafo (cualquier conjunto de pares relacionados puede dibujarse como tal) cuanto la combinación de tres rasgos que Hogan et al. (2021) desarrollan a lo largo de su trabajo: la representación de datos como grafo, la presencia de una capa de esquema o de ontología que dota de semántica a esos datos, y la voluntad explícita de integrar información procedente de fuentes heterogéneas en una representación común y consultable.

Quiero subrayar que la definición de Hogan et al. (2021) es deliberadamente agnóstica respecto del modelo de datos concreto y respecto de la tecnología de almacenamiento. Un grafo de conocimiento puede materializarse mediante RDF, mediante un grafo de propiedades o incluso mediante otras representaciones, y puede residir en un triplestore, en una base de datos de grafos nativa o en estructuras híbridas. Lo que caracteriza a un grafo de conocimiento, frente a un simple grafo de datos, es la presencia de una capa semántica explícita (vocabularios, taxonomías, ontologías, reglas) que permite almacenar afirmaciones y, además, razonar sobre ellas, validarlas y enlazarlas con conocimiento externo. Esta distinción es central para el presente trabajo, porque el valor del EKG no reside en que represente conceptos de programación como nodos y sus relaciones como aristas, algo que cualquier estructura de datos permitiría, sino en que esa representación está dotada de una semántica formal que habilita la inferencia de conocimiento implícito, la verificación de la integridad de los datos y la conexión con recursos de conocimiento abierto.

En el dominio educativo, esta caracterización adquiere matices particulares. Un grafo de conocimiento educativo modela el dominio de aprendizaje (en mi caso, los conceptos fundamentales de la programación en Python) pero también las relaciones pedagógicamente significativas entre esos conceptos, como los prerrequisitos, las jerarquías de generalización y especialización, los errores típicos asociados a cada concepto, o los vínculos entre actividades de evaluación y las competencias que pretenden medir. La literatura sobre grafos de conocimiento educativos, recogida por ejemplo en la revisión de Abu-Salih y Alotaibi (2024), insiste en que la utilidad de estas estructuras depende de su capacidad para capturar la estructura cognitiva del dominio de un modo que sea aprovechable por sistemas de tutoría, recomen-

dación o evaluación. No basta con enumerar conceptos; es preciso representar cómo se articulan entre sí de manera que un sistema automático pueda, por ejemplo, identificar qué concepto subyace a un error y qué prerrequisitos podrían estar mal asimilados.

8.2 RDF-graphs frente a property-graphs

Existen dos grandes paradigmas para materializar un grafo de conocimiento, y la elección entre ellos no es indiferente. El primero es el de los grafos RDF, definidos por el marco de descripción de recursos del W3C (W3C, 2014a). En RDF la unidad mínima de información es la tripleta, una afirmación de la forma sujeto–predicado–objeto, donde sujeto y predicado se identifican mediante IRIs (identificadores globales y desreferenciables) y el objeto puede ser otra IRI o un literal tipado. Un grafo RDF no es más que un conjunto de tales tripletas. Sobre este modelo básico se construye una pila de estándares complementarios. RDFS aporta una semántica ligera de clases y propiedades con sus jerarquías (W3C, 2014b); OWL 2 introduce una capacidad expresiva mucho mayor, con axiomas que permiten describir restricciones, cardinalidades, equivalencias y reglas de inferencia (W3C, 2012); SPARQL proporciona un lenguaje de consulta estandarizado para interrogar estos grafos (W3C, 2013); SHACL permite definir restricciones de forma y validar la conformidad de los datos (W3C, 2017); y SKOS ofrece un vocabulario ligero para representar sistemas de organización del conocimiento como tesauros y taxonomías (W3C, 2009).

El segundo paradigma es el de los grafos de propiedades, popularizado por bases de datos como Neo4j. En este modelo, los nodos y las aristas son entidades de primera clase que pueden llevar etiquetas y, sobre todo, pares clave-valor (las propiedades) adheridos directamente a ellos. La principal ventaja de este enfoque es su naturalidad para representar atributos de las relaciones. Si quiero anotar una arista «requiere» con un peso o una fecha, en un grafo de propiedades basta con añadir esa propiedad a la arista, mientras que en RDF la tripleta es atómica y anotar una relación exige recurrir a mecanismos adicionales como la reificación o las relaciones N-arias. Los grafos de propiedades suelen ofrecer también un rendimiento muy competitivo en el recorrido de caminos y en consultas de vecindad, y su modelo es a menudo percibido como más cercano a la intuición del programador.

Esa comodidad tiene una contrapartida decisiva. El modelo de grafos de propiedades carece de una semántica formal estandarizada. Las etiquetas y propiedades de un grafo Neo4j tienen el significado que el desarrollador les atribuye por convención, pero no existe una capa de ontología interoperable que permita, de forma estándar, inferir conocimiento nuevo, validar la conformidad de los datos contra un esquema declarativo o enlazar las entidades con identificadores globales reconocidos por terceros. Como discuten Hogan et al. (2021) al comparar ambos modelos, la frontera no es absoluta (existen pasarelas y extensiones) pero el ecosistema RDF está construido sobre una base lógica explícita, con perfiles de razonamiento bien definidos y una tradición de interoperabilidad, mientras que el ecosistema de grafos de propiedades prioriza el rendimiento operacional y la flexibilidad del esquema sobre la formalidad semántica.

La tabla siguiente sintetiza, criterio a criterio, el contraste entre ambos paradigmas, donde las primeras filas recogen las diferencias de modelado descritas en esta sección y las tres últimas anticipan las capacidades —inferencia, validación y enlazado— que justifican la elección y que detallo a continuación.

Criterio	RDF/OWL (W3C)	Grafo de propiedades (LPG)
Unidad de información	Tripleta sujeto–predicado–objeto	Nodos y aristas con etiquetas y pares clave-valor
Identificación	IRIs globales y desreferenciables	Etiquetas y propiedades por convención
Anotación de relaciones	Reificación o relaciones N-arias	Propiedad adherida directamente a la arista
Semántica formal	Estandarizada (RDFS, OWL 2, SPARQL, SHACL, SKOS)	Sin semántica formal estandarizada
Rendimiento en recorrido	Puede ser inferior	Muy competitivo en caminos y vecindad
Inferencia	Perfiles de razonamiento (OWL 2 RL)	Lógica de aplicación ad hoc
Validación	Declarativa y estándar (SHACL)	Comprobaciones embebidas en el código
Enlazado externo	IRIs globales (skos:exactMatch / owl:sameAs)	Convenciones ad hoc

8.3 Justificación de RDF/OWL: inferencia, validación y enlazado

La decisión de modelar el EKG de este trabajo en RDF/OWL, y no en un grafo de propiedades, responde directamente a tres requisitos que se derivan de los objetivos y de la hipótesis del proyecto: la necesidad de inferencia, la necesidad de validación y la necesidad de enlazado. La Figura 14 resume el contraste entre ambos paradigmas y sitúa estas tres capacidades del lado de RDF/OWL. Examino cada uno de ellos a la luz de los resultados concretos que obtuve en la construcción del grafo.

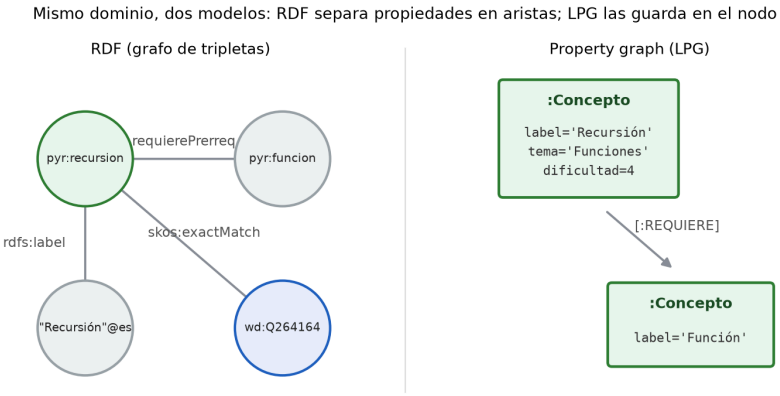

Figura 14. RDF/OWL frente a property graph (LPG).

El primer requisito es la inferencia. La hipótesis del trabajo sostiene que aumentar el modelo de lenguaje con conocimiento estructurado mejora la calidad pedagógica de la retroalimentación, y parte de ese valor reside en el conocimiento que no está afirmado explícitamente pero que puede deducirse. RDF/OWL, mediante perfiles de razonamiento como OWL 2 RL —el perfil adoptado en este trabajo por su buen compromiso entre expresividad y tratabilidad computacional—, permite materializar conocimiento implícito a partir de los axiomas declarados. El efecto es tangible en el grafo construido. Partiendo de 1772 enunciados afirmados sobre 157 conceptos propios, la aplicación del razonamiento OWL-RL eleva el grafo a 4786 enunciados y hace explícitas relaciones que estaban latentes en la combinación de jerarquías, propiedades transitivas y equivalencias. Un ejemplo especialmente ilustrativo es el de las consultas sobre el conjunto de conceptos. Sin inferencia, una determinada consulta de clasificación

devuelve cero resultados, porque la pertenencia de las entidades a la clase consultada no está afirmada literalmente; con inferencia activada, la misma consulta devuelve 157 conceptos (los 157 conceptos propios). Debo precisar que las 30 entidades de Wikidata enlazadas no se cuentan aquí, ya que, al estar vinculadas mediante `skos:exactMatch`, y no mediante `owl:sameAs`, el razonador no propaga hacia ellas el tipo `pyedu:Concepto`, y así el enlazado conecta sin fusionar y la consulta de clasificación recupera únicamente los conceptos propios. Esta capacidad de derivar conocimiento nuevo de forma estándar y verificable no tiene equivalente directo en un grafo de propiedades, donde cualquier «inferencia» equivalente debería programarse a mano como lógica de aplicación, sin garantías formales de corrección ni interoperabilidad.

El segundo requisito es la validación. Un sistema de evaluación educativa que pretenda ser fiable necesita garantías sobre la integridad de los datos que sustentan su razonamiento. SHACL (W3C, 2017) proporciona un mecanismo declarativo y estandarizado para expresar restricciones de forma (cardinalidades, tipos de datos, valores admisibles, patrones estructurales) y verificar automáticamente que el grafo las cumple. En la construcción del EKG, la validación SHACL del grafo canónico resultó CONFORME, sin violaciones, lo que ofrece una garantía objetiva de que la estructura respeta el esquema diseñado; y, para confirmar que la validación no es vacua, un ejemplo deliberadamente inválido es correctamente detectado con seis violaciones. Esta capacidad de validación declarativa, externa a la lógica de la aplicación y reutilizable, es uno de los argumentos de mayor peso a favor del ecosistema RDF. En un grafo de propiedades, la integridad estructural depende habitualmente de comprobaciones embebidas en el código de carga o en restricciones limitadas del propio motor, sin un lenguaje estándar y portable de restricciones equiparable a SHACL.

El tercer requisito es el enlazado. El principio del enlazado de datos consiste en conectar las entidades del grafo propio con identificadores globales de recursos externos, de modo que el conocimiento local se inserte en la red más amplia de datos abiertos. En este trabajo, 30 conceptos del EKG se vinculan mediante `skos:exactMatch` a entidades de Wikidata, y la organización jerárquica se apoya en 19 relaciones `skos:broader` que estructuran el dominio según un esquema de organización del conocimiento (W3C, 2009). La elección de `skos:exactMatch` sobre `owl:sameAs` es deliberada. Emplear `owl:sameAs` obligaría al razonador OWL-RL a fusionar lógicamente el concepto propio con la entidad de Wikidata, tratándolos como un mismo individuo, lo cual es semánticamente incorrecto; `skos:exactMatch`, en cambio, enlaza ambos recursos como equivalentes a efectos de recuperación sin imponer esa fusión de individuos, una decisión que refleja madurez en el uso de datos enlazados. El enlazado a Wikidata no es un adorno. Como ya he mostrado, gracias a esos vínculos de correspondencia y al razonamiento el conocimiento externo pasa a formar parte efectiva de las respuestas a las consultas y enriquece el grafo con conocimiento de una fuente curada y ampliamente reutilizada. Este tipo de interoperabilidad, basada en IRIs globales y en una semántica de identidad estandarizada, es nativa del modelo RDF y constituye el escenario para el que fue diseñado, mientras que en los grafos de propiedades el enlazado con recursos externos carece de un mecanismo estándar comparable y queda relegado a convenciones ad hoc.

En conjunto, inferencia, validación y enlazado configuran una tríada de capacidades que el modelo RDF/OWL ofrece de forma integrada y estandarizada, y que resultan funcionalmente necesarias para los objetivos del proyecto. Reconozco que esta elección tiene también costes. La curva de aprendizaje de la pila semántica es más pronunciada, la anotación de relaciones exige recurrir a la reificación o a las relaciones N-arias,[9] y el rendimiento de algunas operaciones de recorrido puede ser inferior al de un grafo de propiedades optimizado. La decisión no afirma que RDF sea superior en todo escenario, sino que es la opción adecuada cuando la inferencia formal, la validación declarativa y el enlazado con datos abiertos son requisitos de primer orden, como ocurre aquí.

[9]relaciones N-arias que en este grafo he empleado, por ejemplo, para modelar la evaluación de actividades como una relación N-aria reificada con su procedencia, con diez instancias de evaluación

8.4 Construcción del grafo de conocimiento

La construcción de un grafo de conocimiento es un proceso que Hogan et al. (2021) describen como inherentemente iterativo y que combina decisiones de modelado del esquema con la población de instancias. En este trabajo seguí una separación clara entre dos espacios de nombres, que son un espacio de esquema, `pyedu:`, que define la ontología (las clases y propiedades que estructuran el dominio), y un espacio de instancias, `pyr:`, que contiene los conceptos, errores, actividades y demás individuos concretos. Esta separación, habitual en la ingeniería ontológica, facilita el mantenimiento y la reutilización, pues permite evolucionar el esquema con cierta independencia de los datos poblados.

El esquema resultante comprende 20 clases, 21 propiedades de objeto y 7 propiedades de datos, un tamaño deliberadamente contenido pero suficiente para capturar las distinciones relevantes del dominio educativo de la programación en Python. La población del grafo arrojó 157 conceptos propios, una cifra que supera el umbral de 150 conceptos fijado como objetivo en el anteproyecto, articulados mediante un conjunto de relaciones que, una vez aplicado el razonamiento, da lugar a los más de cuatro mil enunciados ya mencionados. El proceso de construcción no fue lineal, pues implicó ciclos de modelado, validación con SHACL, detección y corrección de inconsistencias, y refinamiento del esquema, en un flujo de trabajo que refleja fielmente la naturaleza iterativa que la literatura atribuye a esta actividad. La serialización del grafo se realizó en formato Turtle, una sintaxis legible y compacta para RDF, bajo el perfil OWL 2 RL que gobierna las capacidades de razonamiento.

La tabla siguiente reúne en un solo lugar las magnitudes del grafo resultante que he ido presentando a lo largo del capítulo.

Magnitud del grafo	Valor
Conceptos propios	157
Clases del esquema (`pyedu:`)	20
Propiedades de objeto	21
Propiedades de datos	7
Enunciados afirmados	1772
Enunciados tras razonamiento OWL 2 RL	4786
Enlaces `skos:exactMatch` a Wikidata	30
Relaciones `skos:broader`	19
Instancias de evaluación (relación N-aria)	diez

Un aspecto que merece atención particular es el tratamiento de la procedencia. En un sistema cuyo objetivo último es generar retroalimentación trazable y explicable, no basta con afirmar relaciones. Es necesario poder dar cuenta de su origen. Por ello, además de las relaciones binarias directas, el grafo incorpora reificación con procedencia, que permite anotar afirmaciones con metadatos sobre su fuente, y relaciones N-arias para aquellos casos en que una relación involucra a más de dos entidades o requiere ser tratada como un objeto con atributos propios. Este diseño anticipa las necesidades del módulo de explicabilidad, donde los marcadores de procedencia recuperados del grafo permiten justificar ante el usuario el porqué de cada afirmación generada.

8.5 Reutilización de vocabularios

Una buena práctica consolidada en la comunidad de la web semántica, y recomendada explícitamente por Hogan et al. (2021), es la reutilización de vocabularios existentes en lugar de la creación de términos enteramente nuevos. Reutilizar vocabularios estándar tiene ventajas evidentes. Aumenta la interoperabilidad, reduce el esfuerzo de modelado, hereda el consenso acumulado de comunidades amplias y facilita que herramientas de terceros comprendan el grafo sin necesidad de adaptaciones. En este trabajo, la reutilización se manifiesta en varios niveles. Por un lado, el esqueleto semántico se apoya

íntegramente en los estándares del W3C (RDF, RDFS, OWL, SKOS) en lugar de definir un metamodelo propio, de manera que las jerarquías de generalización emplean `skos:broader`, las correspondencias de identidad con recursos externos emplean `skos:exactMatch`, y la estructura de clases y propiedades se expresa con la maquinaria de OWL y RDFS. Por otro lado, el enlazado a Wikidata constituye una forma de reutilización de conocimiento de dominio, pues, en vez de redefinir desde cero la caracterización de conceptos universales de la programación, el grafo los ancla a entidades ya descritas y mantenidas por una comunidad externa.

Esta política de reutilización implica, no obstante, un equilibrio. El dominio específico de la enseñanza de la programación —con sus errores típicos, sus relaciones de prerrequisito pedagógico y su vinculación entre actividades y competencias— requiere términos propios que ningún vocabulario estándar proporciona con la granularidad necesaria; de ahí la existencia del espacio `pyedu:` con sus 20 clases y sus propiedades específicas. La estrategia adoptada combina, así, la reutilización de los vocabularios generales y bien establecidos para la infraestructura semántica con la definición acotada de términos propios allí donde el dominio lo exige. Con ello evita tanto la dependencia excesiva de vocabularios mal ajustados como la reinvención innecesaria de mecanismos ya estandarizados.

8.6 El grafo como base operativa

Para cerrar este marco teórico quiero subrayar que, en la arquitectura de este trabajo, el grafo de conocimiento no es un artefacto documental ni una representación estática consultada de forma ocasional, sino la base operativa sobre la que opera el sistema de recuperación aumentada. El grafo se integra en un flujo en el que, a partir del análisis del código de un estudiante, se recupera un subgrafo relevante mediante una combinación de similitud semántica (empleando embeddings) y de expansión por consultas SPARQL sobre las relaciones del grafo. Ese subgrafo recuperado, con su conocimiento explícito e inferido y con sus marcadores de procedencia, es lo que se inyecta en el contexto del modelo de lenguaje para fundamentar la retroalimentación.

Es en este uso operativo donde las capacidades discutidas a lo largo del capítulo cobran pleno sentido. La inferencia garantiza que el subgrafo recuperado contenga tanto lo afirmado como lo derivable, lo que amplía el conocimiento disponible para el modelo. La validación garantiza que ese conocimiento es estructuralmente íntegro, lo que reduce el riesgo de propagar errores hacia la generación. El enlazado conecta el dominio local con conocimiento externo verificable, lo que refuerza la fundamentación de las respuestas. Y SPARQL (W3C, 2013), como lenguaje de consulta estándar, hace posible la expansión controlada del subgrafo a partir de las relaciones semánticas, una operación que constituye el corazón del motor de recuperación. Esta concepción del grafo como infraestructura activa (y no como mero repositorio) es coherente con la visión que Hogan et al. (2021) ofrecen de los grafos de conocimiento como sustratos vivos sobre los que se construyen aplicaciones de razonamiento e integración, y justifica que la primera fase del proyecto se dedicara íntegramente a su construcción rigurosa, pues la calidad de todo lo que viene después depende de la solidez de esta base.

9 Marco teórico: LLMs, RAG y afinado eficiente

En este capítulo abordo los fundamentos del segundo gran pilar sobre el que se asienta esta propuesta, que son los modelos de lenguaje masivos (LLMs, por sus siglas en inglés *Large Language Models*) y las técnicas que permiten controlarlos, fundamentarlos y desplegarlos de manera responsable. Si el capítulo anterior se ocupaba de la representación estructurada del conocimiento mediante grafos, aquí me centro en el componente generativo, en el sistema capaz de producir lenguaje natural que sirva de vehículo al *feedback* formativo. La tesis que recorre todo este trabajo es que ninguno de los dos pilares basta por sí solo. El grafo aporta precisión, trazabilidad y una semántica explícita pero carece de fluidez comunicativa, mientras que el modelo de lenguaje aporta naturalidad y capacidad de explicación pero adolece de problemas serios de fiabilidad cuando opera sin anclaje. La arquitectura de generación aumentada por recuperación (RAG) y el *fine tuning* eficiente de parámetros son los puentes que permiten combinar ambas fortalezas, y a explicar su fundamento dedico las páginas que siguen.

9.1 Modelos de lenguaje generativos y sus límites

Los modelos de lenguaje masivos son redes neuronales basadas en la arquitectura *Transformer* que, entrenadas sobre corpus de texto de escala masiva mediante objetivos auto-supervisados de predicción del siguiente token, adquieren una notable capacidad para generar, resumir, traducir, clasificar y razonar sobre lenguaje natural. Su éxito en una amplísima variedad de tareas sin necesidad de entrenamiento específico (el llamado aprendizaje en contexto) ha transformado el panorama de la inteligencia artificial aplicada en apenas unos años. En el dominio concreto de la programación, modelos especializados como los de la familia Codex demostraron que es posible sintetizar código funcionalmente correcto a partir de descripciones en lenguaje natural y, recíprocamente, razonar sobre código existente para explicarlo o depurarlo (Chen et al., 2021). Esta capacidad bidireccional (comprender código y comprender lenguaje) es la que hace de los LLMs un candidato natural para construir sistemas de retroalimentación educativa, pues un modelo capaz de leer la solución de un estudiante, identificar dónde se desvía de lo esperado y verbalizar esa desviación en términos pedagógicamente útiles parece, a primera vista, la herramienta ideal.

Esta primera impresión, sin embargo, debe matizarse con considerable cautela. El problema más documentado y más grave de los modelos generativos es la *alucinación*. Designa la tendencia a producir afirmaciones fluidas, plausibles y gramaticalmente impecables que, no obstante, son factualmente incorrectas o carecen de fundamento en cualquier fuente verificable. Zhang et al. (2023) ofrecen una taxonomía y un análisis sistemático de este fenómeno, distinguiendo entre alucinaciones de fidelidad (cuando el modelo contradice la información proporcionada en el contexto) y alucinaciones de factualidad (cuando contradice el conocimiento del mundo). La raíz del problema es estructural y no meramente anecdótica. Un LLM es, en su núcleo, un estimador de la probabilidad del siguiente token condicionada al texto previo, optimizado para producir continuaciones verosímiles, no continuaciones verdaderas. Cuando la verosimilitud estadística y la verdad coinciden, el modelo acierta; cuando divergen, el modelo no dispone de ningún mecanismo intrínseco que le obligue a preferir la verdad. En un contexto educativo esta limitación resulta especialmente peligrosa, porque un *feedback* incorrecto formulado con la autoridad y la fluidez de un experto puede inducir al estudiante a interiorizar conceptos erróneos, con lo que socava el objetivo formativo que se persigue. Una explicación equivocada sobre por qué una recursión no termina, o la atribución de un error de mutabilidad a una causa que no es la real, perjudica activamente el aprendizaje en lugar de ayudarlo. La Figura 15 resume esta doble taxonomía —fidelidad frente a factualidad— y su raíz estructural.

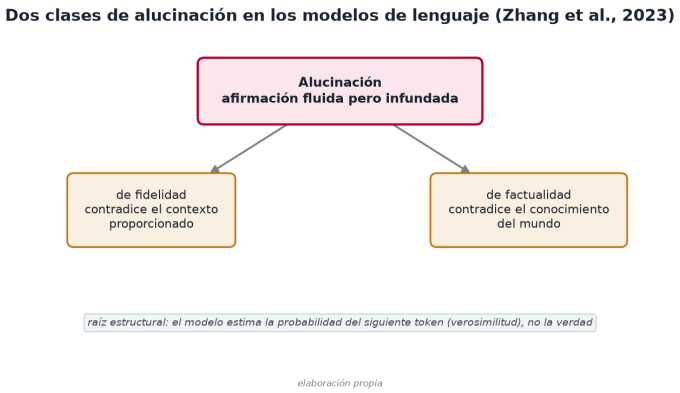

Figura 15. Las dos clases de alucinación (fidelidad y factualidad) y su raíz estructural. *Fuente: elaboración propia a partir de Zhang et al. (2023).*

A la preocupación por la alucinación se suma una crítica de calado más profundo, articulada de manera influyente por Bender et al. (2021). Estas autoras introdujeron la metáfora del *loro estocástico* para subrayar que un modelo de lenguaje, por sofisticada que sea su salida, no comprende el significado de lo que produce en ningún sentido robusto del término. Se limita a recombinar patrones formales observados en sus datos de entrenamiento sin un modelo del mundo subyacente ni una intención comunicativa genuina. Su trabajo señala además riesgos asociados que trascienden lo puramente técnico, como el coste medioambiental del entrenamiento a gran escala, la opacidad de corpus tan vastos que resultan imposibles de auditar, y la consiguiente perpetuación de sesgos sociales (de género, raza, clase o cultura) presentes en los datos. Estas advertencias son directamente pertinentes para mi trabajo y resuenan con las consideraciones éticas que el anteproyecto recogía explícitamente. Si el sistema ha de operar en un entorno educativo, donde la equidad y la ausencia de sesgo no son aspiraciones opcionales sino requisitos, no puedo limitarme a confiar en la salida bruta de un modelo. La reflexión de Bender et al. (2021) refuerza así la conveniencia de un diseño en el que el conocimiento de dominio resida en una estructura explícita, inspeccionable y corregible por docentes (el grafo de conocimiento educativo) y no quede disuelto e inaccesible en los parámetros del modelo.

Quiero además situar estas limitaciones en el plano específico de la evaluación de programación, que es el terreno de aplicación de este TFM. La literatura sobre retroalimentación automática en la enseñanza de programación lleva tiempo señalando que generar *feedback* genuinamente formativo —no un mero veredicto de correcto o incorrecto, sino una explicación que ayude al estudiante a comprender y corregir su modelo mental— es una tarea difícil que las herramientas tradicionales basadas en juegos de pruebas resuelven solo parcialmente (Keuning et al., 2019; Marwan et al., 2020). Los LLMs prometen cerrar esa brecha por su capacidad expresiva, pero arrastran consigo el riesgo de fabricar explicaciones plausibles y falsas. El reto técnico que articula este trabajo puede formularse con precisión, y consiste en aprovechar la fluidez generativa de los LLMs sin heredar su falta de fiabilidad. Las dos familias de técnicas que examino a continuación (la generación aumentada por recuperación y el *fine tuning* eficiente) constituyen las dos respuestas complementarias que la comunidad ha desarrollado para este problema, y son las dos vías que combino en la arquitectura propuesta.

9.2 Generación aumentada por recuperación (RAG)

La generación aumentada por recuperación, propuesta de manera seminal por Lewis et al. (2020), parte de una idea conceptualmente simple pero de consecuencias profundas. En lugar de exigir que todo el conocimiento necesario para responder resida en los parámetros del modelo (lo que se denomina conocimiento *paramétrico*), se acopla al modelo generativo un componente de recuperación que, ante cada consulta, localiza en una base de conocimiento externa los fragmentos más relevantes y los inyecta en el contexto del modelo como conocimiento *no paramétrico*. El LLM deja así de ser un oráculo que responde de memoria para convertirse en un razonador que sintetiza una respuesta a partir de evidencia

que tiene delante. La formulación original de Lewis et al. (2020) combinaba un recuperador denso basado en *embeddings*[10] con un generador secuencia a secuencia, entrenando ambos de forma conjunta. La arquitectura canónica de un sistema RAG comprende, por tanto, tres etapas —la indexación de la base de conocimiento en un espacio vectorial, la recuperación de los fragmentos más próximos a la consulta del usuario y la generación condicionada por esos fragmentos recuperados—, las tres fases que recoge Figura 16.

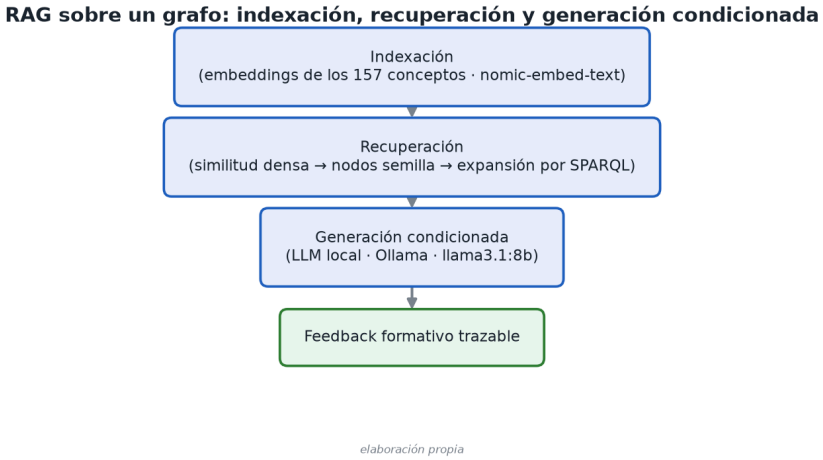

Figura 16. RAG sobre un grafo: indexación, recuperación, generación condicionada.

Las ventajas de este paradigma para mi problema son directas y resuelven varias de las limitaciones expuestas en la sección anterior. En primer lugar, RAG mitiga la alucinación al anclar la generación en fuentes concretas, porque cuando el modelo dispone en su contexto del fragmento de conocimiento pertinente la probabilidad de que invente una afirmación contradictoria disminuye sustancialmente, ya que la continuación verosímil pasa a coincidir con la continuación fiel a la evidencia. En segundo lugar, separa el conocimiento del modelo del modelo mismo, lo que permite actualizar, corregir o auditar la base de conocimiento sin necesidad de reentrenar la red; un docente puede así enmendar un concepto mal representado editando el grafo, no manipulando millones de parámetros opacos. En tercer lugar, y de manera crucial para el componente de explicabilidad que el anteproyecto situaba como objetivo O4, RAG abre la puerta a la *trazabilidad*. Dado que cada respuesta se construye a partir de fragmentos identificables, es posible señalar al usuario qué piezas de conocimiento sustentan cada afirmación, lo que dota al sistema de una procedencia explícita que la generación pura no puede ofrecer. Esta propiedad conecta de manera natural con la reificación y los marcadores de procedencia que se modelaron en el grafo de conocimiento.

Ahora bien, la formulación clásica de RAG asume que la base de conocimiento es una colección de documentos textuales planos, troceados en fragmentos e indexados independientemente unos de otros. Esta suposición encaja mal con la naturaleza de mi base de conocimiento, que no es un corpus de texto sino un grafo de 157 conceptos propios interconectados por relaciones semánticas explícitas —jerarquías de prerrequisitos, asociaciones entre conceptos y errores, vínculos de especialización— enriquecidas tras la inferencia OWL-RL hasta los 4786 enunciados. Recuperar fragmentos aislados de semejante estructura desperdiciaría lo que la hace valiosa, que es la topología relacional. Si un estudiante comete un error de mutabilidad de listas, no basta con recuperar la ficha del concepto «lista»; el valor pedagógico reside en navegar el subgrafo que conecta ese concepto con sus prerrequisitos, con los errores típicos asociados y con los conceptos vecinos que el estudiante podría estar confundiendo. La recuperación plana ignora esta riqueza estructural.

[10] representaciones vectoriales del texto en las que la proximidad geométrica aproxima la similitud semántica

Es aquí donde entra GraphRAG, la variante propuesta por Edge et al. (2024) que adapta el paradigma de recuperación aumentada a bases de conocimiento estructuradas como grafos. La idea central consiste en explotar la estructura del grafo durante la recuperación, ya sea recorriendo relaciones para extraer subgrafos conexos en torno a las entidades relevantes, ya sea aprovechando comunidades o resúmenes jerárquicos del grafo para responder a consultas globales que ningún fragmento aislado podría satisfacer. Frente al RAG clásico, que recupera *trozos* de texto, GraphRAG recupera *vecindarios* de conocimiento y preserva las relaciones que dan sentido a cada nodo. Esta aproximación es la que adopto en la arquitectura de este TFM, y su pertinencia es difícil de exagerar. El subgrafo recuperado aporta al modelo los conceptos implicados y, con ellos, la trama de relaciones que los conecta, lo que permite generar un *feedback* que sitúa el error del estudiante en su contexto conceptual.[11] En la práctica, mi motor de recuperación combina una primera fase de similitud densa, que identifica los nodos del grafo semánticamente más próximos al código analizado mediante *embeddings*, con una segunda fase de expansión mediante consultas SPARQL que recorre las relaciones para extraer el subgrafo relevante. Esta combinación de recuperación vectorial y expansión simbólica materializa el espíritu de GraphRAG sobre la infraestructura de web semántica descrita en el capítulo anterior. El contraste entre el RAG clásico, que recupera trozos de texto independientes, y GraphRAG, que recupera vecindarios de conocimiento preservando las relaciones, se ilustra en la Figura 17.

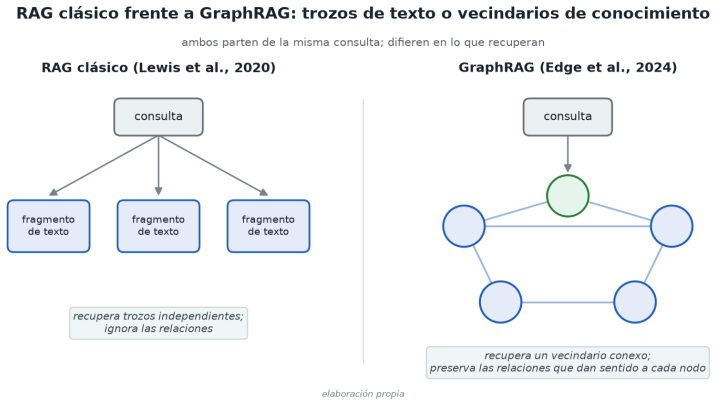

Figura 17. RAG clásico (trozos de texto aislados) frente a GraphRAG (subgrafo conexo). *Fuente: elaboración propia.*

El interés por la confluencia entre grafos de conocimiento y modelos de lenguaje no es exclusivo de GraphRAG, sino que se inscribe en una corriente más amplia que la literatura reciente ha sistematizado. Gupta et al. (2024) ofrecen una panorámica de las distintas estrategias para integrar conocimiento estructurado en sistemas basados en LLMs, distinguiendo entre aproximaciones que inyectan el grafo durante el entrenamiento, aproximaciones que lo utilizan en tiempo de inferencia mediante recuperación (la familia a la que pertenece este trabajo) y aproximaciones híbridas. Su análisis subraya que los grafos de conocimiento aportan a los modelos de lenguaje aquello de lo que estos carecen, a saber, una representación factual explícita, verificable y actualizable, capaz de servir de contrapeso a la tendencia alucinatoria. Recíprocamente, los modelos de lenguaje aportan a los grafos la interfaz en lenguaje natural y la capacidad de razonamiento flexible que las consultas estructuradas por sí solas no proporcionan. Esta complementariedad bidireccional, que Gupta et al. (2024) documentan a través de múltiples casos de uso, constituye el fundamento teórico que justifica la apuesta arquitectónica de mi propuesta. El sistema que construyo puede leerse, en este marco, como una instancia concreta de integración neuro-simbólica en tiempo de inferencia, orientada al dominio específico de la evaluación formativa de programación.

[11] indicando, por ejemplo, qué prerrequisito no se ha consolidado o qué confusión subyace

Las decisiones de implementación se apartaron de lo previsto en el anteproyecto. Aquel documento contemplaba una pila tecnológica basada en el modelo de *embeddings* all-MiniLM-L6-v2, el índice vectorial FAISS y la base de datos de grafos Neo4j. Durante el desarrollo opté, sin embargo, por nomic-embed-text como modelo de *embeddings* —que ofrece representaciones de mayor dimensionalidad y mejor rendimiento en recuperación, además de integrarse de forma nativa con el entorno de despliegue local que describo más adelante— y por la combinación de rdflib y GraphDB para la gestión del grafo, en coherencia con la elección de RDF y SPARQL como sustrato semántico frente al modelo de propiedades de Neo4j. Esta decisión privilegió la fidelidad a los estándares de la web semántica y el razonamiento OWL sobre la conveniencia de un grafo de propiedades, y la documento aquí no como una desviación menor sino como una elección deliberada que el lector debe poder valorar.

9.3 Afinado eficiente de parámetros: LoRA y QLoRA

La generación aumentada por recuperación resuelve el problema del anclaje del conocimiento, pero deja intacta otra cuestión, que es la adecuación del *estilo* y la *competencia de tarea* del modelo. Un LLM de propósito general puede saber programar y puede recibir el contexto adecuado vía RAG, y aun así producir retroalimentación cuyo tono, estructura o granularidad no se ajusten a lo que la situación pedagógica requiere. Para especializar un modelo en una tarea concreta la vía tradicional es el *fine tuning*, continuar el entrenamiento sobre datos específicos de la tarea. El *fine tuning* completo de un modelo de varios miles de millones de parámetros es, sin embargo, prohibitivo. Actualizar todos los pesos exige una memoria de GPU y un coste computacional fuera del alcance de la mayoría de los entornos, incluido el de un TFM. De ahí el interés de las técnicas de *fine tuning* eficiente de parámetros (PEFT), que persiguen adaptar un modelo grande modificando solo una fracción minúscula de sus pesos.

Tabla. Contraste entre la recuperación aumentada (RAG/GraphRAG) y el afinado eficiente (LoRA/QLoRA) según las propiedades discutidas en este capítulo: qué adapta cada técnica, dónde reside el conocimiento, su coste y su trazabilidad.

Dimensión	RAG / GraphRAG	Afinado eficiente (LoRA / QLoRA)
Qué adapta	El conocimiento disponible: ancla la generación en evidencia recuperada	El estilo y la competencia de tarea del modelo
Dónde reside el conocimiento	Externo y no paramétrico (grafo o base de conocimiento)	En los parámetros del modelo (adaptadores de bajo rango)
Actualización del conocimiento	Editar la base sin reentrenar la red	Reentrenar o reafinar los adaptadores
Coste de cómputo	Sin afinado; el coste recae en la indexación y la recuperación	Afina solo una fracción de los pesos; base cuantizada a 4 bits, viable en una sola GPU
Trazabilidad	Alta: procedencia explícita de cada afirmación	Baja: el conocimiento queda disuelto en los pesos
Límite que contrarresta	Alucinación y falta de anclaje factual	Inadecuación de tono, estructura y granularidad de la tarea

La técnica de PEFT que adopto es LoRA (*Low-Rank Adaptation*), introducida por Hu et al. (2021). Su fundamento descansa en una observación empírica con respaldo teórico, según la cual la actualización que el *fine tuning* induce sobre las matrices de pesos de un *Transformer* tiende a ser de *rango intrínseco bajo*, es decir, puede aproximarse bien por una matriz de rango reducido. En lugar de actualizar directamente una matriz de pesos preentrenada (que permanece congelada), LoRA aprende un par de matrices pequeñas cuyo producto, de rango limitado por un hiperparámetro *r*, representa el

incremento de pesos. Si la matriz original tiene dimensiones de millares por millares, las dos matrices de bajo rango contienen órdenes de magnitud menos parámetros entrenables. El resultado es doble. El coste de memoria y cómputo del *fine tuning* se desploma, y los pesos preentrenados quedan intactos, lo que permite mantener un único modelo base y conmutar entre múltiples adaptadores ligeros según la tarea. En mi configuración empleé un rango *r* de 16 con un factor de escala *alpha* de 32, valores moderados que ofrecen suficiente capacidad de adaptación sin sobredimensionar el número de parámetros entrenables.

LoRA reduce drásticamente el coste del *fine tuning*, pero el modelo base sigue debiendo residir en memoria durante el entrenamiento, y para un modelo de siete mil millones de parámetros en precisión de 16 bits esto continúa siendo exigente. Aquí interviene QLoRA, la contribución de Dettmers et al. (2023) que hace posible afinar modelos de gran tamaño en una sola GPU de consumo. QLoRA combina LoRA con la *cuantización* del modelo base a 4 bits. Los pesos congelados se almacenan en una representación de precisión reducida mientras los adaptadores de bajo rango se entrenan en precisión mayor. Su innovación técnica más destacada es el tipo de dato NormalFloat de 4 bits (nf4), diseñado para representar de manera óptima pesos distribuidos normalmente, junto con la doble cuantización y la paginación de optimizadores para gestionar los picos de memoria. El logro de Dettmers et al. (2023) fue demostrar que esta combinación de cuantización agresiva y adaptación de bajo rango preserva la calidad del *fine tuning* casi sin pérdida frente al *fine tuning* en precisión completa y democratiza así el acceso al *fine-tuning* de modelos masivos.

Sobre estos fundamentos construí el Sistema B de mi evaluación comparativa, afinando el modelo Qwen2.5-Coder-7B-Instruct mediante QLoRA con cuantización de 4 bits en formato nf4, los hiperparámetros LoRA mencionados (*r*=16, *alpha*=32) y una GPU RTX 5090. El proceso, sin embargo, no estuvo exento de dificultades, que relato aquí porque ilustran riesgos genéricos del *fine tuning* sobre conjuntos de datos pequeños. El conjunto de entrenamiento se generó sintéticamente a partir de plantillas, y la evaluación se realizó exclusivamente sobre esqueletos *held-out* (es decir, sobre estructuras de problema no vistas durante el entrenamiento) como salvaguarda anti-fuga, para evitar que el modelo se limitara a memorizar ejemplos. Pese a esta precaución, el primer experimento sin regularización mostró un patrón inequívoco de *sobreajuste*. La pérdida sobre el conjunto reservado empeoró época tras época (1,297, 1,380 y 1,410 respectivamente), señal clara de que el modelo se ajustaba cada vez mejor al entrenamiento a costa de generalizar peor. Para mitigarlo introduje dos mecanismos de regularización, NEFTune con un *alpha* de 5 (que inyecta ruido controlado en los *embeddings* durante el entrenamiento) y un *dropout* de LoRA de 0,1. El efecto fue notable, pues la pérdida *held-out* descendió a 1,051 y 1,028, y la precisión de token sobre el conjunto reservado alcanzó 0,842. No quiero, en cualquier caso, sobrevender estos resultados. La muestra es pequeña, los datos son sintéticos y la mejora, aunque consistente, debe interpretarse con la prudencia que impone la escala del experimento. Lo reporto como evidencia de un procedimiento metodológicamente cuidadoso, no como prueba concluyente de un rendimiento generalizable.

La comparación entre el modelo afinado y las demás configuraciones resultó esclarecedora respecto a la complementariedad de las dos técnicas que vengo exponiendo. Sobre los 50 casos *held-out* evaluados con un juez LLM (qwen2.5:32b), el Sistema B afinado mejoró sustancialmente la clasificación de la categoría del error (acierto de 0,70 frente a 0,26 del modelo base) y la calidad de la identificación técnica del problema (3,80 sobre 5 frente a 2,04), pero no destacó en trazabilidad. El Sistema C, basado en GraphRAG sin *fine tuning*, mostró el patrón opuesto, pues sobresalió en la identificación del concepto pedagógico correcto (0,48 frente a 0,18) y, muy especialmente, en trazabilidad (2,92 frente a 1,72), pero quedó por detrás de B en la clasificación del error. Esta divergencia confirma de manera empírica la intuición teórica. El *fine tuning* especializa la competencia técnica del modelo, mientras que la recuperación aumentada aporta el anclaje conceptual y la procedencia. Ninguna de las dos técnicas domina a la otra en todas las dimensiones, lo que motiva directamente el Sistema D híbrido, que combina *fine tuning* y GraphRAG con la esperanza de heredar las fortalezas de ambos. La evaluación ampliada de las cuatro

configuraciones se completó con estos cincuenta casos *held-out*, y su resultado confirma esa esperanza, porque el Sistema D híbrido sintetiza las fortalezas complementarias de B y C, gana o empata en las siete dimensiones consideradas (es el mejor en seis de ellas) y se sitúa por delante de B y de C, que a su vez superan al modelo base A. El híbrido lidera tanto la clasificación de la categoría (0,76, recuperando la fortaleza técnica de B) como la trazabilidad (3,16, heredando el anclaje conceptual de C), y queda muy cerca del máximo en identificación del concepto (0,54) y en identificación técnica (4,04, el valor más alto de las cuatro configuraciones). En las dimensiones del juez ocurre lo propio, ya que D encabeza la valoración divulgativa (3,66 frente a 3,52 de A, 3,38 de B y 3,20 de C) y la técnica (3,62 frente a 3,02 de A, 3,44 de B y 2,44 de C), mientras que la dimensión de sugerencia accionable se reparte en un empate en torno a 2,9 entre las cuatro configuraciones. La complementariedad de B y C, que en la fase preliminar solo se intuía, queda así confirmada y, sobre todo, sintetizada por el sistema híbrido. Subrayo, en línea con el compromiso de integridad que vertebra esta memoria, que ninguno de estos resultados alcanza los objetivos numéricos ambiciosos que el anteproyecto fijaba como hipótesis (una precisión diagnóstica del 85 % o más, alucinaciones por debajo del 5 %). Ni siquiera el Sistema D, el mejor de los cuatro, supera el 0,76 de acierto en categoría, lejos del 85 % perseguido. Lo que reporto son las cifras reales obtenidas sobre una muestra de datos sintéticos, evaluadas de forma automática, sin el panel de docentes humanos ni el análisis de concordancia inter-juez que quedan como trabajo futuro.

Tabla. Evaluación comparativa sobre los 50 casos held-out con el juez `qwen2.5:32b`, para las configuraciones A (base), B (afinado), C (GraphRAG) y D (híbrido), con las cifras tal como las recoge la prosa de este capítulo.

Dimensión	A (base)	B (afinado)	C (GraphRAG)	D (híbrido)
Acierto de categoría del error	0,26	0,70	—	0,76
Acierto de concepto pedagógico	0,18	—	0,48	0,54
Identificación técnica del problema	2,04	3,80	—	4,04
Trazabilidad	1,72	—	2,92	3,16
Valoración divulgativa del juez	3,52	3,38	3,20	3,66
Valoración técnica del juez	3,02	3,44	2,44	3,62
Sugerencia de mejora	≈2,9	≈2,9	≈2,9	≈2,9

El guion (—) marca una dimensión que la prosa de este capítulo no cuantifica para esa configuración, y «≈2,9» recoge el reparto prácticamente empatado de la sugerencia de mejora entre los cuatro sistemas.

9.4 Cuantización y despliegue local

El último elemento del marco técnico es la cuantización entendida no ya como recurso de entrenamiento sino como habilitador del *despliegue local*, una decisión que en este proyecto tiene tanto de técnica como de ética. La cuantización consiste en representar los pesos (y opcionalmente las activaciones) de una red con menor precisión numérica, típicamente enteros de 8 o 4 bits en lugar de números en coma flotante de 16 o 32 bits. Más allá de su uso en QLoRA durante el *fine tuning*, la cuantización en inferencia reduce drásticamente la huella de memoria y acelera la ejecución, hasta el punto de hacer viable correr modelos de varios miles de millones de parámetros en *hardware* modesto, sin recurrir a la nube. El coste es una degradación de la precisión numérica que, con esquemas de cuantización bien diseñados, resulta marginal frente al ahorro de recursos que proporciona.

Para el despliegue de los modelos de inferencia adopté Ollama, un entorno de ejecución que gestiona la descarga, la cuantización y el servicio local de modelos de lenguaje abiertos a través de una interfaz uniforme. Mediante Ollama integré tanto el modelo base de las configuraciones de partida (llama3.1:8b) como el modelo de *embeddings* para la recuperación (nomic-embed-text) y el modelo juez empleado en la evaluación (qwen2.5:32b), todos ellos ejecutándose en local sobre la infraestructura del proyecto sin

que ningún dato saliera del entorno controlado. Esta elección no obedece únicamente a la conveniencia o al ahorro económico de prescindir de las API comerciales. Responde, sobre todo, a un principio de soberanía de los datos que el anteproyecto recogía entre sus consideraciones éticas y que considero irrenunciable en un sistema educativo, ya que el código que un estudiante entrega, sus errores y su trayectoria de aprendizaje constituyen información sensible cuyo tratamiento en servidores de terceros plantea problemas de privacidad difíciles de justificar. Procesar todo localmente garantiza que esos datos permanezcan bajo el control de la institución y del propio estudiante, y elimina de raíz la dependencia de proveedores externos cuyas políticas, disponibilidad o coste escapan al control del docente.

El despliegue local introduce, no obstante, un compromiso que conecta directamente con el objetivo O5 del anteproyecto, el análisis de los compromisos de diseño. Operar en local impone un techo de recursos que obliga a elegir cuidadosamente el tamaño del modelo, el grado de cuantización y la profundidad de la recuperación. Un modelo mayor o una recuperación más exhaustiva mejoran potencialmente la calidad del *feedback* pero incrementan la latencia y el consumo de memoria, hasta hacer inviable la ejecución en *hardware* convencional; un modelo más pequeño o una cuantización más agresiva aligeran el sistema a costa de arriesgar la precisión de las explicaciones. Estos compromisos no son accesorios sino consustanciales a un sistema concebido para ser desplegado de forma sostenible y soberana en un contexto educativo real, y su análisis constituye una de las aportaciones que persigo. La cuantización, en suma, no es un mero detalle de implementación sino la condición de posibilidad de un despliegue que respete la privacidad, y los compromisos que comporta forman parte legítima del objeto de estudio de este trabajo.

En conjunto, los conceptos revisados en este capítulo —las capacidades y los límites de los modelos generativos, la fundamentación mediante RAG y GraphRAG, el *fine tuning* eficiente con LoRA y QLoRA, y la cuantización al servicio de un despliegue local soberano— componen el armazón técnico sobre el que se levanta la arquitectura del sistema. Cada una de estas piezas responde a una limitación concreta. La alucinación y la falta de comprensión se contrarrestan con el anclaje en el grafo vía recuperación; la inadecuación de tarea se corrige con *fine tuning* eficiente; y la tensión entre capacidad y privacidad se gestiona con cuantización y ejecución local. El capítulo dedicado a la arquitectura mostrará cómo estas piezas se ensamblan en un sistema coherente que combina, en su configuración híbrida, las fortalezas complementarias que la evaluación comparativa ha puesto de manifiesto.

10 Marco teórico: evaluación formativa y feedback

La motivación última de este Trabajo de Fin de Máster no es técnica, sino pedagógica. Un sistema que integra grafos de conocimiento educativos con modelos de lenguaje masivos solo tiene sentido si el feedback que produce ayuda realmente a quien aprende a programar. Por ello me detengo, antes de abordar las arquitecturas que articulan el resto de la memoria, en el corpus teórico que define qué es un buen feedback formativo y cómo se categoriza. Importa también el papel que desempeñan las rúbricas y las concepciones erróneas, así como las dificultades específicas que plantea la evaluación del código de los estudiantes principiantes. Este capítulo no pretende agotar un campo tan amplio como el de la evaluación educativa; busco delimitar los conceptos que después operacionalizo en la rúbrica de cinco dimensiones, en el diseño de los prompts y en los criterios con los que el juez LLM valora las respuestas de los distintos sistemas. La tesis de fondo que recorre todo el capítulo es que la calidad del feedback no se mide por su extensión ni por su corrección superficial, sino por su capacidad de cerrar la distancia entre el estado actual del estudiante y el objetivo de aprendizaje, y que esa capacidad depende de propiedades (diagnóstico preciso, explicación fundamentada, adaptación al nivel) que un modelo de lenguaje, por sí solo, no garantiza.

10.1 El poder del feedback como variable de aprendizaje

El punto de partida ineludible es la síntesis de Hattie y Timperley (2007), que constituye una de las referencias más citadas de la investigación educativa contemporánea. A partir de la integración de un volumen considerable de metaanálisis, estos autores estiman que el feedback se sitúa entre las influencias de mayor magnitud sobre el rendimiento del alumnado, con un tamaño del efecto medio de d=0.79, muy por encima del umbral que la propia literatura suele considerar como indicativo de un efecto educativamente relevante. Esta cifra, citada con frecuencia de manera acrítica, requiere sin embargo dos matizaciones que resultan centrales para esta memoria. La primera es que se trata de un promedio que esconde una varianza enorme. No todo feedback ayuda, y algunas formas de retroalimentación tienen efectos nulos o incluso negativos. La segunda es que el propio Hattie y Timperley (2007) advierten de que el feedback solo es poderoso cuando responde adecuadamente a tres preguntas que el aprendiz se formula, explícita o implícitamente, durante el proceso de aprendizaje.

Esas tres preguntas estructuran su modelo y, por extensión, buena parte del diseño de este trabajo. La primera, *¿hacia dónde voy?* (*feed up*), remite a los objetivos de aprendizaje y a los criterios de logro; sin una meta clara, el estudiante no puede interpretar la información que recibe. La segunda, *¿cómo voy?* (*feed back* en sentido estricto), confronta el desempeño actual con esa meta y constituye el núcleo diagnóstico de la retroalimentación. La tercera, *¿hacia dónde después?* (*feed forward*), orienta los pasos siguientes y es, según los autores, la dimensión más frecuentemente descuidada y a la vez la de mayor potencial. A estas tres preguntas, Hattie y Timperley (2007) superponen cuatro niveles en los que el feedback puede operar: la tarea (corrección de un resultado concreto), el proceso (las estrategias subyacentes a la tarea), la autorregulación (la capacidad del estudiante de monitorizar su propio aprendizaje) y el self (elogios o juicios sobre la persona). Su conclusión, que asumo como principio rector, es que el feedback centrado en el proceso y en la autorregulación es el más transferible y duradero, mientras que el feedback centrado en el self (el «muy bien» o el «estás equivocado» sin más) es el menos eficaz e incluso contraproducente. La Figura 18 sintetiza este modelo de las tres preguntas y los cuatro niveles en los que opera la retroalimentación.

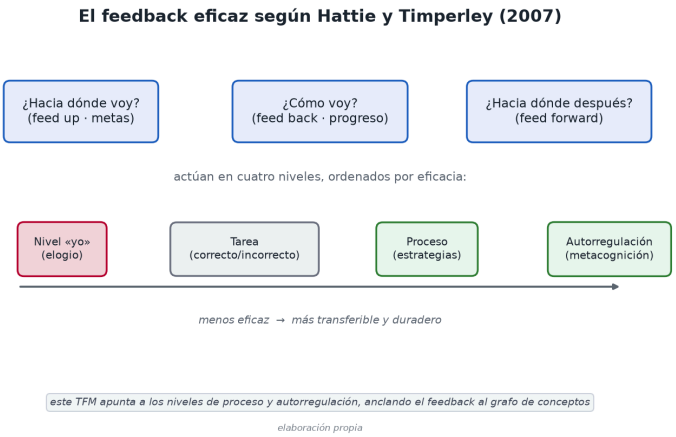

Figura 18. Feedback eficaz según Hattie y Timperley (2007). *Fuente: elaboración propia a partir de Hattie y Timperley (2007).*

Esta jerarquía tiene consecuencias directas sobre cómo evalúo los sistemas A, B, C y D. Un modelo de lenguaje sin augmentación tiende, por su propia naturaleza generativa, a producir respuestas fluidas y plausibles que con frecuencia se quedan en el nivel de la tarea, donde señalan que hay un error y proponen una corrección puntual, pero rara vez explicitan el proceso conceptual que lo subyace ni orientan la autorregulación del estudiante. La hipótesis pedagógica de este trabajo es que anclar la generación en un grafo de conocimiento —es decir, en una representación explícita de los conceptos de programación y de sus relaciones— empuja el feedback hacia el nivel del proceso, porque obliga al sistema a nombrar el concepto implicado y a situarlo en su red de prerrequisitos. Los resultados reales obtenidos, que detallo en capítulos posteriores, ofrecen un apoyo parcial y matizado a esta hipótesis. El sistema con GraphRAG (C) mejora la trazabilidad y la calidad conceptual del feedback, mientras que el modelo afinado (B) mejora la clasificación del error y la explicación técnica. Que ninguno de los dos domine en todas las dimensiones es coherente con la idea de Hattie y Timperley (2007) de que el feedback es multidimensional, y es lo que motiva la propuesta de un sistema híbrido.

10.2 Una taxonomía del feedback en la enseñanza de la programación

Mientras que el modelo de Hattie y Timperley (2007) es general, la enseñanza de la programación dispone de una sistematización propia que resulta más operativa para los fines de esta memoria. La revisión de Keuning et al. (2019) examina un amplio conjunto de herramientas de tutorización y de generación automática de feedback para ejercicios de programación, y propone una taxonomía que distingue varios tipos de retroalimentación según la información que aportan al estudiante. Para los propósitos de este trabajo resulta especialmente útil la distinción entre feedback diagnóstico, correctivo y explicativo, que retomo y operacionalizo a lo largo de la memoria.

El feedback diagnóstico identifica qué falla; detecta la presencia de un error, lo localiza y, en su forma más elaborada, lo clasifica dentro de una categoría (un error de tipo, una condición de bucle mal formulada, un caso límite no contemplado). Es el sustrato de cualquier retroalimentación posterior, porque sin un diagnóstico correcto las propuestas de mejora carecen de fundamento. El feedback correctivo va un paso más allá e indica cómo solucionarlo, ya sea señalando la modificación concreta que debe realizarse o, en el extremo, proporcionando directamente la solución; aquí Keuning et al. (2019) advierten de un riesgo pedagógico bien conocido, el de que un feedback excesivamente directivo sustituya el esfuerzo cognitivo del estudiante y reduzca el aprendizaje a la copia de la corrección sugerida. El feedback explicativo, por último, articula por qué se produce el error y por qué la solución propuesta es correcta, conectando el caso particular con el principio o concepto general que lo gobierna; es el que mejor se alinea con el nivel del proceso de Hattie y Timperley (2007) y el que, según la revisión, aparece con menor frecuencia en las herramientas automáticas, las cuales tienden a concentrarse en la detección y la corrección.

La distinción entre estos tres tipos de feedback puede resumirse en la tabla siguiente.

Tipo de feedback	Pregunta que responde	Qué aporta al estudiante	Consideración pedagógica
Diagnóstico	¿Qué falla?	Detecta el error, lo localiza y, en su forma más elaborada, lo clasifica dentro de una categoría	Es el sustrato de cualquier retroalimentación posterior
Correctivo	¿Cómo se soluciona?	Señala la modificación concreta que debe realizarse o, en el extremo, proporciona directamente la solución	Si es excesivamente directivo, sustituye el esfuerzo cognitivo y reduce el aprendizaje a la copia de la corrección
Explicativo	¿Por qué?	Articula por qué se produce el error y por qué la solución es correcta, conectando el caso particular con el concepto general	Se alinea con el nivel del proceso; es el menos frecuente en las herramientas automáticas

Fuente: elaboración propia a partir de Keuning et al. (2019).

Esta taxonomía estructura la evaluación experimental de esta memoria de un modo muy concreto. La rúbrica de cinco dimensiones que aplico distingue la identificación o diagnóstico del error de su explicación conceptual y de la calidad de la orientación correctiva, porque, siguiendo a Keuning et al. (2019), un sistema puede ser bueno en una dimensión y deficiente en otra. Los resultados reales lo confirman con claridad. En los 50 casos held-out evaluados por el juez LLM (qwen2.5:32b), el sistema afinado (B) alcanza un acierto de categoría de 0.70 frente al 0.26 del modelo base (A), lo que indica una capacidad diagnóstica muy superior en la clasificación del tipo de error; en cambio, el sistema con GraphRAG (C) logra un acierto de concepto de 0.48 frente al 0.18 del base, lo que señala una mejor explicación conceptual, una mejor respuesta a la pregunta del porqué. Que el modelo afinado destaque en el diagnóstico categorial y el modelo augmentado con el grafo destaque en la explicación conceptual no es casual, porque el fine-tuning especializa al modelo en reconocer patrones de error frecuentes, mientras que el grafo le aporta la red de conceptos necesaria para explicarlos. Reconozco aquí una limitación importante, que la muestra de 50 casos sigue siendo modesta, y aunque las diferencias son apreciables, deben leerse como indicios y no como estimaciones estables. La evaluación ampliada con n=50 que incorpora además el sistema híbrido (D) ya se ha completado, y el Sistema D resultó el mejor del conjunto y ganó o empató en las siete dimensiones evaluadas, con un acierto de categoría de 0.76 y un acierto de concepto de 0.54.

10.3 Rúbricas pedagógicas como instrumento de evaluación

El segundo pilar teórico de la evaluación que aquí se realiza son las rúbricas. Una rúbrica es, en su definición más sencilla, un instrumento que descompone una evaluación compleja en un conjunto de dimensiones o criterios, cada uno descrito mediante niveles de logro ordenados. Su valor reside en que hacen explícitos y comunicables los estándares de calidad, transformando juicios globales y difíciles de justificar en valoraciones analíticas y comparables. En la enseñanza de la programación, las rúbricas permiten separar aspectos que de otro modo se confunden —la corrección funcional del código, su legibilidad, el dominio conceptual que evidencia, la adecuación de la explicación— y atribuir a cada uno una valoración independiente.

En este trabajo la rúbrica cumple una doble función. Por una parte, es el objeto que el sistema debe producir, ya que el feedback generado se organiza implícitamente en torno a las dimensiones de la rúbrica, de modo que responda a si el código funciona, a por qué falla y a cómo mejorarlo. La rúbrica es además, y de manera quizá más decisiva desde el punto de vista metodológico, el instrumento con el que evalúo a los propios sistemas. Diseñé una rúbrica de cinco dimensiones que el juez LLM aplica de forma sistemática a cada respuesta generada, asignando valoraciones que en algunas dimensiones son binarias

(acierto o no en la categoría del error, acierto o no en el concepto implicado) y en otras se gradúan en una escala de uno a cinco (calidad de la identificación del error, trazabilidad del feedback hacia las fuentes). Esta operacionalización persigue convertir un constructo pedagógico difuso, la «calidad del feedback», en un conjunto de métricas reproducibles que permiten la comparación A/B/C entre sistemas.

La naturaleza de esta evaluación y sus límites merecen una precisión, porque de ellos depende la integridad de las conclusiones. El anteproyecto preveía un panel de tres a cinco docentes humanos que aplicarían la rúbrica de manera independiente, junto con el cálculo del coeficiente de correlación intra-clase (ICC) para estimar la fiabilidad inter-juez, sobre un conjunto de en torno a trescientas submissions reales. Esa validación con jueces humanos no se ha ejecutado en el alcance temporal de este Trabajo de Fin de Máster y constituye trabajo futuro claramente identificado. La evaluación efectivamente realizada es automática, pues combina métricas objetivas con un juez LLM que aplica la rúbrica sobre datos sintéticos generados por plantillas. Esta elección tiene ventajas evidentes de escalabilidad y reproducibilidad, pero también limitaciones. Un juez basado en un modelo de lenguaje hereda los sesgos y las concepciones erróneas del propio modelo, puede mostrar preferencias sistemáticas por respuestas más largas o más fluidas, y no equivale al criterio de un docente experimentado que conoce a sus estudiantes. Por ello, los targets ambiciosos que el anteproyecto formuló como hipótesis a contrastar no se dan por cumplidos en ningún momento.[12] Son expectativas frente a las cuales contrasto los resultados reales, que son más modestos y que reporto tal como son.

10.4 Concepciones erróneas y la dificultad específica de evaluar el código del principiante

Un buen feedback diagnóstico presupone un modelo de los errores que cometen los estudiantes, y aquí la literatura sobre concepciones erróneas (*misconceptions*) resulta indispensable. Una concepción errónea no es un simple descuido ni un error de transcripción, sino una creencia estructurada y sistemática sobre cómo funciona un lenguaje o un constructo de programación que entra en conflicto con su semántica real. Ejemplos clásicos son la idea de que la asignación establece una relación de igualdad permanente entre dos variables, la confusión entre la comparación y la asignación, o la suposición de que un bucle «sabe» cuándo debe detenerse sin una condición explícita. Estas concepciones son tenaces porque tienen una coherencia interna desde la perspectiva del estudiante. Por eso el feedback eficaz no se limita a corregir el síntoma, sino que debe identificar y confrontar la creencia subyacente. Esta es una de las razones de fondo que justifican el uso de un grafo de conocimiento educativo en este trabajo, porque representar explícitamente los conceptos y sus relaciones (incluidos los prerrequisitos y las dependencias conceptuales) permite que el sistema asocie un error observable con el concepto mal comprendido que probablemente lo origina, en lugar de quedarse en la superficie del fragmento de código defectuoso.

Los ejemplos clásicos antes citados ilustran ese carácter estructurado y sistemático de las concepciones erróneas.

Constructo implicado	Concepción errónea (creencia en conflicto con la semántica real)
Asignación	La asignación establece una relación de igualdad permanente entre dos variables
Comparación frente a asignación	Confusión entre la comparación y la asignación
Bucles	Un bucle «sabe» cuándo debe detenerse sin una condición explícita

Fuente: ejemplos expuestos en este capítulo.

[12] una precisión diagnóstica de al menos el 85 %, una tasa de alucinaciones inferior al 5 %, una utilidad formativa de al menos 4.0 sobre 5 y una trazabilidad del 100 %

La dificultad de evaluar el código de los principiantes está documentada desde hace décadas. El estudio multinstitucional de McCracken et al. (2001) constituye un hito ineludible. Un grupo de investigadores de varias universidades evaluó la capacidad de estudiantes que habían completado un primer curso de programación para resolver problemas de dificultad moderada, y encontró que el rendimiento era considerablemente inferior al que el profesorado esperaba. Más allá de la cifra concreta, la contribución duradera de McCracken et al. (2001) fue evidenciar que muchos estudiantes superan los cursos introductorios sin haber consolidado las habilidades fundamentales de resolución de problemas y de traducción de una solución a código, y que esta carencia pasa a menudo desapercibida en las evaluaciones convencionales. Este hallazgo subraya la necesidad de un feedback que no se limite a verificar si el programa produce la salida correcta, sino que diagnostique la comprensión conceptual subyacente, que es lo que las pruebas de ejecución no capturan.

Sobre esta base, Watson y Li (2014) ofrecen una revisión de las tasas de aprobación en los cursos introductorios de programación a escala internacional y a lo largo del tiempo, y muestran que las dificultades documentadas por McCracken et al. (2001) son persistentes y transversales a contextos muy diversos. Su trabajo refuerza la idea de que el problema no es coyuntural ni atribuible a una institución o método concreto, sino estructural de la enseñanza de la programación, lo que justifica la inversión en herramientas de apoyo al aprendizaje como la que aquí se propone. Por su parte, Marwan et al. (2020) abordan de manera más directa el objeto de este trabajo al estudiar el efecto del feedback automático sobre el aprendizaje en entornos de programación. Su investigación analiza qué propiedades del feedback generado automáticamente lo hacen efectivo, y aporta evidencia de que el feedback que incluye explicaciones (y no solo indicaciones de la siguiente acción a realizar) favorece el aprendizaje y la percepción de utilidad por parte del estudiante, aunque también advierte de que un feedback mal calibrado puede generar dependencia o reducir la motivación. Esta conclusión conecta de manera natural con la taxonomía de Keuning et al. (2019) y con la jerarquía de Hattie y Timperley (2007), pues es el componente explicativo, el que responde al porqué y orienta el proceso, el que aporta el valor formativo diferencial.

La lectura conjunta de estas tres referencias delimita con precisión el espacio en el que se inscribe este Trabajo de Fin de Máster. McCracken et al. (2001) y Watson y Li (2014) establecen que existe un problema real, persistente y conceptual en el aprendizaje de la programación, que las evaluaciones basadas únicamente en la corrección funcional no detectan. Marwan et al. (2020) muestran que el feedback automático puede ayudar, pero solo si es explicativo y está bien calibrado. Y Keuning et al. (2019) advierten de que las herramientas automáticas existentes tienden a concentrarse en la detección y la corrección, descuidando la explicación. La propuesta de integrar un grafo de conocimiento educativo con un modelo de lenguaje mediante una arquitectura RAG busca atender esa carencia, esto es, aportar al feedback automático el componente explicativo y conceptual del que suele carecer, anclándolo en una representación verificable del dominio que reduzca el riesgo de alucinaciones y haga trazable el origen de cada afirmación.

10.5 Síntesis e implicaciones para el diseño

De este recorrido teórico extraigo los principios que guían el resto de la memoria. Primero, que el feedback es una de las influencias más potentes sobre el aprendizaje (Hattie y Timperley, 2007), pero solo cuando opera en el nivel del proceso y de la autorregulación y responde a las preguntas de hacia dónde voy, cómo voy y hacia dónde después. Segundo, que en la enseñanza de la programación distingo el feedback diagnóstico, el correctivo y el explicativo (Keuning et al., 2019), y que cada sistema puede destacar en unas dimensiones y no en otras, lo que exige una evaluación analítica mediante rúbricas. Tercero, que la rúbrica de cinco dimensiones es a la vez el modelo del feedback deseable y el instrumento de evaluación de los sistemas, con la salvedad de que la validación realizada es automática y con datos sintéticos; el panel docente, el ICC inter-juez y las submissions reales quedan como trabajo futuro. Cuarto, que las concepciones erróneas y las dificultades documentadas por McCracken et al. (2001),

Watson y Li (2014) y Marwan et al. (2020) justifican un feedback que diagnostique la comprensión conceptual y la explique, y no solo que verifique la ejecución.

Estos principios se reflejan de forma directa en los resultados reales obtenidos y en su interpretación. El hecho de que el modelo afinado (B) destaque en el diagnóstico categorial (acierto de categoría 0.70) y el modelo con grafo (C) destaque en la explicación conceptual (acierto de concepto 0.48) y en la trazabilidad (2.92 sobre 5 frente a valores inferiores del base) no es un resultado anecdótico, sino una manifestación empírica de la multidimensionalidad del feedback que la teoría anticipa. Que ambos superen al modelo base (A) y que sus fortalezas sean complementarias es lo que fundamenta, desde el marco teórico, la apuesta por un sistema híbrido (D) que combine la especialización diagnóstica del fine-tuning con la capacidad explicativa y la trazabilidad que aporta el grafo de conocimiento. La evaluación ampliada con n=50, ya completada, confirma que esa complementariedad esperada se traduce en una mejora conjunta, pues el sistema híbrido (D) resultó el mejor del conjunto y ganó o empató en las siete dimensiones, con 0.54 de acierto de concepto y 3.16 sobre 5 en trazabilidad. Reporto los resultados reales sin asimilarlos a los targets ambiciosos formulados como hipótesis en el anteproyecto, que no se alcanzan. El mejor sistema (D) llega a 0.76 en categoría, lejos del 85 % previsto, y la validación con panel humano sigue siendo trabajo futuro.

11 Metodología de investigación

11.1 Naturaleza del diseño: una investigación mixta orientada al artefacto

El trabajo que aquí se documenta se sitúa en la intersección entre la investigación en ingeniería del conocimiento y la evaluación empírica de sistemas de aprendizaje automático, y por ello adopta deliberadamente un **diseño mixto** que combina la construcción de un artefacto computacional con su validación experimental. Esta elección no es accidental ni meramente formal. Responde a la naturaleza misma del problema, que exige primero *fabricar* algo que no existe —un grafo de conocimiento educativo de la programación en Python conectado a un modelo de lenguaje mediante recuperación aumentada— y, solo después, *medir* si ese algo se comporta como la hipótesis predice. En la tradición de la *design science research*[13] el conocimiento se produce tanto por la creación del artefacto como por su evaluación rigurosa frente a una necesidad. El presente capítulo describe cómo se ha articulado esa doble producción de conocimiento en tres fases secuenciales pero interdependientes —que la Figura 19 sintetiza—, y rinde cuentas de las decisiones tomadas y de las desviaciones respecto al anteproyecto que aprobó este trabajo.

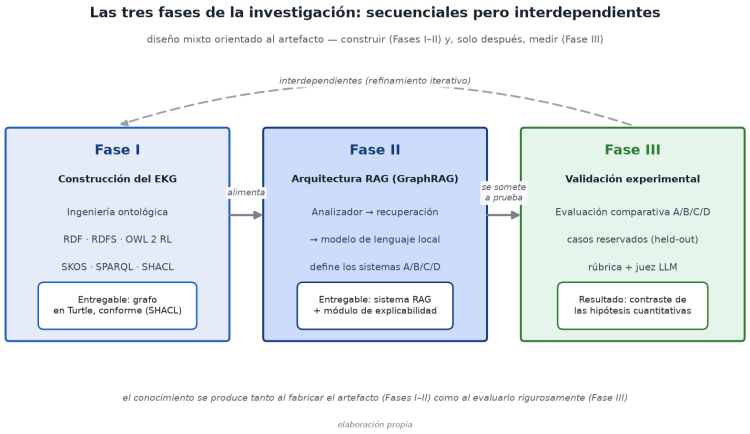

Figura 19. Las tres fases de la investigación —construcción del EKG, arquitectura RAG y validación experimental— como flujo secuencial pero interdependiente del diseño mixto orientado al artefacto.

Quiero fijar desde el principio una distinción que vertebra toda la memoria y que en este capítulo metodológico resulta especialmente delicada. El anteproyecto estableció un conjunto de **objetivos de rendimiento ambiciosos** —precisión diagnóstica igual o superior al 85 %, tasa de alucinaciones igual o inferior al 5 %, utilidad formativa media de al menos 4,0 sobre 5 y trazabilidad del 100 %— que en su momento se formularon como expectativas y aspiraciones. Metodológicamente, esos umbrales no son resultados a defender ni listones que el sistema deba franquear para que el trabajo se considere exitoso, sino **hipótesis cuantitativas a contrastar**, predicciones que el experimento puede confirmar, matizar o refutar. Buena parte del rigor de esta investigación reside en haber mantenido esa frontera nítida y en reportar lo que los datos muestran, no lo que el anteproyecto deseaba. Los resultados reales son, en muchos aspectos, más modestos que aquellas aspiraciones iniciales, y el lector los encontrará expuestos sin maquillaje en el capítulo de resultados; aquí me limito a explicar la maquinaria metodológica que los ha producido y las razones por las que confío (dentro de sus límites) en su validez.

11.2 Fase I: construcción del grafo de conocimiento educativo

La primera fase se ocupó de levantar el sustrato semántico sobre el que todo lo demás se apoya, un grafo de conocimiento educativo (EKG, por sus siglas en inglés) que modela los conceptos de la programación introductoria en Python, sus relaciones pedagógicas y los errores típicos asociados a cada uno. Metodológicamente, esta fase se concibió como un ejercicio de **ingeniería ontológica** guiado por

[13] Hevner y colaboradores han popularizado este marco en sistemas de información, aunque aquí me apoyo sobre todo en la lógica metodológica subyacente

buenas prácticas del modelado de grafos de conocimiento (Hogan et al., 2021) y anclado en los estándares del W3C para la web semántica —RDF como modelo de datos (W3C, 2014a), RDFS y OWL 2 para la axiomática de clases y propiedades (W3C, 2014b; W3C, 2012), SKOS para la organización conceptual jerárquica (W3C, 2009), SPARQL para la interrogación (W3C, 2013) y SHACL para la validación de integridad estructural (W3C, 2017).

El proceso de construcción siguió una lógica iterativa de *competency questions* y refinamiento. Partí de un catálogo de competencias y errores frecuentes documentado en la literatura sobre dificultades de aprendizaje de la programación (McCracken et al., 2001; Watson y Li, 2014) y sobre la generación automática de feedback (Keuning et al., 2019), y a partir de él fui delimitando las clases del esquema, las propiedades que las vinculan y las restricciones que debían cumplirse. El resultado, en su estado canónico verificado, es un grafo con **157 conceptos propios**, organizado en torno a **20 clases** y articulado mediante **21 propiedades de objeto** y **7 propiedades de datos**. El grafo afirma explícitamente **1.772 enunciados** (triples) que, tras aplicar el razonamiento bajo el perfil OWL 2 RL, se expanden hasta **4.786 enunciados** materializados. Esa diferencia entre lo afirmado y lo inferido no es un detalle técnico menor, sino una de las contribuciones metodológicas que quería demostrar, la capacidad del grafo de *generar* conocimiento nuevo a partir de su axiomática. El ejemplo más ilustrativo es la consulta del número de conceptos. Ejecutada sobre el grafo sin inferencia devuelve **cero** resultados, porque la pertenencia a la clase de concepto se deduce y no se declara directamente; ejecutada con inferencia activada devuelve **157** conceptos, que son exactamente los 157 conceptos propios. Las **30 entidades de Wikidata** enlazadas no se cuentan aquí, porque se vinculan mediante `skos:exactMatch` (no `owl:sameAs`), y `skos:exactMatch` no propaga el tipo `pyedu:Concepto` hacia la entidad externa, de modo que esas 30 entidades no se infieren como conceptos del grafo. Ese salto de 0 a 157 es la prueba operativa de que la inferencia funciona y aporta valor real al sistema de recuperación.

El grafo incorpora además construcciones de modelado relativamente sofisticadas cuya inclusión respondió a necesidades concretas del dominio. Para representar evaluaciones que vinculan a la vez un estudiante, una actividad, un concepto y un resultado (relaciones intrínsecamente multidimensionales que no caben en un triple binario) se recurrió a **relaciones N-arias**, materializadas en la clase de evaluación de actividad con **10 instancias** de ejemplo. Para registrar la **procedencia** de las afirmaciones —de dónde proviene cada relación y con qué grado de respaldo— se empleó **reificación**, que permite tratar un enunciado como un objeto sobre el que a su vez se predica. El esquema se organizó en dos espacios de nombres deliberadamente separados, `pyedu:` para el esquema (las clases y propiedades) y `pyr:` para las instancias (los conceptos y datos concretos), siguiendo la recomendación habitual de desacoplar terminología y aserción. La interoperabilidad con la web de datos abierta se garantizó mediante los citados **30 enlaces `skos:exactMatch` a Wikidata** y **19 relaciones `skos:broader`** que tejen la jerarquía conceptual. La elección de `skos:exactMatch` en lugar de `owl:sameAs` fue deliberada, porque `owl:sameAs` obligaría al razonador OWL 2 RL a fusionar lógicamente el concepto propio con la entidad de Wikidata como un único individuo y a propagar tipos y propiedades de forma indebida, mientras que `skos:exactMatch` enlaza ambas entidades como equivalentes sin imponer esa identidad de individuos, una decisión que refleja madurez en el uso de datos enlazados.

La tabla siguiente consolida en un único lugar la composición cuantitativa del grafo canónico verificado y el salto que la inferencia introduce, según se ha descrito en los párrafos anteriores:

Componente del grafo canónico	Recuento
Conceptos propios	157
Clases del esquema	20
Propiedades de objeto	21
Propiedades de datos	7
Enunciados afirmados (triples)	1.772
Enunciados materializados (OWL 2 RL)	4.786
Enlaces `skos:exactMatch` a Wikidata	30
Relaciones `skos:broader`	19
Instancias de evaluación de actividad (relación N-aria)	10
Conceptos devueltos por la consulta sin inferencia	0
Conceptos devueltos por la consulta con inferencia	157

La validación de esta fase no se dejó al juicio subjetivo. Se redactó un conjunto de formas SHACL que codifican las restricciones de integridad que el grafo debe satisfacer, y su ejecución sobre el grafo canónico arroja un veredicto **CONFORME, con cero violaciones**. Para descartar que esa conformidad fuese fruto de unas formas vacuas o demasiado laxas, se construyó deliberadamente un ejemplo *inválido* que viola las restricciones, y el validador lo detectó correctamente señalando **6 violaciones**. Esta prueba de falsabilidad de la propia validación —comprobar que el validador rechaza lo que debe rechazar, y no solo que acepta lo que debe aceptar— me parece metodológicamente importante, porque una batería de tests que nunca falla no demuestra nada. Toda la fase produjo como entregable principal una serialización en **Turtle** del grafo, formato elegido por su legibilidad humana y su idoneidad para el control de versiones textual.

11.3 Fase II: diseño e implementación de la arquitectura RAG

La segunda fase trasladó el foco del conocimiento estático a su explotación dinámica mediante una arquitectura de **generación aumentada por recuperación** (RAG), siguiendo el paradigma introducido por Lewis et al. (2020) y enriquecido, en su variante sobre grafos, por las propuestas recientes de *GraphRAG* (Edge et al., 2024). El razonamiento metodológico de fondo es directo. Un modelo de lenguaje generalista, por capaz que sea, carece de un anclaje fiable en la estructura conceptual específica de un currículo de programación. Tiende a producir explicaciones plausibles pero infundadas —el conocido fenómeno de la generación de texto sin compromiso con la verdad que Bender et al. (2021) caracterizaron con el término de los «loros estocásticos»—. Inyectar en el contexto del modelo un subgrafo recuperado del EKG, pertinente al código concreto del estudiante, debería disciplinar su salida y dotarla de fundamento y trazabilidad. Contrastar esa expectativa es el objeto de la Fase III; construir el mecanismo que la hace posible fue el de la Fase II.

El flujo implementado encadena varios componentes. Un **analizador de código** procesa la entrega del estudiante mediante el árbol de sintaxis abstracta (la biblioteca `ast` de Python) y métricas de complejidad (la biblioteca `radon`), extrayendo las estructuras y rasgos que después servirán para localizar los conceptos implicados. A partir de esa representación, un **módulo de recuperación** combina dos estrategias complementarias: una búsqueda por **similitud semántica** sobre representaciones vectoriales (*embeddings*) de los conceptos del grafo, que identifica los nodos semánticamente más próximos al código analizado, y una **expansión mediante consultas SPARQL** que, partiendo de esos nodos semilla, recorre el grafo para recuperar el subgrafo contextual relevante (errores asociados, conceptos prerrequisito, relaciones jerárquicas). Ese subgrafo se traduce a un fragmento de prompt y se entrega, junto con el código del estudiante, a un **modelo de lenguaje local** servido a través de Ollama, que genera el feedback formativo. La ejecución íntegramente local del modelo no es un capricho técnico, sino

una decisión con fundamento ético explícito en el anteproyecto, pues persigue preservar la privacidad y la soberanía de los datos del estudiantado evitando enviar sus entregas a servicios externos.

Sobre esta arquitectura base se definieron **cuatro sistemas** que constituyen las condiciones experimentales de la fase siguiente, y cuya formulación es en sí misma una decisión metodológica. El **Sistema A** es el modelo de lenguaje base (llama3.1:8b) sin augmentación alguna, y funciona como línea de base o grupo de control, pues representa lo que un LLM generalista produce por sí solo. El **Sistema B** es un modelo afinado mediante *fine-tuning* (Qwen2.5-Coder-7B-Instruct adaptado con QLoRA) sin acceso al grafo, que aísla el efecto de especializar los pesos del modelo en la tarea. El **Sistema C** es el modelo base aumentado con GraphRAG, que aísla el efecto de la recuperación sobre el grafo. Y el **Sistema D** es el híbrido que combina *fine tuning* y GraphRAG, concebido para comprobar si ambas intervenciones son aditivas o complementarias. Este diseño factorial (cruzar la presencia o ausencia de *fine tuning* con la presencia o ausencia de recuperación), que la Figura 20 resume, permite descomponer la contribución de cada palanca por separado, en lugar de medir solo un efecto agregado e indescifrable.

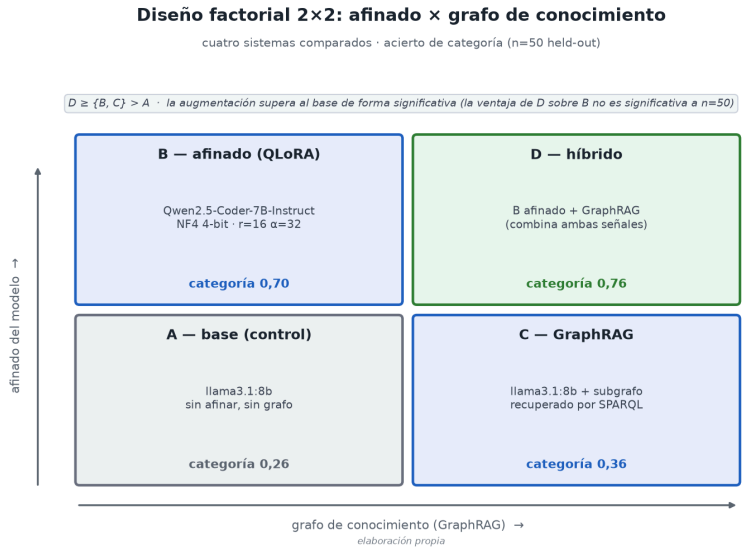

Figura 20. Diseño factorial 2x2 (afinado x grafo) que define A/B/C/D.

El *fine tuning* del Sistema B mereció un tratamiento metodológico cuidadoso por su propensión al sobreajuste, un riesgo agudo cuando el dataset es pequeño. Se empleó **QLoRA** (Dettmers et al., 2023), cuantización de 4 bits en formato nf4 combinada con adaptadores de bajo rango **LoRA** (Hu et al., 2021) de rango 16 y factor alpha 32, entrenados sobre una RTX 5090. El conjunto de entrenamiento se generó **sintéticamente mediante plantillas**, decisión que comento más adelante por sus implicaciones de validez. Una primera ejecución sin regularización exhibió el patrón inequívoco del sobreajuste, ya que la pérdida sobre el conjunto reservado (*held-out*) empeoró época tras época (1,297, después 1,380 y finalmente 1,410), señal de que el modelo memorizaba en lugar de generalizar. La mitigación consistió en introducir regularización (NEFTune con alpha 5 y un *dropout* de 0,1 en los adaptadores LoRA), tras lo cual la pérdida *held-out* descendió a 1,051 y 1,028, con una precisión de token sobre el conjunto reservado de 0,842. Documento estos números, incluido el fracaso inicial, porque el camino hasta una configuración sana forma parte del conocimiento producido; mostrar solo la ejecución corregida y callar la defectuosa escamotearía parte de ese aprendizaje.

El cuarto componente de esta fase fue el **módulo de explicabilidad**, sin el cual la trazabilidad postulada en las hipótesis quedaría en mera promesa. Se implementó una interfaz web con FastAPI que, mediante Cytoscape.js, **visualiza el subgrafo recuperado** con codificación por color y muestra **marcadores de procedencia** que conectan cada afirmación del feedback con su origen en el grafo. Esta visualización no es un adorno, sino el instrumento que hace auditable la cadena de razonamiento del sistema y

que materializa el compromiso con la transparencia que tanto la literatura sobre *learning analytics* responsable (Gašević et al., 2022) como el propio anteproyecto reclaman.

11.4 Fase III: validación experimental

La tercera fase sometió los sistemas a una **evaluación comparativa controlada**, diseñada para responder a la hipótesis principal —que la integración EKG+LLM vía RAG mejora la calidad pedagógica del feedback frente al LLM sin augmentación— y a sus cuatro sub-hipótesis sobre precisión diagnóstica, alucinaciones, utilidad formativa y trazabilidad. El diseño es de tipo **A/B comparativo** sobre un conjunto de casos de prueba, evaluados según una rúbrica de cinco dimensiones inspirada en los principios del feedback eficaz de Hattie y Timperley (2007).

La pieza metodológica más sensible de esta fase, y la que más cuidado exigió, es la **separación estricta entre entrenamiento y evaluación**. Dado que el dataset se generó por plantillas, existía un riesgo real de *fuga de datos*, esto es, que el modelo afinado fuese evaluado sobre casos estructuralmente idénticos a los que vio durante el entrenamiento, lo que inflaría artificialmente su rendimiento. Para conjurar ese riesgo, la evaluación se restringió a **esqueletos reservados** (*held-out*) que no participaron en el *fine tuning* —tal como ilustra la Figura 21—, de modo que ningún sistema fuese examinado sobre material que hubiera memorizado. Esta salvaguarda anti-fuga es, a mi juicio, lo que separa una medición creíble de una ilusión estadística, y por ello acepté la contrapartida de que el conjunto evaluable resultara **pequeño**, un total de **50 casos held-out** en la comparación A/B/C/D que constituye el resultado principal reportado en esta memoria.

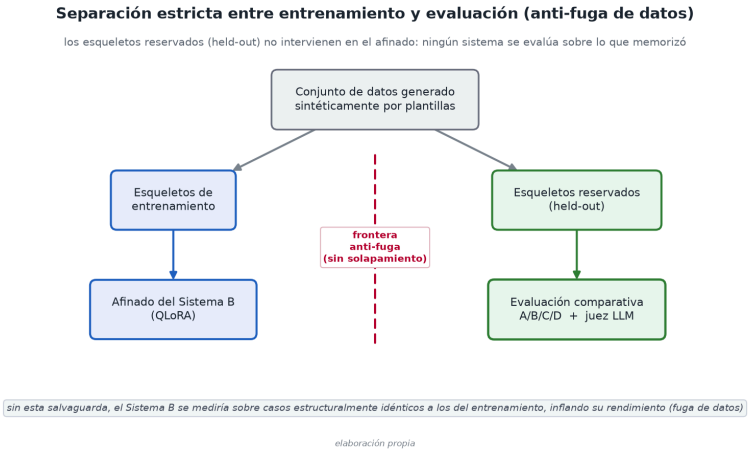

Figura 21. Separación estricta entre entrenamiento y evaluación: los esqueletos reservados (held-out) no intervienen en el afinado, lo que evita la fuga de datos y mantiene creíble la medición.

La evaluación se realizó de forma **automática**, combinando métricas objetivas con un **juez LLM** (el modelo qwen2.5:32b) encargado de puntuar las dimensiones cualitativas de la rúbrica. Sobre esos 50 casos, los resultados muestran un panorama matizado y, sobre todo, **complementario** entre los sistemas, ahora sintetizado por el híbrido. La tabla siguiente recoge las siete dimensiones de la rúbrica para los cuatro sistemas sobre los 50 casos reservados; las dos primeras son aciertos expresados como proporción y las dimensiones cualitativas del juez confirman el mismo patrón que las objetivas.

Dimensión de la rúbrica	Sistema A (base)	Sistema B (QLoRA)	Sistema C (GraphRAG)	Sistema D (híbrido)
Acierto de categoría	0,26	0,70	0,36	0,76
Acierto de concepto	0,18	0,50	0,48	0,54
Identificación del problema (1-5)	2,04	3,80	2,44	4,04
Trazabilidad (1-5)	1,72	3,00	2,92	3,16
Calidad divulgativa	3,52	3,38	3,20	3,66
Calidad técnica	3,02	3,44	2,44	3,62
Calidad de la sugerencia	≈2,9	≈2,9	≈2,9	≈2,9

Estos números indican que el *fine tuning* (B) sobresale en clasificar el tipo de error y en la explicación técnica, mientras que la recuperación sobre el grafo (C) aporta su valor en el acierto de concepto y en la trazabilidad; ambos superan a la línea de base A, pero en dimensiones distintas. Esa complementariedad (ninguno domina en todo) es lo que motivaba y justificaba el Sistema D híbrido, y la **evaluación ampliada A/B/C/D con n=50** ya se ha completado para contrastarlo, con el resultado de que el híbrido **gana o empata en las siete dimensiones de la rúbrica (y es el mejor en seis de ellas)**. Confirma y sintetiza así, en un único sistema, la complementariedad que B y C exhibían por separado, con un ordenamiento global D > {B, C} > A. Insisto, no obstante, en un punto innegociable, y es que **ninguno de los targets numéricos del anteproyecto se da por cumplido**. Ni siquiera el mejor sistema, el híbrido D, alcanza el 85 % de acierto de categoría (se queda en 0,76), y tampoco se da por satisfecho el 5 % de alucinaciones, el 4,0 de utilidad ni el 100 % de trazabilidad; se reporta lo que el experimento midió, que es más modesto y, en su modestia, informativo. A ello se suma que toda la evaluación es automática, sin panel humano, lo que queda como trabajo futuro.

11.5 Decisiones metodológicas y desviaciones respecto al anteproyecto

Conviene declarar las divergencias entre lo planificado y lo ejecutado, y razonarlas en lugar de silenciarlas. La primera y más visible afecta a la **pila tecnológica de la recuperación**. El anteproyecto contemplaba el modelo de *embeddings* all-MiniLM-L6-v2, el índice vectorial FAISS y la base de datos de grafos Neo4j. La implementación real emplea, en cambio, **nomic-embed-text** como modelo de *embeddings* y **rdflib junto con GraphDB** (Ontotext, s.f.) como capa de persistencia y razonamiento del grafo. El cambio de modelo de *embeddings* respondió a su mejor integración en el ecosistema local servido por Ollama y a su calidad sobre texto técnico; el cambio de Neo4j a un *triplestore* RDF nativo como GraphDB fue, en realidad, una **alineación metodológica más profunda** con los objetivos del trabajo, dado que Neo4j es una base de grafos de propiedades, mientras que el corazón de esta investigación es un grafo RDF con axiomática OWL e inferencia bajo el perfil OWL 2 RL, terreno en el que un *triplestore* con razonador integrado y soporte SPARQL/SHACL nativo es la herramienta idónea, y no una elección de conveniencia. FAISS, por su parte, dejó de ser necesario al integrarse la búsqueda vectorial de manera más directa en el flujo. Considero estas sustituciones mejoras informadas surgidas del contacto con el problema real, no concesiones forzadas.

La segunda desviación, de orden epistemológico, concierne a la **naturaleza de los datos**. El anteproyecto preveía trabajar sobre entregas reales (del orden de 300 *submissions*); la evaluación efectivamente realizada se ha hecho sobre **datos sintéticos** generados por plantillas. Esta decisión tuvo

motivaciones prácticas[14] pero acarrea una limitación de **validez ecológica** que no oculto, porque un sistema que funciona sobre código sintético bien estructurado no está garantizado que funcione igual sobre la variedad caótica del código real de un aula. La generación sintética me permitió, eso sí, controlar rigurosamente la separación train/test y disponer de etiquetas fiables, sin las cuales ninguna métrica de precisión diagnóstica sería interpretable.

La tercera desviación, y quizá la más importante de declarar, afecta al **alcance de la validación humana**. El anteproyecto proyectaba una validación con un **panel de 3 a 5 docentes**, el cálculo de la concordancia inter-juez mediante el coeficiente de correlación intraclase (ICC) y la evaluación sobre las 300 *submissions* mencionadas. **Nada de esto se ha ejecutado todavía.** La evaluación de esta memoria es enteramente **automática** (métricas objetivas más juez LLM) sobre datos sintéticos. El panel docente, el ICC inter-juez y la evaluación a gran escala constituyen **trabajo futuro** explícitamente reconocido, y no resultados de este trabajo. Me niego a presentar el juez LLM como un sustituto equivalente del panel humano, porque no lo es. Un juez LLM ofrece consistencia y escala, pero comparte sesgos con los sistemas que evalúa y no captura el juicio pedagógico situado de un docente. Que la validación humana quede pendiente es una limitación de fondo del estado actual del trabajo, y así lo declaro sin atenuantes.

11.6 Reproducibilidad

Un trabajo que no puede reproducirse no es, en sentido estricto, ciencia; la reproducibilidad se ha tratado como un requisito de primer orden y no como una cortesía final. Todos los artefactos del proyecto se mantienen bajo **control de versiones con git**, lo que conserva la trazabilidad de cada cambio en el esquema, los scripts y la configuración experimental. El grafo de conocimiento se distribuye serializado en **Turtle**, formato textual y legible que se versiona limpiamente y que cualquier *triplestore* conforme al estándar puede cargar. Así, el sustrato semántico resulta inspeccionable y reutilizable. La validación SHACL se conserva como un conjunto de formas ejecutables que cualquiera puede correr para comprobar la conformidad del grafo, y las consultas SPARQL que ilustran la inferencia están igualmente disponibles para su reejecución.

En el plano del software, las dependencias quedan fijadas en un fichero de **requisitos** (`requirements`) que ancla las versiones de las bibliotecas empleadas (`ast` y `radon` para el análisis, las utilidades de *embeddings* y recuperación, el cliente de Ollama, rdflib) de modo que el entorno de ejecución pueda reconstruirse de forma determinista. Los procesos no triviales —la construcción y materialización del grafo, el *fine tuning* QLoRA, el flujo de evaluación A/B/C— están encapsulados en **scripts** parametrizados antes que en pasos manuales irrepetibles, lo que elimina el error humano de la cadena y documenta de hecho el procedimiento. Allí donde interviene aleatoriedad (la generación sintética de datos, la partición train/test, la inicialización del entrenamiento) se fijan **semillas** (*seeds*) que hacen las ejecuciones repetibles y los resultados verificables. Por último, la propia memoria se publica como un sitio **MyST** que genera salidas en web, PDF y Word desde una única fuente versionada, de manera que la documentación del trabajo comparte la disciplina de reproducibilidad del trabajo mismo. El conjunto de estas medidas no garantiza que cualquier reejecución reproduzca cifra a cifra los resultados —la variabilidad inherente a los modelos de lenguaje y el reducido tamaño muestral lo impiden—, pero sí asegura que el procedimiento es transparente, auditable y susceptible de ser repetido, criticado y mejorado, que es lo que cabe exigir a un trabajo de investigación.

11.6.1 Reproducibilidad y materiales

A efectos de reproducibilidad dejo consignados en un único lugar los materiales y los parámetros que fijan el experimento. Los modelos empleados, todos servidos localmente con Ollama, fueron los siguientes. Para el Sistema A (base) y el Sistema C (base aumentado con GraphRAG) se usó el modelo `llama3.1:8b`; para el Sistema B (afinado) y el Sistema D (híbrido que combina *fine tuning* y GraphRAG) se usó `Qwen2.5-Coder-7B-Instruct` adaptado mediante QLoRA v3 (cuantización NF4 de

[14] disponibilidad, control sobre las etiquetas de verdad, ausencia de problemas de privacidad con datos de estudiantes reales

4 bits con adaptadores LoRA de rango 16 y factor alpha 32); como juez automático de las dimensiones cualitativas se empleó el modelo `qwen2.5:32b`; y como modelo de *embeddings* para la recuperación semántica, `nomic-embed-text`. El *fine tuning* se ejecutó sobre una GPU RTX 5090.

El análisis estadístico de los resultados —intervalos de confianza obtenidos por remuestreo bootstrap, pruebas de Friedman y contrastes de Wilcoxon con corrección de Holm— se fija con la semilla 11, de modo que el cómputo de las cifras reportadas es determinista y re-derivable a partir de los ficheros de evaluación, sin necesidad de reejecutar los modelos. El grafo se distribuye en Turtle, las formas SHACL y las consultas SPARQL son ejecutables, y las dependencias quedan ancladas en el fichero de requisitos, conforme a lo descrito en los párrafos anteriores. Esta consolidación de semilla, identificadores de modelo y hardware permite que cualquier lector reconstruya el procedimiento y vuelva a derivar las métricas declaradas.

12 Construcción del grafo de conocimiento educativo (O1)

Este capítulo describe el primer objetivo específico del trabajo, la construcción del grafo de conocimiento educativo (en adelante EKG, por *educational knowledge graph*) que constituye el sustrato semántico sobre el que se asienta toda la arquitectura posterior. Si los capítulos del marco teórico justificaron por qué un grafo de conocimiento es una representación adecuada para el conocimiento de un dominio formativo, este capítulo aterriza esa justificación en un artefacto concreto, verificable y ejecutable. Hablo de una ontología en RDF 1.1 serializada en Turtle, con su TBox (el esquema de clases y propiedades) y su ABox (las instancias del dominio de la programación en Python), diseñada bajo el perfil OWL 2 RL para que su explotación combine expresividad razonable y tractabilidad computacional. El resultado es un grafo canónico de 157 conceptos propios y 1772 enunciados afirmados que, tras el cierre deductivo OWL 2 RL, asciende a 4786 enunciados, y que se ha enlazado con la nube de datos abiertos enlazados a través de 30 alineamientos `skos:exactMatch` con Wikidata.

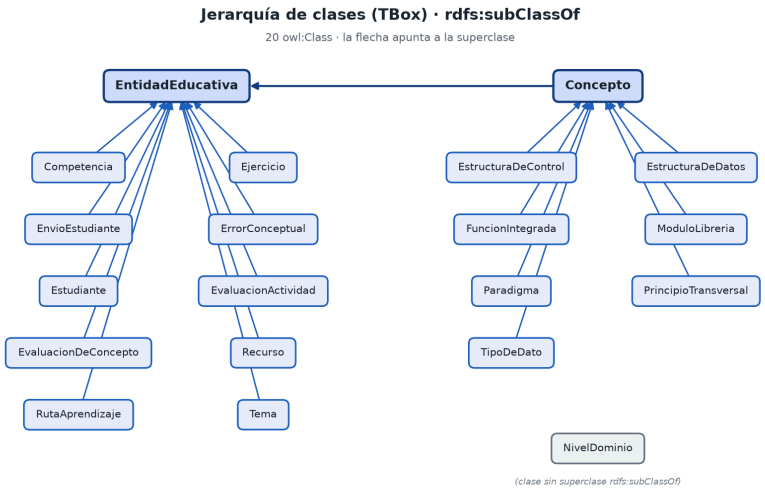

Figura 22. Jerarquía de clases del EKG (TBox), con la relación rdfs:subClassOf. *Fuente: elaboración propia a partir del grafo canónico `ekg-python-150.ttl`.*

Quiero situar el alcance desde el principio. El anteproyecto fijó como meta para O1 un grafo de al menos 150 conceptos y al menos 400 relaciones. Esa meta se ha satisfecho —157 conceptos superan el umbral, y el conjunto de relaciones afirmadas e inferidas lo rebasa con holgura—, pero el cumplimiento de un objetivo de cobertura no debe confundirse con la validación de las hipótesis pedagógicas del trabajo, que se contrastan más adelante y de forma mucho más prudente. El EKG es una pieza de infraestructura, y su corrección formal (que el grafo sea consistente, que valide contra sus restricciones de forma, que infiera lo que debe inferir) es condición necesaria pero no suficiente para que el sistema completo genere retroalimentación útil. Mantengo esa distinción a lo largo del capítulo.

12.1 Del modelo IEEE LOM del anteproyecto a una ontología de dominio

El anteproyecto planteaba la construcción del EKG apoyándose, entre otros referentes, en el estándar IEEE LOM (*Learning Object Metadata*) como vocabulario de metadatos para describir los recursos y objetos de aprendizaje del dominio. Durante la fase de diseño se constató, sin embargo, que LOM resuelve bien un problema que no es del todo el del trabajo. LOM es un esquema de metadatos *sobre* objetos de aprendizaje, pues describe un recurso (su título, su autor, su nivel educativo, su granularidad, sus requisitos técnicos) de cara a su catalogación y descubrimiento. Lo que el trabajo necesita, en cambio, es modelar la *estructura interna del conocimiento del dominio* —qué conceptos existen en la programación en Python, cómo se ordenan por prerrequisito, qué contrastes pedagógicos los enfrentan, qué errores conceptuales típicos aparecen al aprenderlos— de manera que esa estructura pueda recuperarse y razonarse para anclar la generación de un LLM. LOM no ofrece primitivas para expresar que la

recursión es prerrequisito del *backtracking*, ni para reificar la procedencia de la afirmación de que un error concreto recae sobre un concepto concreto.

Por ello, la decisión de diseño ha sido construir una ontología de dominio propia, ligera, y reutilizar para los aspectos de metadatos los vocabularios estándar de la web semántica (Dublin Core, FOAF, SKOS, schema.org) en lugar de LOM. Esto no contradice el anteproyecto sino que lo *extiende*, ya que el espíritu de LOM (describir entidades educativas con metadatos interoperables) se conserva, pero se materializa con un conjunto de vocabularios que se integran de forma nativa en RDF y que son los que se enseñan y manejan en la asignatura de Web Semántica del máster. En concreto, los recursos del grafo se tipan como `schema:LearningResource`, que es el heredero de facto de la noción de objeto de aprendizaje de LOM en el ecosistema de datos enlazados, y se describen con `dcterms:title`, `dcterms:creator` y `schema:url`. Dejo constancia explícita de esta desviación respecto al anteproyecto porque es relevante para la trazabilidad metodológica. El lector que esperase encontrar elementos LOM hallará en su lugar una ontología de dominio acompañada de metadatos en vocabularios estándar, una elección que considero mejor fundamentada para los objetivos de recuperación y razonamiento del TFM, y coherente con la práctica recomendada de reutilizar vocabularios consolidados antes que reinventarlos (Hogan et al., 2021).

12.2 Decisiones de modelado: dos espacios de nombres, esquema e instancias

Una primera decisión, sencilla pero con consecuencias prácticas notables, ha sido separar el esquema de las instancias en dos espacios de nombres distintos. El prefijo `pyedu:` (de *Python educational*, con IRI base `https://w3id.org/ekg-python/schema#`) agrupa el vocabulario, esto es, las clases y las propiedades que definen la TBox. El prefijo `pyr:` (de *Python resource*, con IRI base `https://w3id.org/ekg-python/id/`) agrupa los individuos, a saber, los 157 conceptos, los temas, los ejercicios, los errores, los estudiantes sintéticos y las evaluaciones que pueblan la ABox. Esta separación tiene una virtud directa para la fase de recuperación, ya que permite filtrar trivialmente las consultas SPARQL por tipo de nodo (esquema frente a dato) y facilita que la TBox pueda evolucionar o publicarse de forma independiente de los datos. Los IRI se han construido bajo un dominio `github.io` que será el de publicación del propio sitio del proyecto, para que las entidades sean dereferenciables en el futuro, conforme a un principio básico de los datos enlazados.

La identificación de los individuos se ha hecho mediante IRI legibles y estables derivados de un identificador conceptual en *snake_case* (`pyr:bucle_for`, `pyr:comprension_listas`, `pyr:busqueda_binaria`) en lugar de identificadores opacos numéricos. Esta elección sacrifica algo de neutralidad lingüística a cambio de una legibilidad que ha resultado muy valiosa durante la depuración y la escritura de consultas, y que no compromete la multilingüidad porque las etiquetas humanas no van en el IRI sino en propiedades `rdfs:label` separadas, como se detalla más abajo. La estabilidad del identificador es importante. Estos IRI son las claves que la arquitectura RAG empleará para anclar fragmentos de texto del LLM a nodos del grafo, un identificador inestable rompería la trazabilidad que es uno de los objetivos centrales del trabajo.

12.3 Diseño ontológico: la jerarquía de 20 clases

El esquema define veinte clases organizadas en torno a dos raíces conceptuales, según se aprecia en la Figura 22. La clase más general es `pyedu:EntidadEducativa`, pensada como superclase de todo aquello que es objeto de interés didáctico en el grafo, sea o no un concepto de programación. De ella derivan, mediante `rdfs:subClassOf`, las entidades que no son conceptos propiamente dichos pero sí participan en el dominio formativo, a saber, `pyedu:Tema` (las grandes áreas en que se organiza el currículo, como fundamentos, control, funciones, POO, estructuras, algoritmos, módulos, concurrencia o *testing*), `pyedu:Ejercicio`, `pyedu:Recurso`, `pyedu:Competencia`, `pyedu:Estudiante`, `pyedu:EnvioEstudiante`, `pyedu:ErrorConceptual`, `pyedu:RutaAprendizaje`, `pyedu:EvaluacionActividad` y `pyedu:EvaluacionDeConcepto`. Estas dos últimas son clases de relación n-aria reificada, sobre las que volveré.

La segunda raíz es `pyedu:Concepto`, que modela las unidades de conocimiento del dominio y que se declara a la vez subclase de `pyedu:EntidadEducativa` y de `skos:Concept`. Esta doble adscripción es deliberada y central en el diseño, porque al ser `skos:Concept` cada concepto del grafo hereda toda la maquinaria de SKOS para organizar vocabularios (`skos:broader`, `skos:prefLabel`, etc.), lo que conecta el EKG con una práctica estándar y bien comprendida de modelado de esquemas conceptuales (W3C, 2009). Bajo `pyedu:Concepto` se despliega una subcategorización temática del conocimiento de programación en seis clases hijas que reflejan distintas naturalezas epistemológicas de los conceptos, a saber, `pyedu:EstructuraDeControl` (condicionales, bucles, comprensiones), `pyedu:EstructuraDeDatos` (listas, diccionarios, árboles, tablas hash), `pyedu:TipoDeDato`, `pyedu:FuncionIntegrada`, `pyedu:ModuloLibreria` (módulos de la biblioteca estándar como `collections` o `asyncio`), `pyedu:Paradigma` (composición, herencia, gestores de contexto, *dunder methods*) y `pyedu:PrincipioTransversal` (recursión, *backtracking*, *args/kwargs*, conceptos que atraviesan varias áreas). Esta subcategorización no es ociosa. Como se verá en el capítulo de validación experimental, una de las dimensiones que el sistema diagnostica es la *categoría* del error o del concepto implicado, y disponer de una tipología explícita en el grafo permite evaluar el acierto de categoría de forma objetiva.

Completan las veinte clases `pyedu:NivelDominio`, que se modela como subclase de `skos:Concept` y cuyas instancias (`pyr:basico`, `pyr:intermedio`, `pyr:avanzado`) funcionan como una pequeña escala ordinal de dificultad referenciada desde los conceptos. El número de clases (veinte) no es un objetivo en sí mismo. Es el resultado de buscar el equilibrio entre granularidad suficiente para distinguir lo que pedagógicamente importa y parsimonia suficiente para que el esquema siga siendo manejable, poblable a mano y razonable bajo OWL 2 RL. Cada clase lleva su `rdfs:label` bilingüe y un `rdfs:comment` que documenta su intención, con lo que la TBox resulta autoexplicativa para un lector humano que la abra en Protégé o en un editor de texto.

12.4 Las 21 propiedades de objeto: prerrequisitos, contrastes y jerarquía de propiedades

El corazón relacional del grafo lo constituyen las veintiuna propiedades de objeto, que son las que tejen la red de conexiones entre entidades. Su diseño ha sido la parte del modelado que más reflexión ha exigido, porque es donde la semántica de OWL puede aportar valor real (mediante la inferencia) o, mal usada, generar consecuencias indeseadas.

La propiedad pedagógicamente más importante es `pyedu:esPrerrequisitoDe`, que expresa la dependencia de aprendizaje entre conceptos, de manera que, si A es prerrequisito de B, no se puede comprender B sin dominar antes A. Se ha declarado como `owl:TransitiveProperty`, una decisión que tiene una consecuencia muy concreta y deseable. Si la asignación es prerrequisito del bucle `for`, y el bucle `for` es prerrequisito de la búsqueda lineal, el razonador infiere automáticamente que la asignación es prerrequisito (indirecto) de la búsqueda lineal. Gracias a la transitividad, el grafo solo necesita afirmar los prerrequisitos *directos* (los arcos mínimos), y el cierre deductivo reconstruye toda la malla de dependencias transitivas, que es justo lo que un planificador de rutas de aprendizaje necesita consultar. Esta es una de las demostraciones más claras del valor de razonar sobre el grafo en lugar de almacenar todas las relaciones explícitamente. La propiedad gemela `pyedu:requierePrerrequisito` modela la misma dependencia en sentido inverso desde la perspectiva del concepto que requiere, y también se declara transitiva por la misma razón.

La segunda propiedad de diseño cuidado es `pyedu:contrastaCon`, que conecta dos conceptos que conviene enseñar enfrentados porque su confusión es fuente habitual de errores, como búsqueda lineal frente a búsqueda binaria, copia superficial frente a copia profunda o composición frente a herencia. Esta propiedad se ha declarado `owl:SymmetricProperty`, porque el contraste es intrínsecamente bidireccional, y así, si A contrasta con B, B contrasta con A. La simetría permite afirmar el contraste una

sola vez y que el razonador complete el arco recíproco, lo que evita inconsistencias por olvido (afirmar el contraste en un sentido pero no en el otro) y reduce el mantenimiento del grafo.

La tercera decisión de calado en las propiedades de objeto es el uso de una jerarquía de propiedades mediante `rdfs:subPropertyOf`. Se ha introducido una superpropiedad genérica `pyedu:relacionadoConceptualmenteCon` de la que penden, como subpropiedades, `pyedu:esPrerrequisitoDe`, `pyedu:contrastaCon`, `pyedu:generaliza` y `pyedu:seAplicaA`. La semántica de `rdfs:subPropertyOf` garantiza que toda afirmación con una subpropiedad implica la afirmación con la superpropiedad, y así, si A es prerrequisito de B, entonces A está conceptualmente relacionado con B. Esto da a la fase de recuperación un *mango* uniforme y muy útil, ya que una consulta puede preguntar por todos los conceptos «conceptualmente relacionados» con uno dado y obtener, de golpe, los vinculados por prerrequisito, por contraste, por generalización o por aplicación, sin tener que enumerar cada propiedad específica. Es un patrón de modelado que paga dividendos en la expansión de subgrafos del módulo RAG. La Figura 23 muestra esta superpropiedad con sus subpropiedades y las características OWL declaradas en cada una.

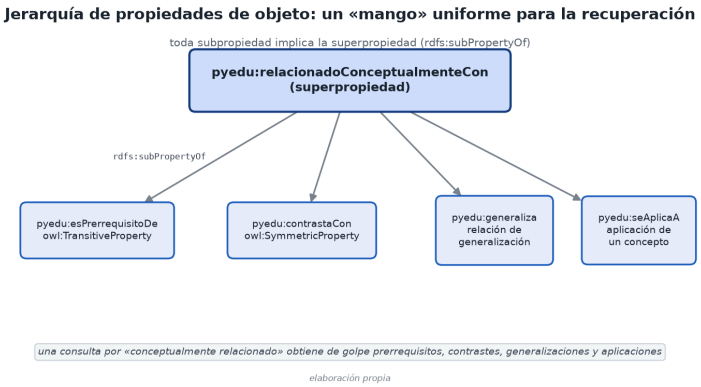

Figura 23. Jerarquía de propiedades de objeto bajo `relacionadoConceptualmenteCon`. *Fuente: elaboración propia.*

El resto de propiedades de objeto modela la conectividad esperable del dominio. La propiedad `pyedu:perteneceATema` adscribe cada concepto a un tema; `pyedu:cubreConcepto` relaciona un ejercicio con los conceptos que practica; `pyedu:evaluaCompetencia` lo conecta con competencias; `pyedu:errorSobreConcepto` indica sobre qué concepto recae un error conceptual; `pyedu:manifiestaError` conecta un envío de estudiante con el error que evidencia; `pyedu:resuelveEjercicio` y `pyedu:enviadoPor` cierran la trazabilidad de los envíos; `pyedu:dominaConcepto` registra qué domina un estudiante; `pyedu:ilustraConcepto` vincula recursos a conceptos; `pyedu:aNivelDominio` asocia un concepto a su nivel; y `pyedu:especializa` y `pyedu:generaliza` expresan relaciones de generalización entre conceptos. A ellas se suman las propiedades que articulan las clases n-arias de evaluación (`pyedu:evaluaA`, `pyedu:porEstudiante`, `pyedu:enEjercicio`, `pyedu:sobreConcepto`), descritas en la sección de relaciones n-arias.

Quiero detenerme en una cautela metodológica sobre los dominios y rangos. OWL no interpreta `rdfs:domain` y `rdfs:range` como restricciones de integridad (como haría una base de datos) sino como axiomas de los que se *infiere* el tipo de los individuos. Declarar que `rdfs:domain` de `pyedu:perteneceATema` es `pyedu:Concepto` no impide afirmar que algo que no es un concepto pertenezca a un tema; lo que hace es *inferir* que ese algo es un concepto. Esta diferencia es sutil pero peligrosa si se modela con mentalidad de esquema relacional. Un dominio o rango demasiado restrictivo puede provocar inferencias de tipo no deseadas o, peor, contribuir a una inconsistencia. Por ello los dominios y rangos del esquema se han declarado de forma deliberadamente *conservadora*, en el nivel de generalidad justo —por ejemplo, el dominio de `pyedu:tieneDificultad` es `pyedu:EntidadEducativa` y no `pyedu:Concepto`, porque también los ejercicios tienen dificultad—, y la verdadera función de

restricción de integridad[15] se ha delegado en SHACL, que es la herramienta diseñada para validar la forma del grafo sin contaminar la semántica de razonamiento. La separación de responsabilidades (OWL para inferir, SHACL para validar) es uno de los aprendizaces de diseño más nítidos del trabajo y se desarrolla en el capítulo dedicado a la validación.

12.5 Las 7 propiedades de datos: literales tipados con XSD y etiquetas multilingües

Frente a las propiedades de objeto, que enlazan recursos entre sí, las siete propiedades de datos conectan recursos con valores literales. Su diseño ha cuidado dos aspectos, a saber, el tipado con la jerarquía de tipos de XML Schema (XSD) y la multilingüidad de las etiquetas. Las propiedades de datos son `pyedu:tieneDificultad` (un `xsd:integer` en la escala 1–5), `pyedu:tieneEnunciado` (el enunciado de un ejercicio, tipado como `rdf:langString` para admitir variantes lingüísticas), `pyedu:tieneEjemploCodigo` (un `xsd:string` con un fragmento de código ilustrativo), `pyedu:enFecha` (un `xsd:date`), `pyedu:obtuvoNota` y `pyedu:conPeso` (sendos `xsd:decimal`) y `pyedu:numeroDeLineas` (un `xsd:integer` que caracteriza un envío). El tipado XSD no es decorativo, pues permite que SHACL valide que una dificultad sea efectivamente un entero entre 1 y 5, que SPARQL pueda ordenar y filtrar por fecha o por nota con la semántica correcta, y que el grafo intercambie estos valores sin ambigüedad entre serializaciones.

La multilingüidad merece comentario aparte. Toda entidad relevante del grafo lleva una o más etiquetas `rdfs:label` con etiqueta de idioma, típicamente una en español (`@es`) y otra en inglés (`@en`). Así, el concepto de la asignación se etiqueta `«Assignment»@en` y `«Asignación»@es`, y la clase abstracta `«Abstract base class»@en` y `«Clase abstracta (ABC)»@es`. Esta doble etiqueta cumple tres funciones. Primero, hace el grafo legible para una audiencia bilingüe y conecta el vocabulario técnico inglés (que el estudiante encontrará en la documentación y los mensajes de error de Python) con su denominación en español (la lengua de instrucción). Segundo, prepara el terreno para el enlazado con Wikidata, cuyas entidades son multilingües por naturaleza, con lo que el alineamiento `skos:exactMatch` resulta más verificable cuando las etiquetas coinciden. Tercero, da al generador de prompts del módulo RAG material textual en ambas lenguas para anclar la explicación. El uso de `rdf:langString` como rango de `pyedu:tieneEnunciado` y de `rdfs:label` con etiquetas de idioma es la forma canónica de gestionar literales multilingües en RDF 1.1 (W3C, 2014a), y se ha aplicado de manera sistemática.

12.6 Reutilización de vocabularios: SKOS, Dublin Core, FOAF y schema.org

Un principio rector del diseño ha sido reutilizar vocabularios consolidados siempre que existiera uno adecuado, en lugar de proliferar términos propios, siguiendo la recomendación de no reinventar lo ya estandarizado (Hogan et al., 2021). El vocabulario propio `pyedu:` se reserva para lo que es genuinamente específico del dominio educativo de la programación; todo lo demás se toma prestado.

De **SKOS** se reutiliza, además de la adscripción de `pyedu:Concepto` a `skos:Concept`, la propiedad `skos:broader` para expresar relaciones de generalidad entre conceptos dentro del esquema conceptual. Hay 19 enlaces `skos:broader` en el grafo, que conectan, por ejemplo, las comprensiones de diccionarios y de conjuntos con la comprensión de listas (más general), o los distintos métodos específicos con la noción genérica de método, o las distintas notaciones de complejidad con la notación O. La elección de `skos:broader` para esta jerarquía de generalidad (distinta de la jerarquía de subclases de la TBox y distinta de la cadena de prerrequisitos) refleja una decisión consciente, porque la generalidad conceptual entre individuos es información de organización del vocabulario, que es para lo que SKOS está pensado, y no una relación de subsunción de clases ni una dependencia de aprendizaje. Mantener estas tres jerarquías separadas (subclase, prerrequisito, *broader*) evita la confusión semántica habitual de colapsarlas todas en una sola relación de «padre». Y, de forma coherente con esta misma filosofía de usar SKOS para enlazar sin imponer identidad lógica, los alineamientos externos a Wikidata se han expresado con

[15] lo que sí debe estar presente, con qué cardinalidad y de qué tipo

`skos:exactMatch` en lugar de `owl:sameAs`, como se justifica en la sección de enlazado. La Figura 24 contrasta estos tres ejes de organización del conocimiento que el diseño mantiene deliberadamente separados.

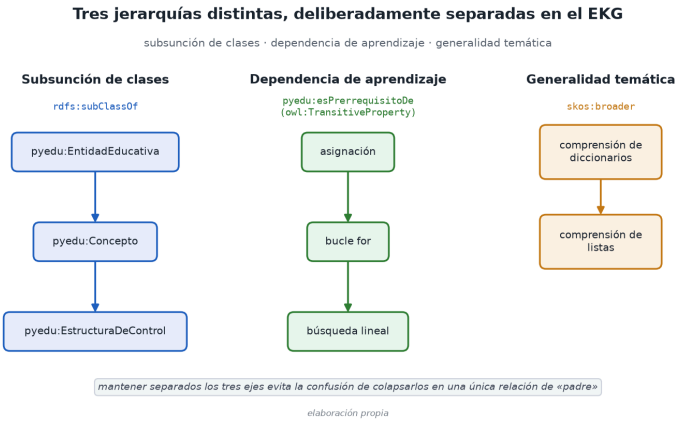

Figura 24. Tres jerarquías distintas (subclase, prerrequisito, broader), separadas a propósito. *Fuente: elaboración propia.*

De **Dublin Core** (`dcterms:`) se reutilizan las propiedades de metadatos y procedencia, a saber, `dcterms:title`, `dcterms:creator`, `dcterms:contributor`, `dcterms:description`, `dcterms:license`, `dcterms:created` y, de forma destacada para la trazabilidad, `dcterms:source`, que ancla la procedencia de afirmaciones a referencias bibliográficas. La propia ontología, como recurso, se autodescribe con Dublin Core indicando su autor, su director como contribuidor, su licencia (CC BY 4.0) y su descripción. De **FOAF** (`foaf:`) se reutiliza `foaf:Person` y `foaf:name` para modelar los estudiantes sintéticos (Alicia, Borja), que se tipan a la vez como `foaf:Person` y `pyedu:Estudiante`; esta doble adscripción aprovecha el vocabulario estándar de personas sin renunciar a la subclase específica del dominio. De **schema.org** (`schema:`) se reutiliza `schema:LearningResource` para tipar los recursos didácticos y `schema:url` para enlazarlos a su localización en la web (por ejemplo, el tutorial oficial de estructuras de datos de Python). Como se anticipó, esta reutilización de schema.org para los recursos es la materialización concreta de la intención que el anteproyecto canalizaba a través de LOM.

12.7 Relaciones n-arias y reificación con procedencia

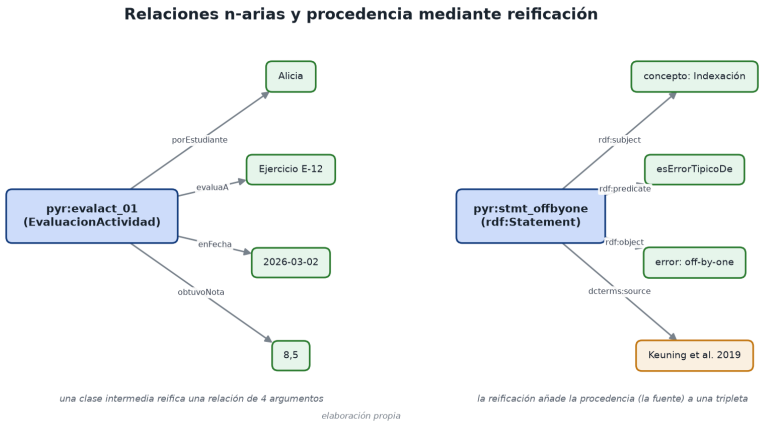

Figura 25. Relaciones n-arias y procedencia mediante reificación.

No toda la información del dominio se deja modelar cómodamente como un arco binario sujeto–predicado–objeto. Dos situaciones del trabajo exigen ir más allá del triple simple, y se han resuelto con los dos patrones canónicos que ofrece RDF, según ilustra la Figura 25.

El primero es el de las **relaciones n-arias**. La evaluación de una actividad no es una relación binaria, pues implica simultáneamente a un estudiante, a un ejercicio, a una fecha y a una nota; afirmar por separado «el estudiante hizo el ejercicio», «la nota es 8.5» y «la fecha es tal» perdería el vínculo entre los cuatro elementos. La solución estándar es reificar la relación n-aria como una clase intermedia cuyos individuos representan el evento de evaluación. Para ello se ha definido la clase `pyedu:EvaluacionActividad`, de la que hay 10 instancias (`pyr:evalact_01` a `pyr:evalact_10`), cada una de las cuales conecta, mediante las propiedades `pyedu:porEstudiante`, `pyedu:evaluaA`, `pyedu:enFecha` y `pyedu:obtuvoNota`, los cuatro componentes de un acto de evaluación concreto. Así, `pyr:evalact_01` afirma que la estudiante Alicia obtuvo un 8.5 en el ejercicio de la suma de una lista el 2 de marzo de 2026, y mantiene todos los componentes solidariamente ligados a un mismo nodo. La clase hermana `pyedu:EvaluacionDeConcepto` aplica el mismo patrón para reificar la evaluación del dominio de un concepto concreto dentro de un ejercicio, con un peso asociado. Este patrón de relación n-aria es el recomendado por el W3C cuando una relación involucra más de dos participantes o requiere atributos propios.

El segundo patrón es la **reificación con procedencia** mediante `rdf:Statement`. Aquí el problema no es la aridad sino la necesidad de hacer afirmaciones *sobre otras afirmaciones*, en concreto de registrar de dónde procede un enunciado. El grafo afirma, por ejemplo, que el error conceptual «off-by-one» recae sobre el concepto de indexación. Pero esa afirmación tiene un origen, ya que se sustenta en la literatura sobre errores típicos de programación. Para no perder esa procedencia se reifica el triple completo como un individuo de tipo `rdf:Statement` (`pyr:stmt_offbyone`) cuyos `rdf:subject`, `rdf:predicate` y `rdf:object` reconstruyen el triple original (el error, la propiedad `pyedu:errorSobreConcepto`, el concepto de indexación) y al que se le adjunta `dcterms:source` apuntando a la referencia bibliográfica correspondiente (Keuning et al., 2019). De este modo el grafo afirma un hecho y, además, documenta su fuente, lo que constituye la base material de la *trazabilidad* que persigo en este trabajo. Cuando el sistema explique a un estudiante que cometió un off-by-one, podrá remontar la afirmación hasta su procedencia documental. La reificación es notoriamente verbosa (cuadruplica el número de triples necesarios para registrar un hecho con su procedencia) y por eso se ha usado con parsimonia, solo allí donde la procedencia añade valor pedagógico real; pero su presencia demuestra que la arquitectura puede transportar marcadores de procedencia desde el grafo hasta el feedback, que es lo que el módulo de explicabilidad explota.

Así queda ese enunciado reificado en el grafo, donde `dcterms:source` ancla su procedencia a la referencia bibliográfica:

pyr:stmt_offbyone a rdf:Statement ;

rdfs:label «Procedencia (reificación): off-by-one sobre indexación según Keuning et al.»@es ;

rdf:subject pyr:err_off_by_one ;

rdf:predicate pyedu:errorSobreConcepto ;

rdf:object pyr:indexacion ;

dcterms:source pyr:ref_keuning2019 .

12.8 Población del grafo: 157 conceptos y la métrica de relaciones

Con el esquema fijado, la fase de población consistió en poblar la ABox con los 157 conceptos del dominio de la programación en Python, distribuidos por las seis subcategorías de concepto y adscritos a los temas del currículo, junto con sus temas, ejercicios, errores conceptuales, recursos, competencias, estudiantes sintéticos y evaluaciones. La población se realizó de forma semiautomática. A partir de un repertorio estructurado de conceptos y sus atributos (categoría, tema, nivel, dificultad, prerrequisitos directos, contrastes y etiquetas bilingües), un script construyó el grafo RDF con `rdflib` (RDFLib Team, s.f.), lo que garantizó la consistencia sintáctica y permitió regenerar el artefacto de forma reproducible ante

cualquier corrección del repertorio. El resultado canónico es un grafo de 1772 enunciados afirmados, que tras el cierre deductivo OWL 2 RL alcanza los 4786 enunciados.[16]

Sobre la métrica de «relaciones» del anteproyecto debo hacer una precisión. El umbral fijado para O1 era de al menos 400 relaciones, y se cumple holgadamente, pero el número exacto depende de qué se cuente como relación. Si se cuentan solo los arcos de prerrequisito y contraste afirmados directamente, la cifra es del orden de varios centenares; si se añaden las adscripciones a tema, nivel y categoría, y muy especialmente si se incorporan las relaciones *inferidas* por transitividad y por la jerarquía de subpropiedades tras el cierre OWL 2 RL, el recuento crece sustancialmente.[17] He preferido reportar las cifras de enunciados afirmados e inferidos, que son inequívocas y verificables ejecutando el cierre, antes que un número único de «relaciones» que requeriría fijar arbitrariamente qué propiedades cuentan. En cualquier lectura razonable, la cobertura relacional del grafo supera el objetivo del anteproyecto.

12.9 Enlazado con la nube de datos abiertos: 30 alineamientos a Wikidata

Para que el EKG no sea una isla sino un nodo de la web de datos, los conceptos que tienen un correlato claro en una entidad de conocimiento general se han alineado con Wikidata mediante 30 enlaces `skos:exactMatch`. Así, el árbol como estructura de datos se alinea con `wd:Q223655`, la búsqueda lineal con `wd:Q787903`, la tabla hash con `wd:Q207440`, y así sucesivamente. La elección de la propiedad es deliberada y la justifico. La propiedad `owl:sameAs` afirmaría identidad lógica fuerte (que ambos IRI denotan el mismo individuo) y, bajo el cierre OWL 2 RL, fusionaría el concepto propio con la entidad de Wikidata propagando propiedades en ambos sentidos, con lo que el razonador trataría a la entidad de Wikidata como si fuera el concepto educativo del grafo (y a la inversa) e importaría además afirmaciones externas. Eso es semánticamente incorrecto, porque el concepto didáctico `pyr:` y la entidad enciclopédica `wd:` son recursos distintos que se *corresponden*, no individuos idénticos. Por ello se ha optado por `skos:exactMatch`, que establece una equivalencia de alineamiento entre esquemas conceptuales sin imponer la fusión lógica de individuos del perfil OWL-RL. Es un enlace fuerte y verificable que no contamina el razonamiento ni arrastra el tipado del recurso enlazado, y refleja una madurez deliberada en el uso de datos enlazados. El alineamiento se ha hecho con cautela y solo donde la correspondencia es defendible, de manera que no todos los 157 conceptos se han enlazado, sino solo aquellos cuya correspondencia es inequívoca; conceptos muy específicos de la enseñanza de Python o de granularidad puramente didáctica se han dejado sin alinear antes que forzar una correspondencia dudosa. El capítulo dedicado al enlazado discute en detalle el procedimiento y la consulta federada que verifica los alineamientos contra el *endpoint* de Wikidata. Aquí basta señalar el efecto inferencial. Una consulta que pregunta por todos los individuos de tipo `pyedu:Concepto` devuelve 0 filas si se ejecuta sin razonamiento (porque ningún individuo declara explícitamente ese tipo, sino solo sus subclases), y devuelve 157 filas con inferencia activada —exactamente los 157 conceptos propios, traídos por el cierre a través de la jerarquía de clases—. Las 30 entidades de Wikidata enlazadas con `skos:exactMatch` no engrosan esa cuenta, y esto es lo deseable, ya que, a diferencia de `owl:sameAs`, `skos:exactMatch` no propaga el tipo `pyedu:Concepto` a la entidad de Wikidata, por lo que el razonador no la infiere como concepto propio. Ese contraste de 0 a 157 es la prueba más tangible de que el grafo no es una mera lista de triples sino una estructura sobre la que el razonamiento añade conocimiento.

12.10 Verificación de la construcción: SHACL y serialización

La construcción del grafo se cierra con su verificación de forma. Se ha definido un conjunto de formas SHACL (`shapes-ekg.ttl`) que codifican las restricciones de integridad que OWL deliberadamente no impone, a saber, que todo concepto tenga al menos una `rdfs:label` y pertenezca al menos a un tema; que su dificultad, si existe, sea un entero entre 1 y 5; que todo enlace `skos:exactMatch` apunte a un IRI con el patrón de una entidad de Wikidata; que todo ejercicio tenga enunciado, cubra al menos un concepto y declare su dificultad; y restricciones análogas para los errores conceptuales y las demás entidades.

[16] el capítulo siguiente analiza en detalle qué triples añade la inferencia y por qué

[17] el salto de 1772 a 4786 enunciados da la magnitud del incremento deductivo

Validado sobre el grafo con inferencia (para que las formas alcancen también los tipos inferidos), el EKG resulta **conforme, con cero violaciones**. Como prueba de que la validación es sensible y no trivialmente satisfactoria, se mantiene un fichero `ejemplo-invalido.ttl` con violaciones deliberadas (un concepto sin tema, una dificultad fuera de rango, un `skos:exactMatch` mal formado), sobre el que SHACL detecta correctamente 6 violaciones. Esta pareja conforme/inválido demuestra que el aparato de validación discrimina y que la conformidad del grafo principal es un resultado significativo y no un artefacto de unas formas vacías. El detalle de las formas, los mensajes de violación y la metodología de validación se desarrollan en el capítulo correspondiente.

El artefacto se serializa en Turtle por su legibilidad, y se exporta sin pérdida a JSON-LD, N-Triples y RDF/XML para garantizar la interoperabilidad con cualquier herramienta del ecosistema. El grafo es cargable en GraphDB (Ontotext, s.f.), en RDFShape (Labra Gayo et al., 2018) y en RDFPlayground, y se ha trabajado con él tanto en `rdflib` desde Python como en GraphDB para la explotación con razonamiento, lo que confirma su portabilidad entre entornos.

12.11 Balance del objetivo O1

Considero el objetivo O1 cumplido en sus términos verificables, ya que existe un EKG de 157 conceptos, con un esquema de 20 clases, 21 propiedades de objeto y 7 propiedades de datos, enlazado a Wikidata con 30 alineamientos, organizado conceptualmente con 19 relaciones `skos:broader`, capaz de relaciones n-arias y de transportar procedencia mediante reificación, conforme a sus restricciones SHACL y enriquecido por el razonamiento OWL 2 RL hasta los 4786 enunciados. El diseño ha priorizado decisiones que pagan dividendos aguas abajo, como la transitividad de los prerrequisitos para reconstruir la malla de dependencias, la simetría de los contrastes para abaratar el mantenimiento, la jerarquía de subpropiedades para ofrecer un mango uniforme a la recuperación, la separación entre OWL para inferir y SHACL para validar, y la reutilización de vocabularios estándar en lugar de IEEE LOM como extensión (no como negación) del planteamiento del anteproyecto.

Mantengo, para terminar, la cautela que abre el capítulo. Disponer de un grafo formalmente correcto y bien razonado es una condición necesaria pero no suficiente para que el sistema completo genere retroalimentación pedagógicamente valiosa. El EKG es la cimentación; que la casa que se construye encima sea habitable (que el feedback sea preciso, poco alucinado, formativo y trazable) es una cuestión empírica que solo la validación experimental, con sus resultados reales y modestos, puede dirimir, y que no debe darse por anticipada por la solidez de los cimientos. Los conceptos, las clases y las propiedades aquí descritos son el vocabulario con el que el resto del trabajo intentará, consciente de sus límites, hablarle de programación a quien aprende.

13 Inferencia y razonamiento

La construcción de un grafo de conocimiento educativo no se agota en la declaración explícita de hechos. Buena parte del valor de una representación basada en estándares de la Web Semántica reside en aquello que no se afirma de manera directa pero puede deducirse de lo afirmado mediante reglas formales de razonamiento. Este capítulo describe el componente de inferencia del *Educational Knowledge Graph* (EKG) construido para este Trabajo de Fin de Máster, las decisiones de diseño que condujeron a adoptar el perfil OWL 2 RL, los motores de razonamiento empleados (`owlrl` sobre `rdflib` y el razonador embebido de GraphDB), y los efectos cuantitativos y cualitativos que la materialización de inferencias produce sobre el grafo. En este proyecto el razonamiento es mucho más que un adorno teórico. Constituye el mecanismo que transforma un conjunto disperso de aserciones en una estructura semánticamente densa, navegable y consultable, sobre la que después apoyé el motor de recuperación de subgrafos de la arquitectura RAG. Sin inferencia, una parte sustancial del conocimiento permanecería latente y, lo que es más grave, invisible para las consultas SPARQL que alimentan al modelo de lenguaje.

13.1 El papel de la inferencia en un grafo educativo

Empiezo por situar qué entiendo por inferencia dentro de las tecnologías del W3C. Un grafo RDF (W3C, 2014a) es un conjunto de tripletas sujeto–predicado–objeto que afirman hechos sobre recursos. Por sí mismo, RDF no atribuye semántica adicional a esos hechos más allá de su mera existencia. Es la combinación con vocabularios de esquema —RDFS (W3C, 2014b) y OWL 2 (W3C, 2012)— la que introduce un cálculo lógico, donde axiomas como la jerarquía de clases (`rdfs:subClassOf`), la jerarquía de propiedades (`rdfs:subPropertyOf`), las restricciones de dominio y rango (`rdfs:domain`, `rdfs:range`), o las características de las propiedades (transitividad, simetría, inversa) habilitan reglas de entailment que derivan tripletas nuevas a partir de las afirmadas. El razonamiento es, así, el proceso de aplicar esas reglas hasta alcanzar un punto fijo en el que no se generan más conclusiones, con lo que materializa el cierre deductivo del grafo (Hogan et al., 2021).[18]

En un dominio educativo, esta capacidad tiene consecuencias muy concretas. Cuando se modela que un concepto de programación como las comprensiones de listas requiere como prerrequisito el manejo de bucles, y que los bucles requieren a su vez el dominio de las estructuras de control básicas, parece natural esperar que el sistema deduzca que las comprensiones de listas dependen, de forma indirecta pero real, de las estructuras de control. Esa deducción no está afirmada explícitamente. Nace de la transitividad de la relación de prerrequisito. Del mismo modo, si se declara que un concepto A es prerrequisito de B, debería poder consultarse el camino inverso (qué conceptos tienen a A como prerrequisito) sin necesidad de duplicar manualmente cada relación en ambos sentidos. La inferencia evita esta redundancia y, sobre todo, garantiza la consistencia. El conocimiento implícito se deriva siempre de forma coherente con lo afirmado, sin posibilidad de que una afirmación directa y su contrapartida inversa entren en contradicción por un descuido del modelador.

Esta distinción entre lo afirmado y lo inferido resultó ser, durante el desarrollo, uno de los aspectos más delicados del proyecto. El grafo canónico contiene 1772 enunciados afirmados, es decir, las tripletas que el proceso de construcción escribe explícitamente en la serialización Turtle. Tras aplicar el razonamiento OWL 2 RL, el grafo materializado asciende a 4786 enunciados. La diferencia, cercana a las 3000 tripletas, no es ruido ni inflación artificial, sino conocimiento legítimamente derivado de los axiomas del esquema `pyedu:` y de las instancias del espacio `pyr:`. Comprender, controlar y verificar esa expansión fue una tarea central. Un grafo que crece de manera incontrolada bajo inferencia suele ser síntoma de un modelado defectuoso, mientras que un grafo cuyo crecimiento es explicable tripleta a tripleta es un grafo sano. La Figura 26 resume el efecto del razonamiento, tanto en el recuento de enunciados como en la consulta de conceptos, que pasa de cero a 157 resultados al activar la inferencia.

[18] Hablo de *punto fijo* en el sentido del encadenamiento hacia adelante (*forward chaining*), esto es, el estado en que reaplicar las reglas no añade ninguna tripleta nueva; ese cierre se calcula una vez y se almacena, de modo que las consultas posteriores no vuelven a pagar el coste del razonamiento.

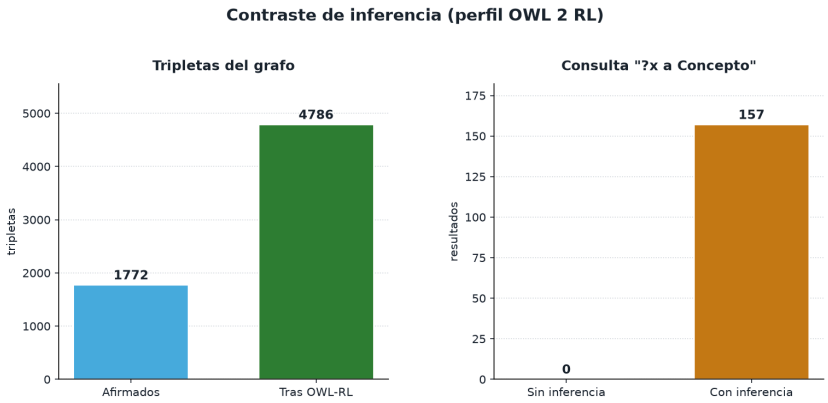

Figura 26. Contraste de inferencia bajo el perfil OWL 2 RL: los enunciados afirmados (1772) se materializan en 4786 tras el razonamiento, y la consulta de conceptos pasa de 0 a 157 resultados. *Fuente: elaboración propia a partir del grafo canónico `ekg-python-150.ttl`.*

13.2 La elección del perfil OWL 2 RL

OWL 2 define varios perfiles (subconjuntos del lenguaje completo) pensados para distintos compromisos entre expresividad y eficiencia computacional (W3C, 2012). El perfil OWL 2 RL (Rule Language) resultó el más adecuado para este proyecto por una combinación de razones técnicas y prácticas. En primer lugar, OWL 2 RL está diseñado específicamente para poder implementarse mediante un sistema de reglas de producción que opera directamente sobre las tripletas, sin necesidad de construir un tableau ni de razonar sobre modelos abstractos. Esto lo hace tratable y, lo que importa para un sistema que ha de responder consultas con latencia razonable, escalable, ya que el cierre puede materializarse hacia adelante (forward chaining) y almacenarse, con lo que las consultas posteriores no pagan el coste del razonamiento. La Figura 27 esquematiza ese proceso, en el que se reaplican las reglas hasta alcanzar el punto fijo y el cierre se almacena de una vez.

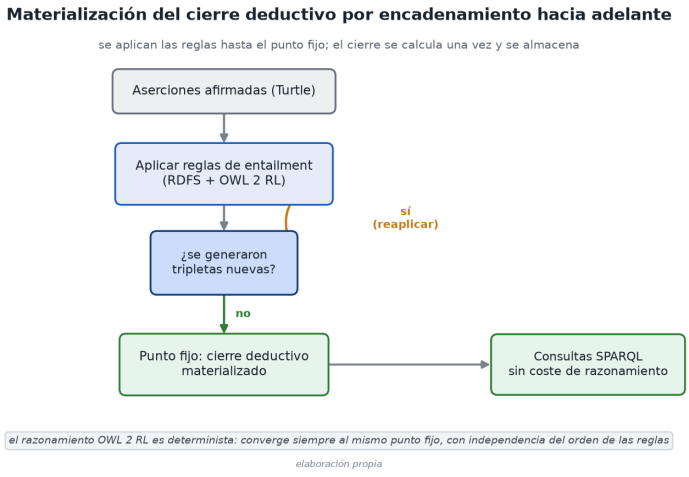

Figura 27. Materialización del cierre por encadenamiento hacia adelante hasta el punto fijo. *Fuente: elaboración propia.*

En segundo lugar, OWL 2 RL cubre con holgura las construcciones que el EKG necesita. El modelo emplea jerarquías de clases entre las 20 clases del esquema, jerarquías de propiedades, dominios y rangos sobre las 21 propiedades de objeto y las 7 propiedades de datos, y (de manera central para el razonamiento educativo) propiedades transitivas, simétricas e inversas. Todas estas características pertenecen al núcleo de OWL 2 RL. Construcciones más exóticas, como las restricciones de cardinalidad existencial que generan individuos anónimos o el razonamiento sobre disyunciones complejas de clases, quedan fuera del perfil, pero ninguna de ellas era necesaria; de hecho, su ausencia es deseable, porque

son las construcciones que comprometen la tratabilidad. Adoptar OWL 2 RL fue, en suma, una decisión de parsimonia, la de elegir el perfil más simple capaz de expresar todo lo que el dominio requería.

En tercer lugar, y no menos importante, OWL 2 RL goza de un soporte excelente en las herramientas empleadas. Tanto la biblioteca `owlrl` para `rdflib` (RDFLib Team, s.f.) como el razonador embebido de GraphDB (Ontotext, s.f.) implementan este perfil de forma directa, lo que permitió disponer de dos materializaciones independientes del mismo grafo y contrastarlas. Esta redundancia no fue casual. Tener dos implementaciones del mismo conjunto de reglas operando sobre las mismas aserciones proporciona una verificación cruzada valiosísima. Si ambos motores convergen al mismo cierre deductivo, gana en confianza la corrección del modelado; si divergen, la divergencia señala con precisión dónde está el problema, ya sea en una característica de propiedad mal declarada o en una diferencia de cobertura entre implementaciones.

13.3 Dos motores de razonamiento, `owlrl` y GraphDB

El flujo de trabajo de inferencia se articuló en torno a dos motores complementarios, cada uno con su papel en el ciclo de desarrollo. La biblioteca `owlrl` opera en memoria sobre un grafo `rdflib` y aplica el conjunto de reglas de entailment de OWL 2 RL (junto con las de RDFS) hasta alcanzar el punto fijo. Su integración en el pipeline de construcción del EKG resultó natural. Una vez ensamblado el grafo canónico de 1772 tripletas afirmadas a partir del esquema y las instancias, una invocación del razonador de `owlrl` materializa el cierre y produce los 4786 enunciados del grafo inferido. Este enfoque tiene la ventaja de ser completamente reproducible y autónomo (no depende de ningún servicio externo) y de integrarse en los mismos scripts de Python que generan el resto de artefactos. Para el desarrollo iterativo, donde cada cambio en el esquema obliga a recalcular el cierre y comprobar sus efectos, esta ligereza fue determinante.

GraphDB, por su parte, aporta un razonador embebido que materializa las inferencias en el momento de la carga y las mantiene actualizadas de manera incremental conforme se insertan o eliminan tripletas. La diferencia operativa es relevante. Mientras que `owlrl` produce un artefacto estático que hay que regenerar por completo ante cualquier cambio, GraphDB gestiona el cierre como parte del estado del repositorio y lo expone de forma transparente a las consultas SPARQL (W3C, 2013). En este proyecto, GraphDB cumplió el papel de entorno de validación y de banco de pruebas para las consultas que después consumiría la arquitectura RAG, y permitió comprobar que el comportamiento inferencial observado con `owlrl` se reproducía en un *triple store* de producción configurado con el mismo perfil OWL 2 RL. La concordancia entre ambos motores sobre el conteo de enunciados y, sobre todo, sobre el resultado de las consultas de conceptos, fue una de las verificaciones que dieron por buena la fase de modelado.

Reconozco que esta dualidad introdujo una pequeña complejidad de configuración. GraphDB ofrece varios *rulesets* predefinidos, y seleccionar el que se corresponde con OWL 2 RL (y no uno más expresivo o más restrictivo) fue un paso que hubo que cuidar para que la comparación con `owlrl` fuese legítima. Una vez alineadas las configuraciones, la equivalencia se mantuvo, lo que refuerza la idea de que el cierre deductivo del EKG no es un artefacto de una herramienta concreta, sino una propiedad del grafo y del perfil de razonamiento elegido.

13.4 El contraste NONE frente a RDFS, de 1772 a 4786 enunciados

El experimento más ilustrativo del impacto de la inferencia consistió en consultar el grafo bajo distintos regímenes de razonamiento y comparar los resultados. Sin razonamiento alguno (régimen NONE), el grafo expone únicamente sus 1772 enunciados afirmados. Activando el razonamiento del perfil (que incluye las reglas de RDFS como subconjunto y las añade las reglas propias de OWL 2 RL), el grafo materializado alcanza los 4786 enunciados. El factor de expansión, próximo a tres, merece un análisis cuidadoso, porque la utilidad de la inferencia no se mide por la cantidad bruta de tripletas que genera, sino por la pertinencia de esas tripletas para las consultas que el sistema necesita resolver.

El caso paradigmático, y el que mejor evidencia por qué la inferencia resulta indispensable en este diseño, es la consulta de conceptos. Una consulta SPARQL que recupera todas las instancias de la clase de concepto del esquema `pyedu:` devuelve, sobre el grafo sin inferencia, cero resultados. Esta cifra puede resultar desconcertante a primera vista, pero su explicación es la que justifica todo el aparato de razonamiento.[19] Los conceptos se afirman en el grafo como instancias de clases más específicas de la jerarquía, o bien su pertenencia a la clase general de concepto se deriva de declaraciones de subclase y de las características de las propiedades que los relacionan, no de una aserción directa de tipo. Sin razonamiento, esas pertenencias permanecen implícitas y la consulta, que busca el tipo general, no encuentra nada que satisfaga su patrón.

Al activar la inferencia, la misma consulta pasa de cero a 157 resultados, que corresponden a los 157 conceptos propios del EKG, materializados ahora como instancias de la clase general gracias a la propagación de tipos por la jerarquía de clases y de propiedades. Me detengo en por qué las 30 entidades de Wikidata enlazadas desde el grafo no engrosan este total. Esos enlaces se establecen mediante `skos:exactMatch`, y no mediante `owl:sameAs`, por una decisión semántica deliberada, ya que `owl:sameAs` iguala lógicamente dos recursos, con lo que el razonador OWL-RL propagaría a la entidad de Wikidata todo lo afirmado o inferido del concepto propio (incluida su pertenencia a la clase de concepto) y fusionaría de facto ambos individuos. Eso es incorrecto, porque la entidad de Wikidata es un recurso externo con identidad propia, no una instancia del esquema `pyedu:`. Al emplear `skos:exactMatch`, el enlace afirma una correspondencia exacta entre los dos recursos sin imponer su fusión lógica, de manera que el tipo no se propaga y las 30 entidades de Wikidata quedan correctamente fuera del recuento de conceptos inferidos. Esta cifra de 157 frente a 0 es, a mi juicio, el indicador más elocuente del valor de la inferencia en el proyecto. Una capacidad de consulta que sin razonamiento es literalmente nula (cero conceptos recuperables) se convierte, con razonamiento, en el acceso completo al inventario conceptual del grafo, al tiempo que sus correspondencias con la Web de datos abiertos enlazados se mantienen explícitas pero semánticamente disciplinadas.

La lectura de este contraste debe hacerse con cuidado metodológico. Que la consulta sin inferencia devuelva cero no es un defecto del grafo, sino una propiedad esperada de un modelado que delega deliberadamente en el razonador la materialización de los tipos generales. Podría haberse optado por afirmar explícitamente cada tipo, lo que habría evitado la dependencia del motor de inferencia, pero esa decisión habría duplicado información, multiplicado las oportunidades de inconsistencia y, en definitiva, traicionado el espíritu de las tecnologías semánticas, que es declarar lo mínimo necesario y dejar que la lógica derive el resto. La cifra de 157 frente a 0 es, así, una validación de la arquitectura semántica adoptada.

13.5 Transitividad de prerrequisitos, inversas y simetría

Más allá del recuento agregado, examino las clases de inferencia que más conocimiento aportan al dominio educativo, porque son ellas las que justifican axioma a axioma el crecimiento del grafo. La transitividad de la relación de prerrequisito es, sin duda, la más significativa desde el punto de vista pedagógico. Al declarar la propiedad de prerrequisito como transitiva en el esquema `pyedu:`, el razonador deriva automáticamente todos los prerrequisitos indirectos a partir de las cadenas de prerrequisitos directos. Si el concepto C requiere B y B requiere A, el grafo materializado contiene la tripleta que afirma que C requiere A, aunque nunca se haya escrito explícitamente; la Figura 28 recoge ese encadenamiento. Sobre un grafo de 157 conceptos con cadenas de dependencia de varios niveles de profundidad, esta clausura transitiva genera un número considerable de tripletas nuevas y, lo que es más valioso, habilita consultas como «¿cuáles son todos los conceptos que un estudiante debe dominar, directa o indirectamente, antes de abordar las comprensiones de listas?» sin necesidad de recorrer manualmente el grafo. Para el motor de recuperación de subgrafos de la arquitectura RAG, disponer

[19] el grafo contiene 157 conceptos propios, ¿cómo es posible que la consulta no devuelva ninguno?

del cierre transitivo de los prerrequisitos significa poder construir, con una sola consulta, el contexto completo de dependencias conceptuales relevante para diagnosticar un error de un estudiante.

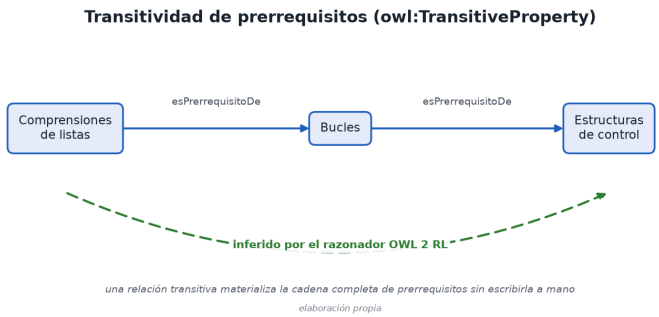

Figura 28. Cierre transitivo de prerrequisitos (owl:TransitiveProperty).

La inferencia de propiedades inversas resuelve un problema complementario. En el modelado, una relación de prerrequisito tiene una dirección natural (A es prerrequisito de B), pero las consultas necesitan a menudo recorrer la relación en sentido contrario —dado A, ¿qué conceptos lo tienen como prerrequisito? Declarar la propiedad inversa permite al razonador materializar automáticamente las tripletas en sentido opuesto, de modo que el grafo queda navegable en ambas direcciones sin que el modelador haya tenido que afirmar cada relación dos veces. Esta economía expresiva atañe tanto a la comodidad como a la integridad. Si las inversas se afirmaran a mano, cualquier omisión rompería la simetría del grafo y dejaría consultas con resultados parciales e impredecibles. Delegar la inversa en el razonador garantiza que la bidireccionalidad sea total y consistente por construcción.

La simetría, por último, se aplica a aquellas relaciones del dominio en las que el vínculo carece de dirección privilegiada, como las relaciones de afinidad o relación entre conceptos en las que afirmar que A se relaciona con B implica necesariamente que B se relaciona con A. Al declarar tales propiedades como simétricas, el razonador completa el par recíproco de cada aserción. Aunque su contribución cuantitativa al crecimiento del grafo es menor que la de la transitividad, la simetría aporta coherencia a la navegación conceptual y evita asimetrías artificiales que confundirían tanto a las consultas SPARQL como a la posterior visualización del subgrafo en el módulo de explicabilidad.

Estas tres clases de razonamiento (transitividad, inversa y simetría) junto con la propagación de tipos por las jerarquías de clases y propiedades y la correspondencia por `skos:exactMatch` con la Web de datos enlazados, explican de forma acumulativa el salto de 1772 a 4786 enunciados. Cada tripleta inferida es trazable a una regla y a un conjunto de premisas afirmadas, lo que permitió auditar el cierre y descartar que el crecimiento respondiera a errores de modelado. Esta trazabilidad de la inferencia es, además, coherente con el compromiso de transparencia que recorre todo el proyecto, porque el grafo no esconde conocimiento mágico, sino que toda conclusión es justificable a partir de hechos y reglas explícitos.

13.6 Reproducibilidad del perfil de razonamiento

Un grafo de conocimiento que se materializa de forma distinta cada vez que se procesa no es un grafo fiable, y mucho menos uno sobre el que pueda construirse un experimento científico replicable. Por ello, la reproducibilidad del perfil de inferencia fue un requisito que se cuidó desde el diseño. El razonamiento OWL 2 RL es, por su propia naturaleza, determinista. Dado un mismo conjunto de aserciones y un mismo conjunto de reglas, el cierre deductivo es único, con independencia del orden en que se apliquen las reglas, porque la materialización hacia adelante converge siempre al mismo punto fijo. Esta propiedad teórica se tradujo en una garantía práctica. La materialización del EKG produce de manera consistente los mismos 4786 enunciados a partir de los mismos 1772 afirmados, y la consulta de conceptos devuelve invariablemente 157 resultados.

La reproducibilidad se reforzó por la vía de fijar y documentar la configuración exacta del razonamiento en ambos motores. En el pipeline basado en `owlrl`, esto significó dejar registrado el régimen de

entailment empleado y versionar tanto el esquema `pyedu:` como las instancias `pyr:` que constituyen las premisas; en GraphDB, supuso fijar el *ruleset* OWL 2 RL como configuración del repositorio, de modo que cualquier carga del mismo grafo canónico regenere idéntico cierre. La concordancia entre las dos materializaciones independientes verifica la corrección, como ya he señalado, y constituye además una prueba de reproducibilidad cruzada, donde dos implementaciones distintas, partiendo de las mismas premisas y el mismo perfil, llegan al mismo resultado. La serialización Turtle del grafo, finalmente, fija el artefacto en un formato textual estable y diffeable, lo que permite versionar el grafo afirmado y comprobar, ante cualquier cambio, qué aserciones se han modificado y cómo repercuten en el cierre.

Esta atención a la reproducibilidad no es un mero formalismo. La fase de validación experimental del Trabajo de Fin de Máster se apoya en el EKG materializado como fuente de contexto para el motor de recuperación; si el grafo variara entre ejecuciones, los resultados del experimento A/B/C dejarían de ser comparables y las conclusiones sobre la aportación del componente GraphRAG perderían su fundamento. Garantizar que el perfil de razonamiento produce siempre el mismo grafo es, en definitiva, una precondición de la integridad de toda la evaluación posterior. Preciso también los límites de lo logrado. La reproducibilidad verificada se refiere al determinismo del cierre deductivo y a la concordancia entre motores sobre el grafo canónico actual, no a una auditoría exhaustiva de equivalencia regla a regla entre `owlrl` y GraphDB sobre casos límite, que excede el alcance de este trabajo y queda apuntada como línea de consolidación futura. Con esa salvedad, el componente de inferencia cumple su función, que es convertir un grafo modesto de aserciones explícitas en una estructura semánticamente rica, consistente y reproducible, sobre la que se sostiene el resto de la arquitectura.

14 Validación de integridad con SHACL

La construcción de un grafo de conocimiento educativo plantea un problema que rara vez se aborda con el rigor que merece. La cuestión es cómo garantizar que las miles de afirmaciones que componen el grafo respetan, de manera demostrable y reproducible, las restricciones estructurales y semánticas que el modelo conceptual presupone. En el caso del EKG canónico desarrollado en este trabajo —157 conceptos propios, 1.772 enunciados afirmados que ascienden a 4.786 tras el cierre deductivo OWL-RL, 20 clases, 21 propiedades de objeto y 7 de datos— la verificación manual resulta sencillamente inviable. Una sola instancia mal tipada, una propiedad obligatoria omitida o un enlace a Wikidata con un identificador malformado pueden propagarse silenciosamente a través de la inferencia y contaminar las consultas SPARQL que alimentan la recuperación de subgrafos en la arquitectura RAG. Por ello, la última etapa de la fase de construcción del grafo (Fase I de la metodología) se dedicó a establecer un mecanismo de validación automática basado en SHACL (*Shapes Constraint Language*), la recomendación del W3C (2017) para la descripción y validación de patrones estructurales sobre grafos RDF. En este capítulo documento el grafo de formas que diseñé, los *constraints* reforzados que incorporé para elevar el nivel de exigencia más allá de las comprobaciones triviales, el resultado de la validación (conforme, con cero violaciones) y el control negativo que confirma la sensibilidad del validador.

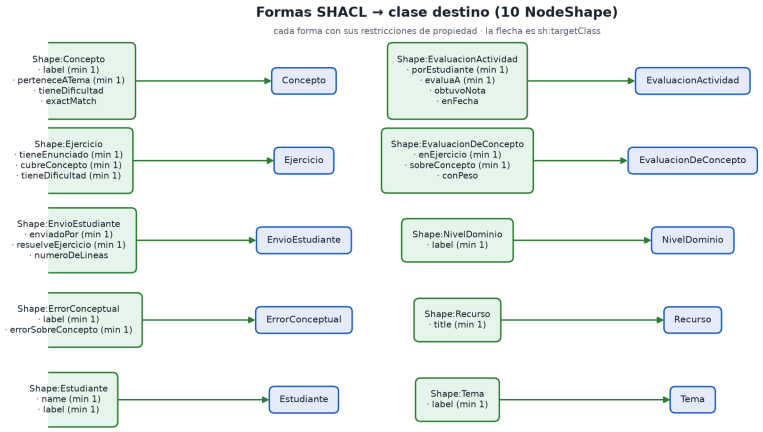

Figura 29. Las 10 formas SHACL (NodeShape) y sus clases destino. *Fuente: elaboración propia a partir de las formas SHACL del proyecto sobre el grafo canónico `ekg-python-150.ttl`.*

14.1 El papel de SHACL en el ciclo de vida del grafo

Quiero situar SHACL en el lugar correcto dentro del ecosistema de tecnologías de la Web Semántica que empleé en este proyecto, porque su función se confunde con frecuencia con la de OWL. Como recuerdan Hogan et al. (2021) en su panorámica sobre grafos de conocimiento, OWL y SHACL responden a filosofías distintas y, en buena medida, complementarias. OWL (W3C, 2012) opera bajo la *hipótesis de mundo abierto* y la *ausencia de nombres únicos*, pues su propósito es inferir nuevo conocimiento a partir del existente, y por ello la falta de una afirmación no se interpreta como falsedad sino como desconocimiento. SHACL, en cambio, adopta una semántica de validación próxima a la *hipótesis de mundo cerrado* y define qué patrones *debe* satisfacer un grafo concreto para considerarse íntegro y señala como violación toda desviación respecto de esos patrones. Aplicar restricciones OWL con la intención de validar conduce a confusiones bien conocidas (una cardinalidad OWL no rechaza datos, sino que infiere igualdades o pertenencias), mientras que SHACL fue concebido para el control de calidad de los datos.

En el flujo de trabajo de este TFM, ambos lenguajes cooperan en una secuencia deliberada. El esquema, definido en el espacio de nombres `pyedu:`, declara mediante OWL las clases, jerarquías y propiedades del dominio; las instancias, en el espacio `pyr:`, pueblan ese esquema con conceptos, actividades, evaluaciones y enlaces externos. Sobre el grafo materializado ejecuto primero el razonador OWL 2 RL,

que expande los 1.772 enunciados afirmados hasta 4.786 mediante el cierre deductivo (propagación de tipos por la jerarquía de clases, herencia de dominios y rangos, transitividad de `skos:broader`, etc.). Y es *sobre ese grafo enriquecido* donde aplico la validación SHACL. Esta ordenación, resumida en la Figura 30, no es accidental. Validar antes de inferir produciría falsos positivos, ya que muchas afirmaciones que un *constraint* exige no están escritas explícitamente sino que se derivan. Por ejemplo, una instancia puede no declarar de forma explícita su pertenencia a una clase si dicha pertenencia se infiere a partir del rango de una propiedad; si validáramos sin inferencia previa, una forma que exige esa pertenencia la marcaría erróneamente como ausente.

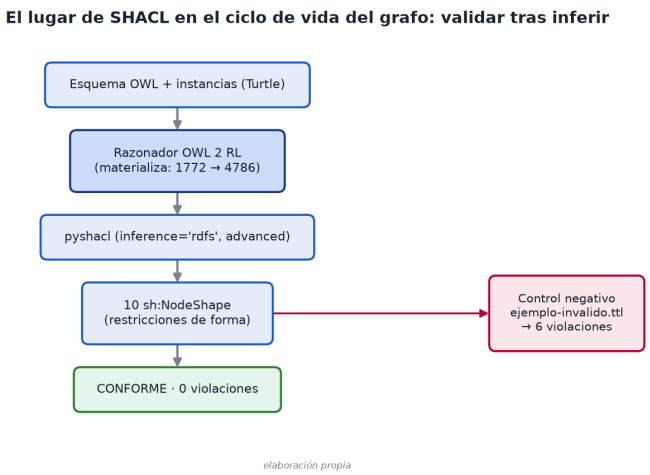

Figura 30. Validar tras inferir: lugar de SHACL en el ciclo de vida.

La herramienta escogida para ejecutar la validación es **pyshacl** (Sommer y Car, s.f.), la implementación de referencia en Python del estándar SHACL, que se integra de manera natural con `rdflib` (RDFLib Team, s.f.), la biblioteca sobre la que se construye toda la capa de grafos del proyecto. Una característica decisiva de pyshacl para nuestro caso es su capacidad de realizar **inferencia RDFS** de forma integrada en el propio proceso de validación. Activando la opción correspondiente (`inference=«rdfs»`), pyshacl expande el grafo de datos con las consecuencias de las declaraciones `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain` y `rdfs:range` (W3C, 2014b) inmediatamente antes de comprobar las formas. Esto refuerza la garantía descrita en el párrafo anterior incluso si el grafo se validara de manera aislada, ya que las pertenencias de clase derivables por subsunción quedan disponibles para las formas, lo que evita los falsos negativos que de otro modo se producirían. En la práctica, el grafo que llega a pyshacl ya ha pasado por OWL-RL, y la inferencia RDFS adicional actúa como una red de seguridad que cierra cualquier laguna entre lo que el razonador OWL materializó y lo que las formas presuponen.

14.2 Diseño del grafo de formas: una NodeShape por clase

El grafo de formas se organiza siguiendo un principio sistemático que facilita su mantenimiento y su trazabilidad respecto del esquema, pues **a cada clase relevante del dominio le corresponde una `sh:NodeShape`** cuyo objetivo (`sh:targetClass`) es esa clase. La Figura 29 recoge las diez formas resultantes y la clase destino de cada una. De este modo, la estructura del grafo de formas refleja como un espejo la estructura del esquema OWL, y cualquier modificación en una clase (añadir una propiedad obligatoria, restringir un rango) tiene un punto único y evidente donde traducirse a una restricción de validación. Cada `NodeShape` agrupa un conjunto de `sh:PropertyShape` que describen, propiedad a propiedad, las condiciones que deben cumplir los valores asociados a las instancias de la clase.

Las formas cubren las clases nucleares del EKG. La forma asociada a la clase de conceptos exige que cada concepto disponga de una etiqueta legible (`rdfs:label`) con cardinalidad mínima, que sus relaciones jerárquicas mediante `skos:broader` apunten a otros conceptos del grafo y no a recursos arbitrarios,

y que los enlaces de identidad `skos:exactMatch` (cuando existan) se dirijan a recursos del espacio de Wikidata. La forma asociada a las actividades de evaluación verifica la presencia de las propiedades descriptivas obligatorias y la coherencia de los rangos de las propiedades de objeto que las vinculan con conceptos y con la clase N-aria `EvaluacionActividad`. Esta última, que materializa una relación N-aria entre una actividad, un concepto evaluado y los metadatos de procedencia (con 10 instancias en el grafo canónico), recibe una forma específica que comprueba que cada nodo de reificación enlaza efectivamente con todos los participantes que la relación requiere; se trata de un punto particularmente sensible, porque las relaciones N-arias reificadas son frágiles ante omisiones.[20]

El diseño *una NodeShape por clase* tiene además una virtud metodológica. Al alinear formas y clases, el grafo de formas se convierte en una *especificación ejecutable* del modelo conceptual, ya que leerlo equivale a leer las invariantes que considero inviolables. Esto resulta valioso en un grafo destinado a alimentar un sistema RAG, donde la calidad de la recuperación depende directamente de que las instancias estén bien formadas. Un concepto sin etiqueta no podría mostrarse en la visualización del subgrafo recuperado (la capa de explicabilidad que construí con FastAPI y Cytoscape.js); un `skos:exactMatch` malformado rompería el enriquecimiento con conocimiento externo de Wikidata. Las formas protegen, por tanto, tanto la integridad abstracta del grafo como la operatividad concreta de toda la arquitectura aguas abajo.

14.3 Constraints reforzados

El valor de una validación SHACL depende por completo de la exigencia de sus restricciones. Un grafo de formas que solo comprobara la existencia de etiquetas aportaría una garantía pobre. Una decisión consciente de diseño fue, por tanto, **reforzar los *constraints*** con tres familias de comprobaciones que elevan sustancialmente el nivel de aseguramiento. La Figura 31 resume estas tres familias, el mecanismo SHACL de cada una y la garantía concreta que protege.

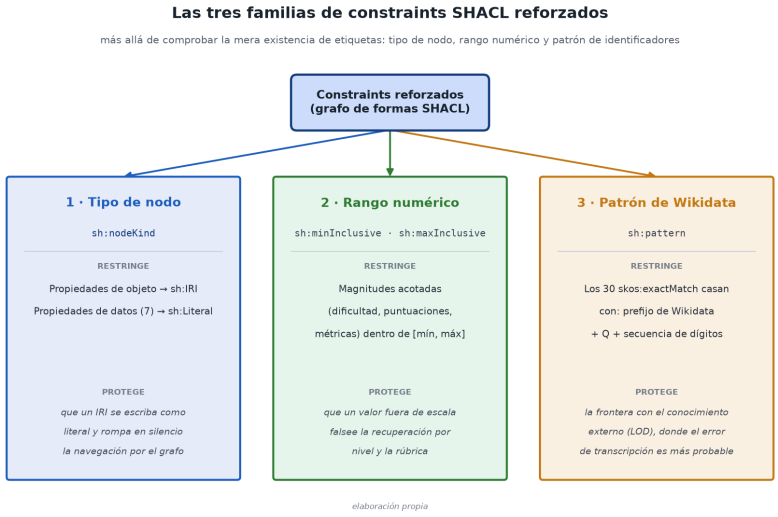

Figura 31. Las tres familias de constraints SHACL reforzados: tipo de nodo, rango numérico y patrón de identificadores. *Fuente: elaboración propia a partir de las formas SHACL del proyecto.*

La primera es el control del **tipo de nodo** mediante `sh:nodeKind`. Esta restricción distingue entre IRIs, literales y nodos en blanco, e impide confusiones que en RDF son sintácticamente legales pero semánticamente erróneas. Las propiedades de objeto —las que vinculan recursos entre sí, como `skos:broader` o las que conectan actividades con conceptos— se restringen a `sh:IRI`, de modo que ningún valor pueda ser accidentalmente un literal. Recíprocamente, las propiedades de datos (las 7 *datatype properties* del esquema) se restringen a `sh:Literal`. Sin esta comprobación, un error tipográfico que escribiera el

[20] un nodo intermedio al que le falta uno de sus participantes rompe la semántica de toda la afirmación sin producir un error sintáctico

identificador de un concepto como una cadena de texto en lugar de como una referencia a un recurso pasaría inadvertido y rompería silenciosamente la navegación por el grafo.

La segunda familia atañe a las **restricciones de rango sobre valores numéricos**, mediante `sh:minInclusive` y `sh:maxInclusive`. Las propiedades de datos que representan magnitudes acotadas (niveles de dificultad, puntuaciones, métricas asociadas a las evaluaciones) quedan sujetas a límites inferior y superior, de manera que un valor fuera del intervalo admisible se rechaza. Esta comprobación es especialmente pertinente en un grafo educativo donde, por ejemplo, un nivel de dificultad debe situarse dentro de una escala definida; un valor espurio falsearía tanto la recuperación adaptada al nivel como la rúbrica de evaluación.

La tercera familia, y posiblemente la más característica de este trabajo, es la **validación del patrón de los identificadores de Wikidata** mediante `sh:pattern`. Los 30 enlaces `skos:exactMatch` que conectan conceptos propios con entidades de Wikidata constituyen el puente del EKG con el conocimiento abierto. La elección de `skos:exactMatch` en lugar de `owl:sameAs` es deliberada, porque `owl:sameAs` llevaría al razonador OWL-RL a fusionar lógicamente el concepto propio con la entidad de Wikidata, lo cual es semánticamente incorrecto[21] `skos:exactMatch` enlaza ambos recursos como equivalentes sin imponer esa fusión, una distinción que refleja madurez en el uso de datos enlazados. Por esta misma razón, la consulta de conceptos sigue devolviendo 157 entidades propias incluso tras activar la inferencia, ya que las 30 entidades de Wikidata, enlazadas con `skos:exactMatch`, **no** se infieren como `pyedu:Concepto` —porque `exactMatch` no propaga el tipo, a diferencia de lo que ocurriría con `owl:sameAs`—. Dada la centralidad de estos enlaces, la forma de conceptos impone que todo valor de `skos:exactMatch` case con una expresión regular que exige el prefijo canónico de la entidad de datos de Wikidata seguido de un identificador de la forma `Q` más una secuencia de dígitos. Un enlace con un dominio incorrecto, con una letra distinta de `Q` o con caracteres no numéricos tras ella se marca como violación. Esta restricción protege la frontera entre el grafo local y el externo, que es donde un error de transcripción resulta más probable y más dañino, pues degrada el enriquecimiento semántico sin manifestarse como un fallo evidente.

14.4 Resultado: conforme, cero violaciones

Tras ejecutar pyshacl sobre el grafo canónico (con OWL-RL aplicado previamente e inferencia RDFS activada durante la propia validación), el informe de validación reporta el grafo como **CONFORME, con cero violaciones**. Es decir, las 4.786 afirmaciones del grafo enriquecido satisfacen sin excepción todas las formas y todos los *constraints* reforzados que he descrito. Cada instancia respeta los tipos de nodo prescritos, todas las magnitudes acotadas caen dentro de sus intervalos, los 30 enlaces a Wikidata cumplen el patrón exigido, las relaciones N-arias reificadas enlazan con todos sus participantes y las propiedades obligatorias están presentes.

Interpreto este resultado con la prudencia que la integridad científica reclama. Una validación conforme **no demuestra que el grafo sea correcto en su contenido** —no certifica que las relaciones jerárquicas entre conceptos sean pedagógicamente acertadas ni que los enlaces a Wikidata apunten a la entidad conceptualmente adecuada—; demuestra únicamente que el grafo es *estructuralmente íntegro* respecto de las invariantes que el modelo declara. La corrección del contenido es una cuestión distinta, que se aborda parcialmente mediante la revisión del esquema y la evaluación experimental de la arquitectura, y que en el fondo depende del juicio experto. SHACL aporta una garantía necesaria pero no suficiente; su valor reside en eliminar de raíz toda una categoría de errores (los estructurales y de tipado) para que la atención pueda concentrarse en los problemas de fondo.

14.5 Control negativo: el ejemplo inválido detecta 6 violaciones

Un resultado conforme, por sí solo, no permite concluir que el validador funcione, pues un grafo de formas mal construido, sin objetivos correctamente asignados o con restricciones que nunca se activan,

[21] tratándolos como un único individuo y propagando entre ambos todos sus tipos y propiedades

también informaría cero violaciones sobre cualquier entrada. Afirmar la integridad del grafo exige, por ello, demostrar de manera complementaria que el validador es **sensible** —que detecta efectivamente las violaciones cuando estas existen—. Esta es una práctica metodológica elemental tomada de la validación de instrumentos, donde todo procedimiento de prueba debe acreditarse con un *control positivo de error* además del caso nominal.

Con ese propósito construí deliberadamente un **ejemplo inválido**, una versión del grafo en la que introduje seis defectos representativos de las categorías que las formas pretenden capturar —un concepto sin etiqueta obligatoria, un `skos:exactMatch` con un identificador de Wikidata que incumple el patrón, una propiedad de objeto cuyo valor es un literal en lugar de un IRI, un valor numérico fuera del intervalo admisible y la omisión de participantes en una relación N-aria—. Ejecutada pyshacl sobre este grafo defectuoso, el informe reporta **6 violaciones**, una por cada defecto inyectado, e identifica en cada caso el nodo afectado, la propiedad implicada y el *constraint* incumplido. La coincidencia exacta entre el número de defectos introducidos y el número de violaciones detectadas confirma que el grafo de formas no produce ni falsos negativos (defectos que pasarían inadvertidos) ni falsos positivos espurios en este escenario controlado.

La conjunción de ambos resultados (cero violaciones sobre el grafo canónico y seis violaciones sobre el control negativo) constituye la evidencia de que la validación es a la vez *correcta* (el grafo bueno pasa) y *sensible* (el grafo defectuoso se rechaza con la granularidad esperada). Sin el control negativo, el resultado conforme sería ambiguo; con él, adquiere fuerza probatoria.

14.6 Reproducibilidad e integración en el flujo

Incorporé la validación como un paso reproducible y automatizable del proceso de construcción del grafo. Al estar implementada sobre pyshacl y `rdflib`, la comprobación puede ejecutarse de forma idéntica en cualquier máquina sin más dependencia que el entorno Python del proyecto, lo que la hace adecuada para integrarse en una rutina de verificación continua. Cada vez que el esquema o las instancias se modifican, basta con regenerar el grafo, aplicar el razonamiento y volver a validar para obtener una garantía actualizada de integridad estructural. La inferencia RDFS integrada en pyshacl asegura que la validación siga siendo coherente aunque el grafo se examine de manera aislada, sin depender de que el cierre OWL-RL se haya materializado externamente. Esta automatización es coherente con la filosofía general del proyecto, que persigue un despliegue local, autocontenido y reproducible, alineado con las consideraciones de soberanía de los datos discutidas en el apartado ético.

En conjunto, la validación SHACL cierra la fase de construcción del EKG con una garantía formal y verificable, pues el grafo de 157 conceptos, sus 30 puentes a Wikidata y sus relaciones N-arias reificadas con procedencia constituyen una base estructuralmente íntegra sobre la que la arquitectura RAG puede operar con confianza. Las formas, alineadas una a una con las clases del esquema y reforzadas con controles de tipo de nodo, rangos numéricos y patrones de identificadores externos, transforman el modelo conceptual en una especificación ejecutable cuyo cumplimiento se demuestra de manera reproducible. El resultado conforme y el control negativo de seis violaciones, leídos conjuntamente, constituyen evidencia de la integridad del grafo sin sobreinterpretar su alcance, porque SHACL certifica la forma, no el acierto del contenido, y es esa modestia en sus pretensiones lo que hace de la validación una pieza fiable del aseguramiento de calidad del trabajo.

15 Explotación con SPARQL

Una vez construido el grafo de conocimiento educativo (EKG), enriquecido mediante razonamiento OWL 2 RL y validado con SHACL, el siguiente paso natural consiste en interrogarlo. De poco serviría disponer de un grafo canónico de 157 conceptos propios y miles de enunciados afirmados e inferidos si no contásemos con un mecanismo expresivo para extraer de él la información pertinente en cada situación. Ese mecanismo es SPARQL (W3C, 2013), el lenguaje de consulta estándar para grafos RDF, que en esta arquitectura desempeña un doble papel. Es, en primer lugar, la herramienta de exploración y auditoría que me ha permitido comprobar la integridad y la riqueza del modelo durante su desarrollo. Y es además, de manera más relevante para los objetivos del trabajo, el motor de recuperación que, integrado en la fase de expansión del subsistema GraphRAG, materializa la conexión entre el grafo simbólico y el modelo de lenguaje. En este capítulo describo cómo he explotado el EKG mediante SPARQL, recorriendo desde las formas de consulta básicas hasta la consulta-caso de uso que produce el subgrafo entregado al LLM, y deteniéndome en algunas decisiones de modelado que el lenguaje hace explícitas, como la diferencia entre recorrer prerrequisitos mediante un *property path* o mediante una propiedad declarada transitiva.

Sitúo este capítulo dentro de la lógica general de la memoria. Los capítulos precedentes han establecido *qué* contiene el grafo y *cómo* se garantiza su corrección; este capítulo aborda *cómo se consume*. Las consultas que presento no son ejercicios ilustrativos desligados del sistema, sino las operaciones reales que ejecuta la arquitectura, ejecutadas sobre el almacén GraphDB (Ontotext, s.f.) mediante el endpoint SPARQL y, durante el desarrollo, también desde `rdflib` en memoria para pruebas rápidas. La distinción importa porque algunas consultas dependen de que el grafo consultado sea el grafo *inferido* (el de 4786 enunciados tras OWL-RL) y no el meramente afirmado (1772). Como se verá, una misma consulta puede devolver cero resultados o ciento cincuenta y siete según se ejecute sobre uno u otro, y esa diferencia es el valor que aporta el razonamiento a la explotación. Esos ciento cincuenta y siete son los conceptos propios del modelo que el razonador clasifica como `pyedu:Concepto`; anticipo que las treinta entidades de Wikidata enlazadas desde el grafo no engrosan esa cifra, pues, como se detalla más adelante, se vinculan mediante `skos:exactMatch` (que no propaga el tipo) y no mediante `owl:sameAs`, por lo que el razonador no las infiere como conceptos propios.

15.1 SELECT y CONSTRUCT: dos formas para dos propósitos

Tres formas de consulta SPARQL y su uso en el sistema

Forma SPARQL	Qué devuelve	Uso en el sistema
`SELECT`	una tabla de filas	inspección y métricas (p. ej. conceptos por tema)
`CONSTRUCT`	un subgrafo RDF nuevo	evidencia para GraphRAG (Turtle / Cytoscape.js)
`UPDATE (INSERT/DELETE)`	modifica el grafo	depuración (09_update_depuracion.rq)
`property path esPrerrequisitoDe+`	la cadena completa de prerrequisitos	alternativa al razonador para recorrer la transitividad

elaboración propia

Figura 32. SELECT / CONSTRUCT / UPDATE y su uso. *Fuente: elaboración propia a partir de las consultas SPARQL del proyecto.*

SPARQL ofrece cuatro formas de consulta (`SELECT`, `CONSTRUCT`, `ASK` y `DESCRIBE`), de las cuales en este trabajo empleo sobre todo las dos primeras, porque responden a necesidades distintas y complementarias que conviene no confundir. La Figura 32 sintetiza estas formas y el papel que cumple cada una. La forma `SELECT` devuelve una tabla de resultados (un conjunto de filas con columnas correspondientes a las variables proyectadas). Es la forma adecuada para la inspección, la depuración y el cálculo de métricas, porque su salida es tabular y se presta a recuentos, agregaciones y verificaciones puntuales. Cuando durante la construcción del EKG necesitaba saber cuántos conceptos propios había definido, qué propiedades de objeto estaban realmente en uso o si una determinada instancia de evaluación quedaba correctamente vinculada a su actividad, recurría a un `SELECT`. La consulta canónica de recuento de conceptos (de la que hablaré más adelante) es un `SELECT (COUNT(…) AS ?n)` cuyo único resultado es un número. Ese número resume de un vistazo el estado del grafo. A estas formas de lectura el entregable añade, por indicación del director, un ejemplo de la cara complementaria de SPARQL

1.1, su capa de actualización `UPDATE` (la consulta `09_update_depuracion.rq`), que reúne las tres operaciones de administración y depuración del grafo —alta de hechos con `INSERT DATA`, corrección con `DELETE … INSERT … WHERE` y retirada con `DELETE WHERE`— para dejar constancia de que el mismo lenguaje que interroga el EKG es también el que lo mantiene; tras cada edición conviene revalidar con SHACL y recalcular las cifras del grafo, pues cambian con cada cambio.

La forma `CONSTRUCT`, en cambio, no devuelve una tabla sino un grafo RDF nuevo, construido a partir de una plantilla de tripletas que se rellena con las soluciones del patrón de la cláusula `WHERE`. Esta diferencia, que en una primera lectura puede parecer un mero detalle de formato de salida, resulta decisiva en un sistema GraphRAG. Lo que el modelo de lenguaje necesita recibir no es una tabla de filas heterogéneas, sino un fragmento coherente del grafo (un *subgrafo*) que pueda serializarse de forma legible y que preserve la estructura relacional original: qué concepto requiere qué prerrequisito, qué error procede de qué fuente, qué enunciado evaluativo conecta una actividad con una competencia. `CONSTRUCT` es, así, la forma natural de extraer subgrafos, porque su resultado es a su vez un grafo que puede volver a serializarse en Turtle, recorrerse, visualizarse en la interfaz web con Cytoscape.js o linealizarse para componer el contexto del *prompt*. En la práctica, mi flujo de trabajo combina ambas. Empleo `SELECT` para localizar los nodos de anclaje relevantes (los conceptos que la similitud de *embeddings* ha señalado como pertinentes para el código del estudiante) y `CONSTRUCT` para extraer, alrededor de esos anclajes, el subgrafo que constituirá la evidencia recuperada. Esta separación de responsabilidades (localización tabular frente a extracción estructural) refleja una distinción conceptual que prefiero mantener nítida.

15.2 Conceptos por tema: agregación y navegación de la jerarquía

Una de las consultas más frecuentes durante la construcción y la exploración del EKG es la que organiza los conceptos por tema, es decir, la que recorre la dimensión temática del grafo para responder a preguntas del tipo «¿qué conceptos pertenecen al tema de las estructuras de control?» o «¿cuántos conceptos hay en cada bloque temático?». Esta operación se apoya en la estructura SKOS del modelo (W3C, 2009), que organiza los conceptos en una jerarquía mediante `skos:broader`, de la que el grafo contiene diecinueve enlaces, y en las propiedades de objeto del esquema `pyedu:` que asocian cada concepto con su tema. Un `SELECT` que agrupe por tema y cuente los conceptos asociados produce una panorámica inmediata de la distribución del conocimiento modelado, como muestra la consulta siguiente.

PREFIX pyedu: <…esquema…>

PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?tema (COUNT(?concepto) AS ?n)

WHERE {

?concepto a pyedu:Concepto ;

pyedu:perteneceATema ?tema .

}

GROUP BY ?tema

ORDER BY DESC(?n)

El interés de este tipo de consulta no es meramente descriptivo. Sirve, en primer lugar, de control de calidad. Si un tema aparece con un número anómalamente bajo de conceptos, o si algún concepto no se agrupa bajo ningún tema, la consulta lo revela de inmediato y señala una posible omisión en el modelado. La navegación temática es, además, la base de varias funcionalidades de la arquitectura, porque la recuperación de subgrafos no se limita a los conceptos directamente señalados por la similitud,

sino que a menudo interesa expandir hacia los conceptos hermanos del mismo tema o hacia el tema padre, para ofrecer al modelo de lenguaje un contexto pedagógicamente más completo. La cláusula `GROUP BY`, combinada con funciones de agregación como `COUNT` —y, cuando interesa el detalle, con `GROUP_CONCAT` para listar las etiquetas de los conceptos de cada tema en una sola fila—, permite pasar con fluidez de la vista agregada a la vista detallada sin cambiar la estructura esencial de la consulta. La jerarquía `skos:broader` añade además una dimensión de generalidad, pues una consulta puede recuperar tanto los conceptos de un tema como los conceptos más generales que los subsumen, lo que resulta útil para graduar el nivel de la explicación que recibirá el estudiante en función de su madurez.

15.3 Errores y procedencia: la reificación al servicio de la trazabilidad

Un aspecto en el que el EKG va más allá de un catálogo de conceptos es el modelado explícito de los errores de programación y, sobre todo, de su procedencia. La hipótesis H1d del anteproyecto exige trazabilidad, y la trazabilidad solo es posible si el grafo registra de dónde procede cada afirmación. Para ello el modelo recurre a la reificación con procedencia. Ciertos enunciados, en particular los que asocian un error típico con un concepto o con una fuente bibliográfica, no se afirman como simples tripletas anónimas, sino que se reifican para poder colgar de ellas metadatos sobre su origen. De este modo, cuando el sistema afirma que un determinado patrón erróneo —por ejemplo, la confusión entre asignación y comparación, o el desbordamiento de índice en el recorrido de una lista— está asociado a un concepto del currículo, esa asociación puede acompañarse de un marcador que indique en qué fuente se documentó dicho error. Así se ancla el conocimiento del grafo en literatura reconocida sobre dificultades de aprendizaje en programación (Watson & Li, 2014; McCracken et al., 2001; Keuning et al., 2019).

Consultar los errores junto con su procedencia es, en consecuencia, una operación que atraviesa la reificación. Una consulta `SELECT` que recupere, para cada error, el concepto al que se asocia y la fuente de la que procede, debe navegar desde el nodo del error hasta el nodo de reificación que describe la relación, y de ahí hasta la propiedad de procedencia, como muestra la consulta siguiente.

PREFIX pyedu: <…esquema…>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX dcterms: <http://purl.org/dc/terms/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?error ?concepto ?fuente

WHERE {

?e a pyedu:ErrorConceptual ;

rdfs:label ?error ;

pyedu:errorSobreConcepto ?c .

?c rdfs:label ?concepto .

FILTER(LANG(?error) = «es»)

FILTER(LANG(?concepto) = «es»)

OPTIONAL {

?stmt rdf:subject ?e ;

rdf:predicate pyedu:errorSobreConcepto ;

rdf:object ?c ;

dcterms:source ?ref .

?ref dcterms:title ?fuente .

}

}

ORDER BY ?error

El bloque `OPTIONAL` es el que atraviesa la reificación, ya que vincula el nodo de enunciado (`?stmt`) que describe la relación error-concepto con la referencia bibliográfica (`?ref`) de la que procede, de modo que la fuente acompaña al error solo cuando está documentada y no lo hace desaparecer cuando falta. El resultado es una tabla en la que cada fila es trazable hasta su origen, lo que constituye el insumo del módulo de explicabilidad. Los marcadores de procedencia que la interfaz web muestra junto al subgrafo recuperado no son una decoración, sino el reflejo directo de estas tripletas de procedencia almacenadas en el grafo. Sobre los límites del mecanismo, advierto que la trazabilidad que el sistema garantiza es la del *subgrafo recuperado* (es decir, la evidencia que se entrega al modelo está perfectamente anclada), lo cual no equivale automáticamente a que el texto finalmente generado por el LLM respete esa procedencia con fidelidad del cien por cien; cerrar esa brecha entre la evidencia trazable y la generación fiel es una de las cuestiones que la evaluación experimental aborda, y los resultados reales obtenidos muestran que el grafo mejora la trazabilidad de manera apreciable[22], aunque sin alcanzar todavía el objetivo aspiracional de trazabilidad total que fijaba el anteproyecto.

15.4 Prerrequisitos transitivos: *property path* frente a propiedad transitiva

Una de las relaciones pedagógicamente más significativas del EKG es la de prerrequisito, el hecho de que el dominio de un concepto presuponga el dominio de otro. Las funciones recursivas presuponen las funciones; las funciones presuponen las estructuras de control y las variables; los diccionarios presuponen las listas y el concepto de clave. Esta relación tiene una propiedad lógica evidente. Es transitiva. Si A es prerrequisito de B y B lo es de C, entonces A es, mediatamente, prerrequisito de C. La cuestión interesante, y que SPARQL hace explícita de forma elegante, es que hay (al menos) dos maneras distintas de obtener el cierre transitivo de los prerrequisitos, y que esas dos maneras responden a filosofías diferentes con consecuencias prácticas distintas.

La primera vía es declarar la propiedad de prerrequisito como `owl:TransitiveProperty` en el esquema y delegar el cálculo del cierre en el razonador OWL 2 RL. Bajo este enfoque, el razonamiento materializa de antemano todas las tripletas de prerrequisito implícitas, de manera que el grafo inferido contiene ya, como hechos afirmados, «A es prerrequisito de B», «B es prerrequisito de C» y también «A es prerrequisito de C». Una consulta `SELECT` que pregunte por los prerrequisitos transitivos de un concepto se reduce entonces a un patrón de tripleta sencillo, porque todo el trabajo lo ha hecho el razonador en la fase de materialización. La ventaja es la simplicidad de la consulta y la coherencia con el resto de la semántica OWL del modelo; el coste es que las tripletas inferidas engrosan el grafo materializado[23] y que el cierre transitivo queda fijado en el momento de la inferencia, por lo que cualquier modificación de los prerrequisitos exige reejecutar la materialización para mantener la coherencia.

La segunda vía es no declarar transitividad en el esquema y, en su lugar, calcular el cierre en tiempo de consulta mediante un *property path*. SPARQL 1.1 permite escribir patrones de camino con operadores de cierre, donde `pyedu:esPrerrequisitoDe+` recorre uno o más pasos de la propiedad, y `pyedu:esPrerrequisitoDe*` recorre cero o más. Una consulta de la forma

SELECT DISTINCT ?prerequisito

[22] puntuaciones de trazabilidad en una escala de 1 a 5 de 1,72 para el LLM base frente a 2,92 para la variante con GraphRAG, y de 3,16 para el sistema híbrido que combina *fine tuning* y GraphRAG

[23] son parte de esos 4786 enunciados que el grafo alcanza tras OWL-RL frente a los 1772 afirmados

WHERE {

?concepto pyedu:esPrerrequisitoDe+ ?prerequisito .

FILTER(?concepto = pyedu:FuncionesRecursivas)

}

obtiene el cierre transitivo sin necesidad de que el razonador haya materializado ninguna tripleta adicional, pues el motor de consulta navega el grafo en profundidad en el momento de evaluar el patrón. La ventaja es la flexibilidad —el cierre se calcula sobre el estado vigente del grafo, sin paso previo de materialización, y no aumenta el tamaño del almacén— y un control más fino, porque el operador `+` excluye el camino de longitud cero (el propio concepto) mientras que `*` lo incluye, distinción que importa al decidir si el resultado debe contener o no el concepto de partida. El coste es que el cómputo recae sobre cada consulta y que la semántica del *property path* no es idéntica a la de la propiedad transitiva en todos los casos límite, porque los *property paths* operan sobre la conectividad del grafo y no participan en el resto del aparato inferencial de OWL.

En el sistema he optado por aprovechar ambas posibilidades de forma deliberada. El razonamiento OWL-RL materializa las relaciones cuya transitividad forma parte de la semántica estable del dominio y que conviene tener disponibles de manera uniforme para todas las consultas; en cambio, para las expansiones exploratorias de la recuperación de subgrafos —donde interesa controlar con precisión la profundidad del recorrido y no inflar el grafo permanente— recurro a *property paths* con operadores de cierre acotados. Esta convivencia ilustra una idea que considero central en el trabajo, la de que la web semántica ofrece más de un mecanismo para una misma necesidad, y que la elección informada entre ellos, lejos de ser una cuestión meramente técnica, expresa un compromiso entre coste de materialización, flexibilidad de consulta y claridad semántica. Hacer explícita esa elección, en vez de tomarla por inercia, es parte del rigor que un trabajo de investigación debe documentar.

15.5 La consulta-caso de uso: el subgrafo para el RAG

Todas las piezas anteriores convergen en la consulta que da sentido a la explotación del grafo dentro de la arquitectura, la que produce el subgrafo de evidencia que se entregará al modelo de lenguaje. Esta es, propiamente, la consulta-caso de uso del capítulo, porque es la que conecta el grafo simbólico con la generación neuronal y la que materializa el componente «Graph» del GraphRAG. Su funcionamiento se entiende mejor si se recuerda el flujo en el que se inserta. Cuando un estudiante envía una solución de programación, el analizador de código construye su árbol de sintaxis abstracta con el módulo `ast` y calcula métricas de complejidad con `radon`; a partir de ese análisis se obtiene una representación textual de las características relevantes del código, que se proyecta al espacio de *embeddings* mediante el modelo `nomic-embed-text`. La similitud entre ese vector y los vectores de los conceptos del EKG identifica un conjunto de conceptos de anclaje, aquellos semánticamente más próximos a lo que el estudiante ha escrito o ha errado. Hasta aquí el proceso es de recuperación densa por similitud. La aportación específica del grafo comienza después, cuando, en lugar de entregar al modelo únicamente esos conceptos aislados, una consulta `CONSTRUCT` los toma como semillas y extrae a su alrededor un subgrafo coherente.

La consulta `CONSTRUCT` que ejecuta esta expansión recibe los identificadores de los conceptos de anclaje y construye un grafo nuevo que incluye, para cada uno de ellos, su etiqueta y descripción, sus prerrequisitos (recorridos hasta una profundidad acotada mediante el *property path* de cierre), los errores típicos asociados junto con su procedencia, el tema al que pertenecen y, cuando procede, su enlace `skos:exactMatch` al recurso correspondiente de Wikidata, enlazado así, y no mediante `owl:sameAs`, para evitar que el perfil OWL-RL fusione lógicamente el concepto propio con la entidad de Wikidata. La plantilla de la cláusula `CONSTRUCT` declara la forma de las tripletas que se desean en el subgrafo resultante, y el patrón `WHERE` las puebla a partir del grafo inferido. El resultado es un fragmento del EKG (típicamente de unas decenas de tripletas, según la profundidad de expansión

configurada) que es a la vez compacto y rico. Es compacto porque se ciñe a lo relevante para el código analizado, y rico porque preserva la estructura relacional que da contexto pedagógico a cada concepto. Esquemáticamente, la consulta adopta la forma que recoge el listado siguiente.

CONSTRUCT {

?c rdfs:label ?etiqueta ;

pyedu:tieneDescripcion ?desc ;

pyedu:perteneceATema ?tema ;

pyedu:esPrerrequisitoDe ?prereq ;

pyedu:presentaError ?error .

?error pyedu:procedeDe ?fuente .

?c skos:exactMatch ?wikidata .

}

WHERE {

VALUES ?c { <…conceptos de anclaje…> }

?c rdfs:label ?etiqueta .

OPTIONAL { ?c pyedu:tieneDescripcion ?desc }

OPTIONAL { ?c pyedu:perteneceATema ?tema }

OPTIONAL { ?c pyedu:esPrerrequisitoDe+ ?prereq }

OPTIONAL { ?c pyedu:presentaError ?error .

OPTIONAL { ?error pyedu:procedeDe ?fuente } }

OPTIONAL { ?c skos:exactMatch ?wikidata }

}

El uso intensivo de `OPTIONAL` no es accidental. Garantiza que la ausencia de un dato concreto (un concepto sin enlace a Wikidata, un error sin fuente documentada) no haga desaparecer al concepto entero del subgrafo, sino que simplemente omita la rama correspondiente. Esta robustez frente a la información parcial es indispensable en un grafo real, donde no todos los conceptos están enriquecidos al mismo nivel de detalle. El cierre transitivo de prerrequisitos mediante el *property path* `+` aporta aquí su flexibilidad característica y permite ajustar cuánta cadena de conocimiento previo se incorpora al contexto sin alterar el grafo permanente. El subgrafo así construido se serializa en Turtle, se linealiza a un formato textual legible y se inserta en la plantilla del *prompt* que el generador compone para el LLM local servido por Ollama (`llama3.1:8b`), con lo que el modelo recibe, junto al código del estudiante, el andamiaje conceptual fundamentado que debe sustentar su retroalimentación.

Quiero cerrar con una nota metodológica coherente con el resto de la memoria. Este mecanismo de recuperación es el que distingue a los sistemas con GraphRAG (C y D) de los que carecen de él (A y B), y los resultados de la evaluación automática —realizada sobre cincuenta casos *held-out* con un juez LLM (`qwen2.5:32b`), no todavía con el panel docente humano previsto como trabajo futuro— muestran que la augmentación con el subgrafo aporta mejoras reales y medibles en el acierto de concepto (que asciende de 0,18 en el LLM base a 0,48 con GraphRAG) y en la trazabilidad, aunque no en todas las dimensiones por igual y sin que ninguno de los objetivos numéricos aspiracionales del anteproyecto pueda darse por alcanzado. La evaluación del sistema híbrido D (que combina el modelo afinado con la augmentación GraphRAG) se ha completado con esta muestra de cincuenta casos. D gana o empata en

las siete dimensiones evaluadas (es el mejor en seis de ellas) y confirma así que la complementariedad observada entre el *fine tuning* (B) y el GraphRAG (C) puede sintetizarse en un único sistema, donde D supera a B y C, que a su vez superan al LLM base A. Aun así, ninguno de los objetivos numéricos del anteproyecto se alcanza, ya que el mejor sistema, D, llega a 0,76 en el acierto de categoría, lejos del umbral aspiracional, y la validación sigue pendiente del panel humano reservado como trabajo futuro. La consulta-caso de uso descrita en este capítulo es, por tanto, la pieza que convierte la promesa teórica del GraphRAG en una intervención concreta y auditable; su eficacia real, con sus luces y sus límites, se documenta en los capítulos de resultados.

16 Enlazado de datos y reutilización de vocabularios

Uno de los principios que más decididamente han orientado la construcción del grafo de conocimiento educativo (EKG) de este trabajo no figuraba en las metas cuantitativas del anteproyecto, pero ha condicionado casi todas las decisiones de modelado. Me refiero a la voluntad de no reinventar lo que la comunidad de la web semántica ya había acordado. La promesa de los datos enlazados, tal como la articulan Hogan et al. (2021) en su panorámica sobre grafos de conocimiento, no se agota en publicar tripletas RDF; reside en conectar esas tripletas con un ecosistema más amplio de vocabularios y de recursos identificados de forma estable, para que el grafo deje de ser una isla y pase a formar parte de una red de significados compartidos. Este capítulo describe cómo he materializado ese principio en el EKG. Detallo la reutilización deliberada de vocabularios estándar antes de acuñar términos propios, el enlazado de treinta conceptos del dominio con Wikidata mediante `skos:exactMatch` (con la verificación, una por una, de cada identificador), el uso de consultas federadas para aprovechar conocimiento externo sin replicarlo y la red de relaciones `skos:broader` que organiza jerárquicamente el dominio. El relato es también, en parte, una crónica de las pequeñas frustraciones y de los hallazgos que acompañan a un trabajo de enlazado hecho con cuidado.

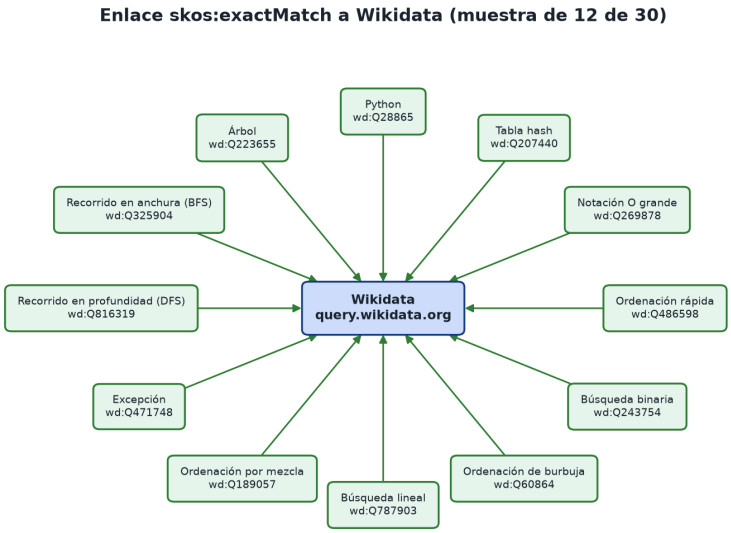

Figura 33. Enlace a Wikidata mediante skos:exactMatch (muestra de 12 de los 30 enlaces). *Fuente: elaboración propia a partir del grafo canónico `ekg-python-150.ttl`.*

16.1 Reutilizar antes de acuñar

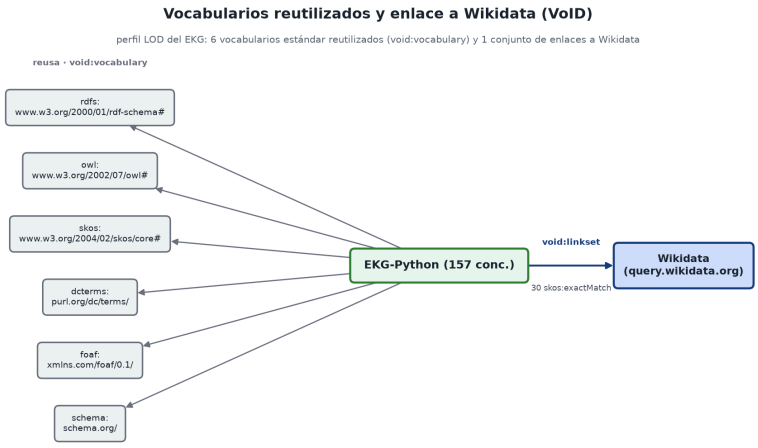

Figura 34. Vocabularios reutilizados y enlace a la Web de Datos, según void.ttl. *Fuente: elaboración propia a partir de la descripción del conjunto de datos `void.ttl`.*

La primera regla que me impuse al diseñar el esquema fue conceptualmente sencilla de enunciar y sorprendentemente exigente de cumplir. Antes de crear una clase o una propiedad nueva en el espacio de nombres propio `pyedu:`, debía comprobar si algún vocabulario consolidado ya ofrecía un término adecuado. Esta disciplina responde a una de las recomendaciones más reiteradas de la literatura sobre datos enlazados (Hogan et al., 2021), y su justificación es a la vez práctica y epistemológica. Práctica, porque cada término reutilizado es un término que otros agentes (humanos o automáticos) ya saben interpretar, lo que multiplica la interoperabilidad del grafo sin coste adicional. Y epistemológica, porque acuñar un identificador nuevo para algo que ya tiene nombre introduce ruido en la web de datos y obliga, tarde o temprano, a tender puentes de equivalencia que podrían haberse ahorrado.

En la práctica, la reutilización se ha apoyado en cuatro vocabularios consolidados, que la Figura 34 sitúa junto al enlace del grafo con la nube de datos abiertos. SKOS (W3C, 2009), el *Simple Knowledge Organization System*, ha resultado especialmente natural para representar la estructura conceptual del dominio, pues los conceptos de programación no son tanto clases ontológicas en sentido estricto cuanto unidades de un sistema de organización del conocimiento, y propiedades como `skos:broader`, `skos:narrower` o `skos:prefLabel` capturan con precisión las relaciones de generalización y las etiquetas preferentes sin comprometernos con una semántica de clases más rígida de lo necesario. Dublin Core (DC) ha proporcionado el vocabulario de metadatos descriptivos (autoría, fechas, descripciones) que toda colección de recursos necesita y que sería absurdo redefinir. FOAF, pese a su orientación originaria hacia las redes sociales, ofrece términos útiles para modelar a los agentes que intervienen en el grafo, como autores de recursos o evaluadores. Y schema.org, con su amplísima cobertura de tipos relativos a recursos educativos, ha permitido alinear ciertas entidades del EKG con un vocabulario que motores de búsqueda y plataformas de terceros reconocen de forma nativa.

El resultado de esta política es que el esquema propio quedó reducido a aquello que verdaderamente constituye la aportación específica del dominio: las veinte clases, las veintiuna propiedades de objeto y las siete propiedades de datos que articulan los conceptos de programación, los errores típicos, las actividades de evaluación y sus relaciones. Todo lo demás (metadatos, jerarquía conceptual, identificación de agentes, descripción de recursos) se apoya en vocabularios externos. El alcance de esa reutilización tiene límites, porque no ha sido total ni siempre limpia. El dominio de la enseñanza de la programación tiene particularidades (la noción de «error conceptual recurrente», por ejemplo, o las relaciones N-arias entre actividad, concepto evaluado y resultado) para las que ningún vocabulario estándar ofrecía un encaje directo, y ahí sí hubo que acuñar. Pero el criterio rector se mantuvo firme. Acuñar fue siempre la excepción justificada, no el reflejo por defecto.

16.2 Treinta enlaces a Wikidata, verificados uno a uno

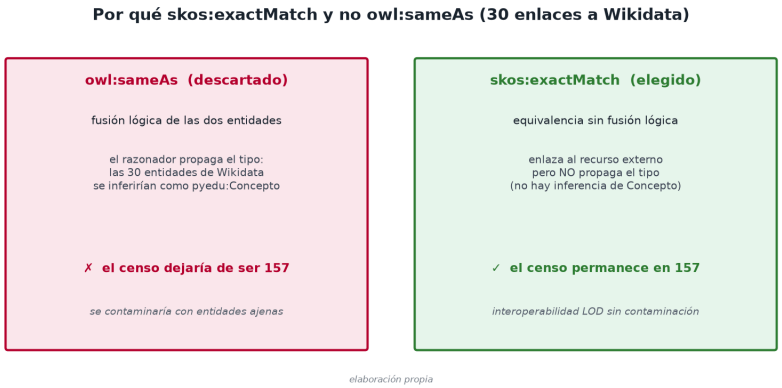

Figura 35. Por qué skos:exactMatch y no owl:sameAs (el censo permanece en 157). *Fuente: elaboración propia.*

El enlazado externo más visible del EKG es el conjunto de treinta afirmaciones `skos:exactMatch` que conectan conceptos propios con sus entidades equivalentes en Wikidata, del que la Figura 33 muestra una selección de doce enlaces. La elección de Wikidata como punto de anclaje no fue casual. Se trata de la mayor base de conocimiento colaborativa y multilingüe disponible, con identificadores estables (los QID), una cobertura amplia de conceptos informáticos y un endpoint SPARQL público que permite, como veremos, consultas federadas. Conectar, por ejemplo, el concepto propio de «recursión» o el de «bucle» con su QID correspondiente significa que cualquier agente que consuma el grafo puede, siguiendo el enlace, obtener etiquetas en decenas de idiomas, definiciones, relaciones adicionales y vínculos hacia otros recursos, sin que el EKG tenga que almacenar ni mantener nada de eso.

La elección del predicado de enlace no fue trivial y merece detenerse en ella. La semántica de `owl:sameAs` (W3C, 2012) es fuerte y poco indulgente. Afirma identidad estricta entre dos recursos, de modo que todo lo que se predica de uno se predica del otro. Bajo el perfil OWL 2 RL adoptado en el grafo, esa fortaleza tiene una consecuencia que en este caso resultaría indeseable. El razonador OWL-RL fusionaría lógicamente el concepto propio con la entidad de Wikidata y propagaría a uno todos los tipos y relaciones del otro como si fueran un único individuo. Eso significaría, entre otras cosas, que las treinta entidades de Wikidata pasarían a inferirse como instancias de `pyedu:Concepto`, una conclusión que no se corresponde con la realidad, porque esas entidades son recursos externos enlazados, no conceptos acuñados en nuestro dominio. Por esa razón opté deliberadamente por `skos:exactMatch`, que afirma una correspondencia de equivalencia entre dos conceptos sin imponer la fusión lógica de individuos que el perfil OWL-RL deriva de `owl:sameAs` (Figura 35). Este predicado enlaza sin fundir y, por ello, no propaga el tipo. Tal decisión (preferir un enlace que conecta sin colapsar identidades) refleja una madurez en el uso de datos enlazados, que distingue entre afirmar una equivalencia útil para la navegación y el descubrimiento y forzar una identidad lógica con todas sus consecuencias inferenciales.

La materialización con OWL-RL ilustra bien el efecto de esta elección. Tras el cierre OWL-RL el grafo alcanza las 4786 tripletas, y la consulta de conceptos del dominio pasa de recuperar 0 elementos sin inferencia a recuperar 157 con inferencia activada; esos 157 son los conceptos propios del EKG. Las treinta entidades de Wikidata, enlazadas con `skos:exactMatch`, no engrosan esa cifra porque `skos:exactMatch` no propaga el tipo `pyedu:Concepto` (a diferencia de lo que ocurriría con `owl:sameAs`), de modo que el enlazado externo enriquece el grafo con vías de navegación y federación sin contaminar el censo de conceptos propios. Es un ejemplo pequeño pero ilustrativo de cómo la elección cuidadosa del predicado de enlace preserva la integridad semántica del grafo. Y, aun sin la fusión lógica, un enlace mal puesto sigue siendo un riesgo. Una correspondencia de equivalencia falsa induce a error a cualquier agente que la siga. Por eso cada uno de los treinta enlaces se verificó manualmente, comprobando que el QID apuntaba a la entidad pretendida y no a un homónimo, a un concepto vecino o a una acepción distinta.

Esa verificación, que en abstracto parece un mero trámite, resultó ser uno de los trabajos más instructivos de toda la fase de construcción del EKG. Narraré una anécdota concreta, porque condensa la lección. Al enlazar uno de los conceptos creí poder ahorrarme la consulta, pues recordaba de memoria, de un trabajo anterior, el QID asociado a esa noción, y lo introduje directamente confiando en mi propia «tabla hash» mental. El identificador era sintácticamente válido, resolvía a una entidad real de Wikidata y, en una primera lectura apresurada, parecía corresponder al concepto deseado. Solo al verificarlo con el debido escepticismo (consultando la etiqueta, la descripción y las relaciones de esa entidad) descubrí que el QID recordado designaba una acepción próxima pero no idéntica a la que yo quería enlazar. Mi memoria había recuperado un valor que existía, que no producía ningún error visible y que, sin embargo, era incorrecto, una victoria pirrica del recuerdo sobre la comprobación. De haberlo dejado pasar, el razonador habría tratado dos conceptos distintos como uno solo y habría propagado esa identificación espuria por toda la red de inferencias. Extraigo de aquí una moraleja deliberadamente humilde. En el enlazado de datos, recordar no es saber, y el coste de verificar un QID es siempre menor que el coste de depurar las consecuencias de uno equivocado.

Esa experiencia confirmó por qué decidí verificar los treinta enlaces sin excepción, incluidos aquellos sobre los que no albergaba ninguna duda. La equivalencia afirmada por `skos:exactMatch` no admite grados de confianza intermedios. O es correcta o es falsa, y la única forma de distinguir lo uno de lo otro es ir a mirar. El proceso de verificación consistió, para cada concepto, en localizar el QID candidato, recuperar de Wikidata su etiqueta preferente, su descripción y sus relaciones jerárquicas y de tipo, y contrastar todo ello con la definición que el concepto tenía en el EKG. Solo cuando esa contrastación resultaba concluyente se afirmaba la equivalencia. El número resultante (treinta) es modesto en relación con los 157 conceptos, y lo es de forma consciente, porque preferí un conjunto pequeño de enlaces de los que respondo plenamente a un conjunto amplio de enlaces plausibles pero no verificados. En el enlazado de datos, como en tantas otras tareas, la precisión pesa más que la cobertura cuando lo que está en juego es la integridad semántica del grafo.

16.3 Consultas federadas: aprovechar sin replicar

El enlazado a Wikidata no se queda en una afirmación declarativa de identidad; habilita un mecanismo operativo de gran valor, el de las consultas federadas. SPARQL (W3C, 2013) incorpora la cláusula `SERVICE`, que permite que una consulta lanzada contra el grafo local delegue parte de su evaluación a un endpoint remoto (el de Wikidata, en este caso) y combine los resultados de ambos. La lógica es elegante. En lugar de copiar a nuestro grafo las etiquetas multilingües, las descripciones o las relaciones adicionales de los conceptos enlazados, dejamos que esos datos residan donde se mantienen y los recuperamos bajo demanda en el momento de la consulta. El EKG conserva así lo que le es propio (la estructura del dominio educativo, las actividades de evaluación, la procedencia de cada afirmación) y se apoya en Wikidata para lo que Wikidata hace mejor, sin asumir la carga de replicar ni de mantener actualizado un conocimiento que evoluciona fuera de nuestro control.

Esta arquitectura encaja con la filosofía general del trabajo, que privilegia el despliegue local y la soberanía sobre los datos propios pero reconoce el valor de enriquecerlos con fuentes externas abiertas. La federación, no obstante, impone limitaciones. La dependencia de un endpoint remoto introduce un punto de fallo. Si el servicio de Wikidata no está disponible o responde con lentitud, las consultas federadas se degradan o fallan, mientras que el conocimiento local sigue plenamente accesible. Por esa razón, la federación se ha reservado para enriquecimientos complementarios (etiquetas alternativas, contexto adicional, navegación hacia recursos externos) y no para la funcionalidad crítica del sistema de evaluación, que opera íntegramente sobre el grafo materializado en local mediante GraphDB (Ontotext, s.f.). Esta separación entre lo que el sistema necesita para funcionar y lo que la federación aporta como valor añadido me parece una decisión de diseño prudente, ya que aprovecha la riqueza de la web de datos sin hacer que el sistema dependa de ella para sus tareas esenciales.

16.4 La red de relaciones skos:broader

El último componente del enlazado interno que merece atención es la red de diecinueve relaciones `skos:broader` que vertebra jerárquicamente el dominio. Mientras que `skos:exactMatch` tiende puentes hacia el exterior, `skos:broader` organiza el interior. Afirma que un concepto es más específico que otro y establece una jerarquía de generalización que recorre el dominio desde las nociones más abstractas hasta las más concretas. Que un concepto de «recursión de cola» tenga como concepto más amplio el de «recursión», y este a su vez el de «control de flujo», por poner un ejemplo del tipo de relaciones modeladas, no es un mero adorno taxonómico, sino información estructural que el motor de recuperación del sistema RAG explota directamente.

En efecto, la utilidad pedagógica de esta jerarquía se manifiesta en la fase de recuperación de subgrafos. Cuando el sistema identifica que un fragmento de código del estudiante involucra un concepto determinado, la red `skos:broader` permite expandir la recuperación hacia conceptos más generales o más específicos y contextualizar el error dentro de una estructura de conocimiento en lugar de tratarlo de forma aislada. Esta expansión jerárquica conecta con una idea central de la teoría del feedback formativo (Hattie & Timperley, 2007), según la cual un feedback eficaz no se limita a señalar el error puntual, sino que lo sitúa en relación con los conceptos que el estudiante debería dominar y con aquellos sobre los que ese concepto se asienta. La elección de SKOS para modelar esta jerarquía, en lugar de una subsunción de clases OWL más pesada, responde de nuevo al principio de reutilización y de parsimonia, ya que `skos:broader` expresa la relación de generalización que el dominio necesita, con una semántica bien definida y ampliamente reconocida (W3C, 2009), sin arrastrar los compromisos lógicos que una jerarquía de clases impondría.

La red `skos:broader` del EKG, con sus diecinueve relaciones, es todavía modesta en relación con el total de 157 conceptos, pues no todos están aún integrados en la jerarquía, y su densificación es una de las tareas naturales de ampliación del grafo. He preferido, también aquí, un conjunto de relaciones jerárquicas correctas y revisadas a una cobertura exhaustiva pero apresurada, en coherencia con el criterio que ha guiado todo el enlazado.

16.5 Balance del enlazado

Vistos en conjunto, los cuatro frentes descritos (reutilización de vocabularios, enlaces `skos:exactMatch` a Wikidata, consultas federadas y red `skos:broader`) responden a una misma convicción, la de que un grafo de conocimiento vale tanto por lo que afirma como por aquello con lo que se conecta. La reutilización de SKOS, DC, FOAF y schema.org mantiene el esquema propio en sus justos límites; los treinta enlaces verificados a Wikidata, lejos de ser un gesto meramente decorativo, materializan conocimiento adicional bajo inferencia y abren la puerta a la federación; y la jerarquía `skos:broader` aporta la estructura que el feedback formativo necesita para situar cada error en su contexto. La anécdota del QID recordado de memoria sintetiza, mejor que cualquier formulación abstracta, la lección metodológica que ha presidido este trabajo, pues en el enlazado de datos la disciplina de verificar prevalece sobre la confianza en la propia memoria, y la modestia en las cifras es a menudo la forma más fiel de la precisión. Las cifras reportadas son las realmente obtenidas y verificadas, y los límites señalados (cobertura parcial de la jerarquía, dependencia de la disponibilidad del endpoint en la federación) se declaran abiertamente como puntos de partida para la ampliación futura del grafo.

17 Arquitectura del sistema RAG (O2)

Una vez consolidado el grafo de conocimiento educativo (EKG) descrito en los capítulos anteriores (157 conceptos propios, 1772 enunciados afirmados que ascienden a 4786 tras la materialización OWL-RL, validación SHACL conforme y enlazado con Wikidata), el segundo objetivo del trabajo (O2) consiste en diseñar e implementar la arquitectura que pone ese grafo al servicio de un modelo de lenguaje masivo para generar retroalimentación formativa sobre código de programación. En este capítulo presento una visión de conjunto de dicha arquitectura. Enumero los cinco componentes previstos en el anteproyecto, justifico las decisiones de diseño que la articulan, trazo el flujo extremo a extremo que recorre una entrega de un estudiante desde el fuente hasta el comentario formativo, y abordo las divergencias entre las tecnologías concretas que anticipé en la planificación y las que finalmente he adoptado en la implementación. Los dos componentes más exigentes (el analizador de código basado en árbol de sintaxis abstracta y el motor de recuperación de subgrafos) se desarrollan en detalle en los capítulos siguientes (15 y 16), por lo que aquí me limito a situarlos en el conjunto y a explicar cómo encajan con el resto de piezas.

La motivación de fondo es la que ya argumenté en el marco teórico (capítulo 6), donde un LLM utilizado de forma aislada produce con frecuencia retroalimentación plausible pero infundada, en la que aparecen tanto aciertos pedagógicos como afirmaciones inventadas o atribuciones erróneas de conceptos, sin que el sistema pueda justificar de dónde procede cada juicio (Bender et al., 2021; Zhang et al., 2023). La generación aumentada por recuperación (RAG) introducida por Lewis et al. (2020), y su extensión a grafos de conocimiento conocida como GraphRAG (Edge et al., 2024; Gupta et al., 2024), constituye la estrategia que adopto para anclar la generación en una fuente de conocimiento explícita, estructurada y trazable. La arquitectura que sigue es, en esencia, una instanciación de GraphRAG sobre el dominio concreto de la enseñanza de la programación en Python, en la que la «base de conocimiento» no es un corpus textual sino un grafo RDF curado con semántica formal.

17.1 Visión de conjunto y principios de diseño

Antes de descender al detalle de cada componente fijaré los principios que han guiado el diseño, porque condicionan muchas de las decisiones técnicas posteriores. El primero es la soberanía y el despliegue local, es decir, que todo el procesamiento (análisis del código del estudiante, cálculo de embeddings, consulta al grafo y generación del comentario) ocurre en infraestructura propia, sin que el código ni los datos de los estudiantes salgan hacia servicios de terceros. Esta restricción, que en el anteproyecto justifiqué por motivos de privacidad y de alineamiento con el marco ético del trabajo, no es un mero detalle de despliegue. Descarta de raíz las APIs comerciales de embeddings y de generación, y obliga a seleccionar modelos que quepan en una sola estación de trabajo. El segundo principio es la trazabilidad, pues cada afirmación que el sistema emita sobre la entrega debe poder asociarse a un subgrafo concreto del EKG, para que tanto el docente como el estudiante puedan inspeccionar la procedencia del juicio. Esto exige que la recuperación devuelva no solo texto sino entidades identificables del grafo, y que esa procedencia se propague hasta la capa de visualización (capítulo 19). El tercero es la modularidad, ya que cada componente expone una interfaz nítida y puede sustituirse o evaluarse de forma aislada, lo que ha resultado decisivo para poder comparar los cuatro sistemas (A, B, C y D) del diseño experimental sin reescribir la canalización completa. La Figura 36 sintetiza estos principios rectores y la consecuencia de diseño que se desprende de cada uno.

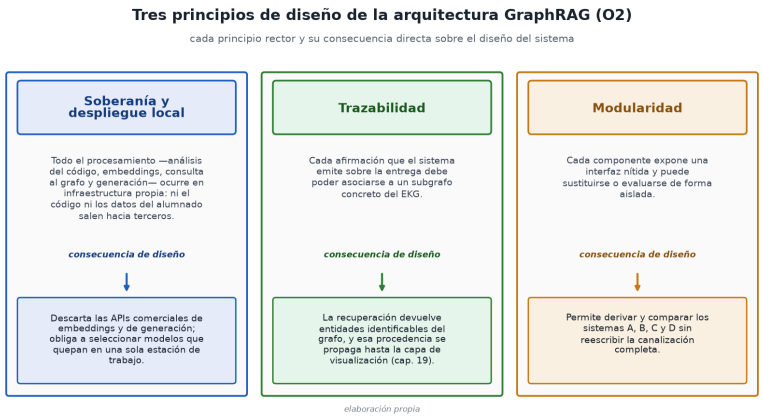

Figura 36. Los principios rectores de la arquitectura (soberanía y despliegue local, trazabilidad y modularidad) y su consecuencia directa sobre el diseño.

Sobre estos principios se asientan los cinco componentes que el anteproyecto fijó como núcleo de O2 y que estructuran el resto del capítulo. Son (1) un **analizador de código** que extrae representación estructural y métricas de la entrega del estudiante; (2) un **indexador de embeddings** que proyecta conceptos y fragmentos a un espacio vectorial para la búsqueda por similitud; (3) un **motor de recuperación de subgrafos** que combina esa búsqueda con expansión simbólica sobre el grafo; (4) un **generador de prompts estructurados** que ensambla el contexto recuperado en una instrucción para el modelo; y (5) la **integración con un LLM local** que produce la retroalimentación. Estos componentes no forman una cadena estrictamente lineal, ya que el analizador y el indexador alimentan ambos al motor de recuperación, y la procedencia recogida durante la recuperación se reutiliza tanto en el generador de prompts como, más adelante, en la capa de explicabilidad. La descripción que sigue respeta, no obstante, el orden en que los datos los atraviesan en una ejecución típica.

El cuadro siguiente sintetiza esos cinco componentes, su función y la tecnología con la que se implementan, antes de desarrollarlos uno a uno.

#	Componente	Función	Tecnología
1	Analizador de código	Extrae representación estructural y métricas de la entrega del estudiante	`ast` + `radon`
2	Indexador de embeddings	Proyecta conceptos y fragmentos a un espacio vectorial para la búsqueda por similitud	`nomic-embed-text` (Ollama)
3	Motor de recuperación de subgrafos	Combina la búsqueda vectorial con la expansión simbólica sobre el grafo	índice vectorial + SPARQL
4	Generador de prompts estructurados	Ensambla el contexto recuperado en una instrucción para el modelo	—
5	Integración con un LLM local	Produce la retroalimentación formativa	`llama3.1:8b` (Ollama)

17.2 Componente 1: el analizador de código

El primer componente recibe el fuente Python de la entrega y produce una representación que el resto del sistema pueda explotar. Su núcleo es el análisis del **árbol de sintaxis abstracta (AST)** mediante el módulo `ast` de la biblioteca estándar de Python, complementado con el cálculo de **métricas de**

complejidad apoyado en la herramienta `radon`. La elección de trabajar sobre el AST, en lugar de sobre el texto plano del programa o sobre representaciones de más bajo nivel, responde a una doble necesidad. Por un lado, el AST permite identificar con fiabilidad las construcciones del lenguaje efectivamente presentes en la entrega (bucles, comprensiones, definiciones de función, manejo de excepciones, llamadas a determinadas funciones incorporadas) que después se mapean a conceptos del EKG; un análisis basado en expresiones regulares sería frágil ante variaciones de estilo o de formato. Por otro lado, las métricas de `radon` (complejidad ciclomática, índice de mantenibilidad, recuento de líneas lógicas) aportan señales cuantitativas sobre la calidad estructural del código que enriquecen el diagnóstico más allá de la mera corrección funcional.

La salida del analizador es, así, un conjunto estructurado de evidencias formado por las construcciones sintácticas detectadas, las métricas de complejidad y, cuando procede, indicios de errores o de antipatrones. Esta representación cumple un papel doble en la arquitectura. Sirve, en primer lugar, como **consulta** para el motor de recuperación, en el sentido de que las construcciones detectadas determinan qué conceptos del grafo son relevantes para esta entrega concreta. Y constituye, en segundo lugar, parte del **contexto factual** que se inyecta en el prompt, de modo que el LLM razone sobre hechos verificados del código y no sobre una lectura aproximada del texto fuente. El detalle de este componente (incluido el mapeo concreto entre nodos del AST y conceptos del EKG, así como el tratamiento de las entregas que ni siquiera compilan) se aborda en el capítulo 15. Aquí basta retener que el analizador es la frontera entre el artefacto del estudiante y la representación semántica del dominio, pues traduce código en señales que el grafo entiende.

17.3 Componente 2: el indexador de embeddings

El segundo componente construye y mantiene el índice vectorial que habilita la recuperación por similitud semántica. Su tarea es proyectar tanto los conceptos del EKG (con sus etiquetas, definiciones y, cuando existen, descripciones provenientes de Wikidata) como los fragmentos derivados del análisis de la entrega a un espacio de vectores densos, de manera que la cercanía geométrica aproxime la afinidad semántica. Sobre ese índice, el motor de recuperación puede después localizar los conceptos del grafo más próximos a las evidencias extraídas del código aun cuando no exista una coincidencia léxica exacta entre el código del estudiante y la terminología del grafo.

El modelo de embeddings adoptado es **`nomic-embed-text`**, servido localmente a través de Ollama, la misma infraestructura que aloja el modelo generativo. Esta unificación de la pila de inferencia tiene una ventaja operativa nada menor. Un único servicio gestiona tanto el cálculo de embeddings como la generación, lo que simplifica el despliegue local y respeta el principio de soberanía sin introducir dependencias adicionales. La decisión de emplear `nomic-embed-text` en lugar del `all-MiniLM-L6-v2` previsto en el anteproyecto se discute con detalle en la sección de divergencias tecnológicas, más abajo, por su relevancia para la integridad del relato. El indexador se concibe, además, como un componente que se reconstruye de forma incremental, de modo que, cuando el EKG se amplía o se reenlaza, basta recalcular los vectores de las entidades afectadas, sin necesidad de reindexar el grafo completo, lo que resulta práctico durante el desarrollo iterativo del propio grafo descrito en la fase I.

17.4 Componente 3: el motor de recuperación de subgrafos

El tercer componente es el corazón de la estrategia GraphRAG y aquello que diferencia esta arquitectura de un RAG textual convencional. Su función es seleccionar, para cada entrega, el **subgrafo del EKG** que constituye el contexto pertinente para la generación. La recuperación se realiza en dos etapas complementarias. La primera es una **búsqueda por similitud** sobre el índice vectorial, donde, a partir de las evidencias extraídas por el analizador, se localizan los conceptos del grafo semánticamente más próximos, lo que proporciona un conjunto de «nodos semilla». La segunda es una **expansión simbólica mediante SPARQL** (W3C, 2013), en la que, a partir de esos nodos semilla, se navega el grafo siguiendo relaciones explícitas (prerrequisitos, generalizaciones `skos:broader`, asociaciones de error a concepto) para incorporar el vecindario relevante. Esta combinación es deliberada. La búsqueda vectorial aporta

robustez frente a la variabilidad superficial del código y de la terminología, mientras que la expansión SPARQL aporta la precisión y la estructura que solo un grafo con semántica formal puede ofrecer. El resultado no es una lista plana de pasajes, sino un fragmento conexo del grafo con sus relaciones, que es lo que después permite la visualización del subgrafo y los marcadores de procedencia.

Esta arquitectura híbrida (vectorial más simbólica) es la que explota plenamente el trabajo de construcción del grafo realizado en O1. Las relaciones de prerrequisito y las jerarquías `skos:broader` no son adornos, sino los caminos que la expansión recorre para situar un error concreto en su contexto conceptual y pedagógico (por ejemplo, para vincular un mal uso de una comprensión de listas con el concepto de bucle del que depende, o con el prerrequisito que probablemente no se ha consolidado). La consulta de conceptos que, sin inferencia, devolvía 0 resultados y, con la materialización OWL-RL, devuelve 157[24] ilustra hasta qué punto la calidad de la recuperación depende del razonamiento previo sobre el grafo. El detalle algorítmico de la recuperación (umbrales de similitud, profundidad de expansión, criterios de poda del subgrafo) se desarrolla en el capítulo 16, y la profundidad de recuperación se retoma además en el análisis de trade-offs del capítulo 22.

17.5 Componente 4: el generador de prompts estructurados

El cuarto componente ensambla el subgrafo recuperado, las evidencias del analizador y la información de nivel del estudiante en una instrucción coherente para el modelo de lenguaje. Su diseño no es un asunto menor de «ingeniería de prompts», ya que de la forma en que se serializa el subgrafo y se enmarca la tarea depende en buena medida que el modelo se ciña al contexto proporcionado en lugar de recurrir a su conocimiento paramétrico no verificado. El generador convierte el subgrafo en una representación textual estructurada (conceptos implicados, sus definiciones, las relaciones recuperadas y los identificadores que permiten rastrear la procedencia) y la combina con un encuadre de la tarea que explicita el rol del modelo (asistente de retroalimentación formativa, no corrector sumativo), las dimensiones que debe cubrir y el nivel del estudiante al que debe adaptar el registro y la profundidad de la explicación.

Este componente es también el lugar donde se materializa el principio pedagógico que vertebra todo el trabajo. La retroalimentación que se persigue es la formativa en el sentido de Hattie y Timperley (2007), orientada al proceso y a la autorregulación, no la mera señalización del error. Por ello el prompt no solo pide identificar qué está mal, sino situar el problema en el mapa conceptual del dominio, sugerir el siguiente paso de aprendizaje y, cuando procede, remitir al prerrequisito que conviene reforzar. La estructura del prompt está pensada, además, para inducir respuestas en las que cada afirmación pueda asociarse a una entidad del subgrafo, lo que prepara el terreno para la evaluación de trazabilidad del capítulo 21. Conviene acotar el alcance. La inclusión del subgrafo en el prompt reduce la propensión a la invención, pero no la elimina; el modelo conserva libertad para parafrasear y para introducir matices no respaldados, y por ese motivo la trazabilidad se mide empíricamente en lugar de darse por garantizada.

17.6 Componente 5: la integración con el LLM local

El quinto componente es el modelo generativo que, a partir del prompt estructurado, produce el comentario de retroalimentación. La integración se realiza con un **LLM local servido mediante Ollama**, y el modelo base de referencia es **`llama3.1:8b`**. La elección de un modelo de 8 000 millones de parámetros y de un entorno de inferencia local es, de nuevo, una consecuencia directa del principio de soberanía. Un modelo de ese tamaño puede ejecutarse en la estación de trabajo del proyecto sin recurrir a servicios externos, lo que mantiene el código y los datos de los estudiantes dentro de la infraestructura controlada. El precio de esa decisión es una capacidad de razonamiento inferior a la de los modelos de frontera, lo que hace aún más pertinente la augmentación, porque, al ser modesto el modelo, anclarlo en un grafo de conocimiento curado tiene el potencial de compensar parte de esa limitación.

[24] los 157 conceptos propios; las 30 entidades de Wikidata, enlazadas con `skos:exactMatch`, no se infieren como `pyedu:Concepto`, porque `skos:exactMatch` no propaga el tipo, a diferencia de lo que ocurriría con `owl:sameAs`

Este componente es, además, el punto de articulación con la otra gran línea de trabajo de la tesis, que es el *fine tuning*. El diseño experimental contempla cuatro configuraciones que comparten el resto de la arquitectura y difieren en este componente y en si la recuperación está activa. El **Sistema A** emplea el modelo base `llama3.1:8b` sin augmentación; el **Sistema B** sustituye el modelo por una variante afinada mediante QLoRA sobre `Qwen2.5-Coder-7B-Instruct` (Dettmers et al., 2023; Hu et al., 2021); el **Sistema C** mantiene el modelo base pero activa la recuperación GraphRAG completa; y el **Sistema D** combina el modelo afinado con la recuperación, en una configuración híbrida. Que estas cuatro variantes puedan construirse sin más que reconfigurar el componente generativo y activar o desactivar la recuperación es la mejor evidencia de que el principio de modularidad se ha respetado en la práctica. El detalle del *fine tuning* se trata en el capítulo 17, y la comparación de sistemas, en el capítulo 18.

La tabla siguiente recoge las cuatro configuraciones experimentales según sus dos puntos de variación: el modelo generativo y la activación de la recuperación.

Sistema	Modelo generativo	Recuperación GraphRAG
A	`llama3.1:8b` (base)	Inactiva
B	`Qwen2.5-Coder-7B-Instruct` afinado con QLoRA	Inactiva
C	`llama3.1:8b` (base)	Activa
D	`Qwen2.5-Coder-7B-Instruct` afinado con QLoRA	Activa

17.7 Flujo extremo a extremo

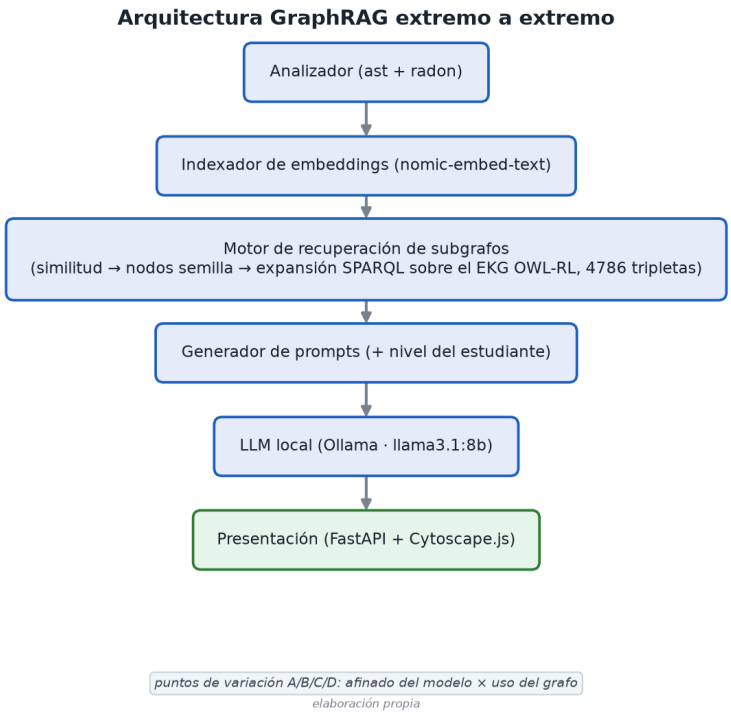

Figura 37. Arquitectura GraphRAG extremo a extremo (puntos de variación A/B/C/D).

Recorro ahora la arquitectura de principio a fin con una entrega concreta, porque el modo en que los datos atraviesan los cinco componentes aclara mejor que cualquier diagrama las dependencias entre ellos. El proceso comienza cuando el estudiante envía un fuente Python como respuesta a un ejercicio. Ese fuente entra en el **analizador de código**, que lo parsea a su AST, identifica las construcciones del lenguaje presentes, calcula las métricas de complejidad con `radon` y detecta indicios de error. El resultado es un conjunto de evidencias estructuradas que describen qué hace y cómo está construida la entrega.

Esas evidencias alimentan a continuación al **motor de recuperación**. En su primera etapa, las construcciones y términos detectados se proyectan al espacio vectorial mediante el **indexador de embeddings** (`nomic-embed-text` vía Ollama) y se cotejan con el índice de conceptos del EKG para obtener los nodos semilla más afines. En la segunda etapa, una consulta SPARQL parte de esos nodos y expande el vecindario relevante siguiendo las relaciones del grafo (prerrequisitos, jerarquías `skos:broader`, vínculos de error a concepto), y devuelve un subgrafo conexo con su procedencia. Este subgrafo es la pieza central del contexto, pues contiene los conceptos implicados, sus definiciones (algunas enriquecidas desde Wikidata por los enlaces `skos:exactMatch`) y las relaciones que los conectan.

El **generador de prompts estructurados** recoge entonces tres flujos de información (las evidencias del analizador, el subgrafo recuperado y el nivel del estudiante) y los ensambla en una instrucción que encuadra la tarea como retroalimentación formativa, fija las dimensiones que el comentario debe cubrir y serializa el subgrafo de modo que cada afirmación pueda anclarse a una entidad identificable. Ese prompt se entrega al **LLM local** (en la configuración que corresponda, ya sea base o afinado, con o sin recuperación activa), que genera el comentario de retroalimentación. Finalmente, el comentario, junto con el subgrafo que lo sustenta y sus marcadores de procedencia, se entrega a la capa de presentación, donde la interfaz web construida con FastAPI y Cytoscape.js (capítulo 19) muestra al docente y al estudiante tanto el texto del feedback como la visualización del subgrafo recuperado, con código de color y procedencia, y cierra así el lazo de explicabilidad.

En este recorrido importa marcar dónde se bifurca el flujo según el sistema, como recoge la Figura 37. En el **Sistema A**, el motor de recuperación queda inactivo y el prompt se reduce a las evidencias del analizador y al fuente, de manera que el modelo base responde sin augmentación. En el **Sistema C**, el flujo es el completo recién descrito sobre el modelo base. En los sistemas **B** y **D** se sustituye el componente generativo por el modelo afinado, sin recuperación en B y con recuperación en D. Esta estructura común con puntos de variación bien acotados es la que ha hecho viable el diseño experimental comparativo con un esfuerzo de implementación contenido.

17.8 Decisiones tecnológicas frente a la propuesta inicial

Conviene reconocer con claridad las divergencias entre las tecnologías que el anteproyecto anticipó para O2 y las que la implementación ha terminado adoptando. El anteproyecto mencionaba como candidatos el modelo de embeddings `all-MiniLM-L6-v2`, la biblioteca FAISS como índice vectorial y la base de datos de grafos Neo4j como triplestore. La implementación real emplea, en su lugar, **`nomic-embed-text` para los embeddings** y la combinación de **`rdflib` y GraphDB** para el almacenamiento y la consulta del grafo. No se trata de desviaciones casuales, sino de decisiones de diseño que conviene justificar, porque afectan a propiedades sustantivas del sistema y porque importa dejar constancia de lo que efectivamente se ha construido. La Figura 38 presenta esa correspondencia entre la pila planificada en el anteproyecto y la finalmente adoptada, componente a componente.

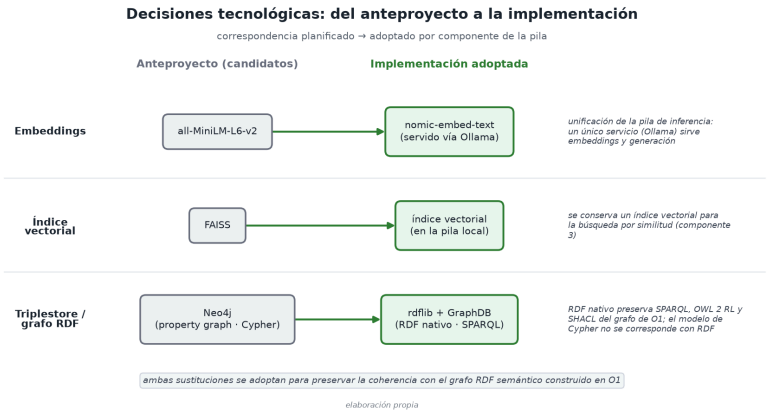

Figura 38. Correspondencia entre la pila tecnológica planificada en el anteproyecto y la adoptada en la implementación, por componente.

La sustitución de `all-MiniLM-L6-v2` por `nomic-embed-text` responde, en primer lugar, al principio de unificación de la pila de inferencia. Al servirse el modelo de embeddings a través de la misma instancia de Ollama que aloja el modelo generativo, el despliegue local se simplifica, ya que un único servicio gestiona ambas tareas, lo que reduce las dependencias, facilita la reproducibilidad del entorno y mantiene coherente el principio de soberanía. A ello se suma que `nomic-embed-text` ofrece una ventana de contexto y una calidad de representación apropiadas para textos técnicos y para las definiciones de los conceptos del EKG, sin un coste computacional que comprometa el despliegue en una sola estación de trabajo. La elección, en suma, prioriza la coherencia operativa del sistema local por encima de la mera réplica de la herramienta nombrada en la planificación.

La segunda divergencia es la del triplestore. Aquí la decisión está fuertemente condicionada por la naturaleza del trabajo de O1. El EKG se ha construido como un grafo RDF con semántica formal, que comprende perfil OWL 2 RL, materialización por inferencia, validación SHACL y enlazado mediante `skos:exactMatch` a Wikidata. Se eligió deliberadamente `skos:exactMatch` sobre `owl:sameAs` para enlazar sin imponer la fusión lógica de individuos que el razonador OWL 2 RL aplicaría con `owl:sameAs` (fusión que confundiría el concepto propio con la entidad de Wikidata), una decisión que refleja madurez en el uso de datos enlazados. Esa apuesta por el ecosistema de la Web Semántica del W3C, con RDF, RDFS, OWL 2, SPARQL, SHACL y SKOS (W3C, 2014a, 2014b, 2012, 2013, 2017, 2009), hace de `rdflib` y GraphDB la elección natural, porque ambos son herramientas RDF nativas que soportan SPARQL estándar y razonamiento sobre OWL/RDFS de manera directa. `rdflib` proporciona la manipulación programática del grafo en Python durante la construcción y el desarrollo, mientras que GraphDB (Ontotext, s.f.) aporta un triplestore con capacidades de inferencia y consulta SPARQL adecuadas para la fase de explotación. Neo4j, en cambio, se asienta sobre el modelo de grafos de propiedades y su lenguaje Cypher, que no se corresponde directamente con la semántica RDF ni con la materialización OWL-RL sobre la que descansa todo el razonamiento del EKG; adoptarlo habría obligado a renunciar a parte del aparato semántico construido en O1 o a sostener una capa de traducción artificial entre dos modelos de datos distintos. La decisión de emplear un triplestore RDF nativo es la que da coherencia al conjunto, ya que la recuperación de subgrafos mediante expansión SPARQL del componente 3 solo es posible, en los términos descritos, porque el grafo vive en un almacén que entiende RDF y SPARQL de forma nativa.

Reconocer estas divergencias no es una concesión a la planificación fallida, sino una muestra de cómo el diseño se ha refinado a medida que las decisiones de O1 imponían sus consecuencias sobre O2. El anteproyecto fijaba candidatos razonables con la información disponible en su momento; la implementación, al haberse comprometido firmemente con el paradigma de la Web Semántica para el grafo, ha terminado seleccionando las herramientas que mejor preservan esa coherencia de extremo a extremo. Dejar constancia explícita de este ajuste, y de sus motivos, forma parte del mismo proceder que recorre

el resto de la memoria, donde los resultados experimentales se reportan tal como son y no como se esperaba que fueran.

17.9 Síntesis

La arquitectura presentada instancia el paradigma GraphRAG sobre el dominio de la enseñanza de la programación y articula cinco componentes (analizador de código basado en AST, indexador de embeddings, motor de recuperación de subgrafos híbrido, generador de prompts estructurados e integración con un LLM local) en torno a tres principios rectores, a saber, soberanía mediante despliegue local, trazabilidad de cada afirmación hasta un subgrafo del EKG, y modularidad que ha permitido derivar las cuatro configuraciones experimentales (A, B, C y D) sin reescribir la canalización. El flujo extremo a extremo conduce una entrega desde su fuente Python hasta un comentario formativo acompañado de su subgrafo de procedencia, pasando por el análisis estructural, la recuperación vectorial y simbólica, el ensamblado del prompt y la generación. Las divergencias tecnológicas respecto al anteproyecto (`nomic-embed-text` en lugar de `all-MiniLM-L6-v2`, y `rdflib`/GraphDB en lugar de Neo4j/FAISS) se han adoptado de forma deliberada para preservar la coherencia con el grafo RDF semántico construido en O1, y se documentan aquí con transparencia. Los capítulos 15 y 16 desarrollan, respectivamente, el analizador de código y el motor de recuperación, que son los dos componentes de mayor complejidad de esta arquitectura.

18 Analizador de código estudiantil

El primer componente de la canalización de recuperación aumentada, y la puerta de entrada de cualquier envío del estudiante al sistema, es el analizador de código. Su cometido consiste en transformar una pieza de código fuente (habitualmente incorrecta, incompleta o conceptualmente errónea, como corresponde a un contexto de aprendizaje) en una representación estructurada que pueda dialogar con el resto de la arquitectura. Dicho de otro modo, el analizador es el traductor que media entre el lenguaje de programación que escribe el aprendiz y el lenguaje de la recuperación semántica que opera sobre el grafo de conocimiento educativo (EKG) y sobre el índice de embeddings. Esta mediación dista de ser trivial, porque el código de un principiante rara vez es sintácticamente impecable, frecuentemente combina errores de naturaleza muy distinta y casi nunca viene acompañado de una descripción explícita de la intención del autor. El reto no consiste meramente en extraer información del código, sino en extraer la información *pedagógicamente relevante* que permita después recuperar los conceptos del EKG implicados en el error y formular un diagnóstico fundamentado.

En este capítulo describo las tres tareas que vertebran el analizador. En primer lugar, la extracción sintáctica mediante el módulo `ast` de la biblioteca estándar de Python, que convierte el texto del programa en un árbol de sintaxis abstracta recorrible. En segundo lugar, el cálculo de métricas de complejidad con la herramienta `radon`, en particular la complejidad ciclomática y el índice de mantenibilidad, que aportan una caracterización cuantitativa del código más allá de su mera estructura sintáctica. Y, en tercer lugar, la construcción de la *query* textual que describe el código y su error, y que constituye el insumo que se proyecta al espacio de embeddings para la recuperación de los subgrafos relevantes. Anticipo que este análisis se mantiene deliberadamente en el plano estático. No se ejecuta el código del estudiante. Esta decisión, motivada tanto por razones de seguridad (no es prudente ejecutar código arbitrario procedente de un aprendiz) como por la naturaleza del feedback formativo que persigo, condiciona el alcance de lo que el analizador puede detectar y lo discuto al final del capítulo.

18.1 Extracción sintáctica con `ast`

El punto de partida es el módulo `ast` de la biblioteca estándar de Python, que expone el mismo analizador sintáctico que utiliza el propio intérprete del lenguaje. La función `ast.parse` toma una cadena con código fuente y devuelve un árbol de sintaxis abstracta, una estructura jerárquica de nodos en la que cada nodo representa una construcción del lenguaje (una asignación, una llamada a función, un bucle, una definición de función, una comparación) y donde los nodos hijos representan los componentes sintácticos de cada construcción. Frente a un análisis basado en expresiones regulares o en heurísticas sobre el texto plano, trabajar sobre el AST aporta una ventaja decisiva, pues la representación es la *interpretación canónica* del código tal como la entiende el lenguaje, lo que elimina la ambigüedad inherente a cualquier tratamiento puramente léxico. Donde una expresión regular vería una cadena que contiene la palabra `for`, el AST distingue con precisión un nodo `For` de una variable llamada `for_each` o de la aparición de ese término dentro de un comentario.

El analizador recorre el árbol mediante el patrón de visitante que el propio módulo proporciona a través de la clase `ast.NodeVisitor`. Este recorrido permite contabilizar y caracterizar las construcciones presentes en el envío: cuántas funciones se definen y con qué firmas, qué estructuras de control aparecen (condicionales, bucles `for` y `while`, manejo de excepciones), qué operaciones se realizan, qué nombres se invocan y qué llamadas a funciones de la biblioteca estándar o a métodos se efectúan. De este recorrido emerge un perfil estructural del código que ya no es texto, sino un conjunto de hechos sobre lo que el programa *hace* desde el punto de vista sintáctico. Ese perfil resulta esencial porque muchos de los conceptos modelados en el EKG (bucles, acumuladores, recursión, comprensión de listas, manejo de excepciones, mutabilidad de estructuras) tienen una correspondencia bastante directa con patrones reconocibles en el árbol. Detectar, por ejemplo, que el alumno emplea un bucle `while` con una condición que nunca se modifica dentro del cuerpo abre la puerta a relacionar su envío con el concepto de *bucle infinito* y con los conceptos prerrequisito asociados en el grafo.

Ese recuento se concreta, en la implementación del corpus, en una función que recorre el árbol obtenido y contabiliza funciones, bucles, condicionales y llamadas, y que devuelve un valor nulo cuando el código no es parseable y conmuta entonces al tratamiento del `SyntaxError` que describo a continuación.

def ast_feats(codigo):

try:

arbol = ast.parse(codigo)

except SyntaxError:

return None

c = Counter(type(n).__name__ for n in ast.walk(arbol))

return {«funciones»: c.get(«FunctionDef», 0), «bucles»: c.get(«For», 0) + c.get(«While», 0),

«condicionales»: c.get(«If», 0), «llamadas»: c.get(«Call», 0),

«retornos»: c.get(«Return», 0), «lineas»: codigo.count(«\n») + 1}

Un aspecto que merece una reflexión explícita es el tratamiento del código sintácticamente inválido. Como `ast.parse` reproduce el comportamiento del intérprete, ante un programa con un error de sintaxis lanza una excepción `SyntaxError` y no produce árbol alguno. Lejos de ser un inconveniente, esta circunstancia es en sí misma una fuente de información valiosa, ya que el hecho de que el código no compile, junto con la posición y el mensaje del error sintáctico, constituye un dato diagnóstico de primer orden, especialmente frecuente en los primeros estadios del aprendizaje, donde los errores de indentación, los dos puntos olvidados o los paréntesis sin cerrar son la norma. El analizador captura por tanto la excepción y la incorpora al flujo, de manera que, cuando hay `SyntaxError`, se registra el tipo de error, su localización y su mensaje, y la canalización prosigue con una descripción del problema basada en esa información en lugar de en el perfil estructural completo, que en ese caso no puede construirse. Esta dualidad (análisis estructural cuando el código es parseable, análisis del error sintáctico cuando no lo es) refleja con fidelidad la heterogeneidad de los envíos reales de un estudiante.

Quiero subrayar también lo que el AST *no* captura. Al ser una representación de la sintaxis y no de la ejecución, el árbol no revela errores semánticos que solo se manifiestan en tiempo de ejecución.[25] Para una porción de estos casos el sistema dispone de información complementaria (típicamente el resultado de comparar la salida esperada con la obtenida, o el mensaje de la excepción producida al ejecutar el código contra un conjunto de pruebas en un entorno controlado externo al analizador), información que se integra después en la *query* textual. El analizador, en sentido estricto, se limita al plano estático; la riqueza diagnóstica del sistema surge de combinar ese plano con las señales dinámicas que se le aportan desde fuera.

18.2 Métricas de complejidad con `radon`

La extracción sintáctica describe *qué* construcciones contiene el código, pero no cuán complejo o cuán mantenible es. Para complementar esa dimensión cualitativa con una caracterización cuantitativa se incorpora `radon`, una biblioteca de Python especializada en métricas de software que opera, igual que el analizador propio, sobre el árbol de sintaxis abstracta y sin ejecutar el programa. De las métricas que `radon` ofrece, el analizador utiliza principalmente dos, la complejidad ciclomática y el índice de mantenibilidad.

La complejidad ciclomática, propuesta originalmente por McCabe, cuantifica el número de caminos linealmente independientes a través del flujo de control de una pieza de código. Intuitivamente, cada estructura que introduce una bifurcación (un `if`, un `elif`, un `for`, un `while`, un `and` o un `or`

[25] un `IndexError`, una división por cero dependiente de los datos, un resultado numéricamente incorrecto pero sintácticamente impecable

dentro de una condición, una cláusula `except`) incrementa el número de caminos posibles y, con ello, la complejidad ciclomática. Una función con un valor bajo sigue un flujo prácticamente lineal y resulta fácil de seguir y de probar; un valor elevado señala una función con muchas ramificaciones, más difícil de comprender y más propensa a contener errores. En el contexto educativo esta métrica adquiere un matiz interesante que se aparta de su uso habitual en ingeniería del software profesional. No se trata aquí de penalizar la complejidad como un *code smell*, sino de *caracterizar el nivel del estudiante y la naturaleza de su solución*. Un valor ciclomático desproporcionadamente alto para un problema sencillo puede delatar que el aprendiz ha resuelto mediante una maraña de condicionales anidados lo que admitía una solución más idiomática y compacta,[26] lo que orienta el feedback hacia conceptos del EKG relacionados con el refactor, la simplificación o el uso de construcciones de más alto nivel.

El índice de mantenibilidad es una métrica compuesta que `radon` calcula combinando la complejidad ciclomática, el volumen de Halstead (derivado del recuento de operadores y operandos) y el número de líneas de código, y devuelve un valor en una escala acotada donde los valores más altos indican código más mantenible y los más bajos, código difícil de comprender y modificar. Para el aprendiz, un índice de mantenibilidad bajo suele correlacionarse con código verboso, repetitivo o con identificadores poco expresivos, todos ellos aspectos sobre los que el feedback formativo puede incidir con provecho. Debo matizar, no obstante, el peso relativo de estas métricas en el diagnóstico final. Ni la complejidad ciclomática ni el índice de mantenibilidad determinan por sí solos si el código es *correcto*, pues un programa puede ser de baja complejidad y, aun así, producir un resultado equivocado, y a la inversa. Su papel en la arquitectura es complementario y descriptivo. Enriquecen la caracterización del envío y aportan señales que matizan el tono y el contenido del feedback, pero no sustituyen al diagnóstico del error, que se apoya principalmente en la estructura sintáctica, en las señales dinámicas externas y en la recuperación sobre el EKG.

La siguiente tabla resume las dos métricas que el analizador toma de `radon`, ambas calculadas sobre el árbol de sintaxis abstracta y sin ejecutar el programa.

Métrica	Qué cuantifica	Lectura del valor	Señal pedagógica
Complejidad ciclomática (McCabe)	Caminos linealmente independientes del flujo de control; cada bifurcación (`if`, `elif`, `for`, `while`, `and`/`or`, `except`) la incrementa	Valor bajo: flujo prácticamente lineal, fácil de seguir y de probar. Valor elevado: muchas ramificaciones, más difícil de comprender y propenso a errores	Caracteriza el nivel del alumno y su solución; un valor desproporcionado para un problema sencillo orienta hacia el refactor, la simplificación o construcciones de más alto nivel
Índice de mantenibilidad	Métrica compuesta que combina la complejidad ciclomática, el volumen de Halstead (operadores y operandos) y el número de líneas de código	Escala acotada: valores altos indican código más mantenible; valores bajos, código difícil de comprender y de modificar	Un índice bajo suele correlacionarse con código verboso, repetitivo o con identificadores poco expresivos

Una limitación que asumo es que `radon` está concebida para código que al menos sea sintácticamente válido, puesto que, como el resto del análisis, necesita construir el árbol. Por ello, cuando el envío contiene errores de sintaxis estas métricas no se calculan y el analizador prescinde de ellas. La caracterización se apoya entonces en la información del `SyntaxError`. Esta dependencia refuerza la idea, ya apuntada, de que el analizador opera con dos modos según la parseabilidad del código, y de que

[26] por ejemplo, recurriendo a una estructura de datos o a una abstracción que aún no domina

la riqueza de la caracterización es mayor cuando el programa, aun siendo incorrecto en su lógica, es estructuralmente bien formado.

18.3 Construcción de la *query* textual

Las dos primeras tareas producen una representación rica pero heterogénea, formada por un perfil estructural derivado del AST, un conjunto de métricas de complejidad y, según el caso, la información de un error sintáctico o de una excepción en ejecución aportada desde fuera del analizador. El problema es que la recuperación de subgrafos del EKG se articula, en su primera etapa, sobre similitud semántica en un espacio de embeddings, y los embeddings se calculan a partir de *texto en lenguaje natural*, no de árboles sintácticos ni de tablas de métricas. La tercera tarea del analizador consiste, por tanto, en sintetizar toda esa información en una *query* textual, una descripción en prosa que verbaliza qué hace el código, qué error presenta y en qué contexto se enmarca, y que será el insumo que se proyecte al espacio vectorial para recuperar los conceptos del grafo más afines.

La construcción de esta consulta es deliberadamente un acto de *traducción y abstracción*. No se trata de volcar el código fuente literalmente al embedding (lo que sesgaría la recuperación hacia coincidencias superficiales de identificadores o de tokens concretos), sino de describir el código en términos de las *construcciones y los conceptos* que pone en juego, de modo que el vocabulario de la consulta se aproxime al vocabulario con el que están descritos los conceptos en el EKG. Así, en lugar de incrustar `for i in range(len(lista)): total = total + lista[i]`, la consulta verbaliza que el código recorre una secuencia mediante un bucle indexado y acumula un total, y que el error reside, pongamos, en una inicialización incorrecta del acumulador o en un desbordamiento del índice. Esta verbalización tiende un puente entre el plano del código y el plano conceptual del grafo, que es donde reside el conocimiento pedagógico que el sistema quiere movilizar.

La *query* integra varias capas de información, como resume la Figura 39. Una primera capa describe la estructura del programa a partir del perfil del AST, es decir, las construcciones presentes, el patrón algorítmico que parece seguir el código (acumulación, búsqueda, recorrido anidado, recursión) y los elementos del lenguaje implicados. Una segunda capa incorpora la caracterización de complejidad procedente de `radon` cuando resulta pertinente, no como cifras aisladas sino traducida a una valoración cualitativa que pueda influir en la recuperación y en el tono del feedback. Una tercera capa, central para el diagnóstico, describe el error. Si se trata de un fallo sintáctico, se incorpora el tipo y el mensaje del `SyntaxError`; si es un fallo en ejecución o un resultado incorrecto, se incorpora la información de la excepción o de la discrepancia entre la salida esperada y la obtenida, según lo que se haya aportado al analizador. Y una cuarta capa, cuando está disponible, sitúa el envío en su contexto, que es el enunciado del ejercicio o la competencia que se pretendía ejercitar, lo que ayuda a desambiguar la intención del estudiante y a orientar la recuperación hacia la región del grafo realmente relevante.

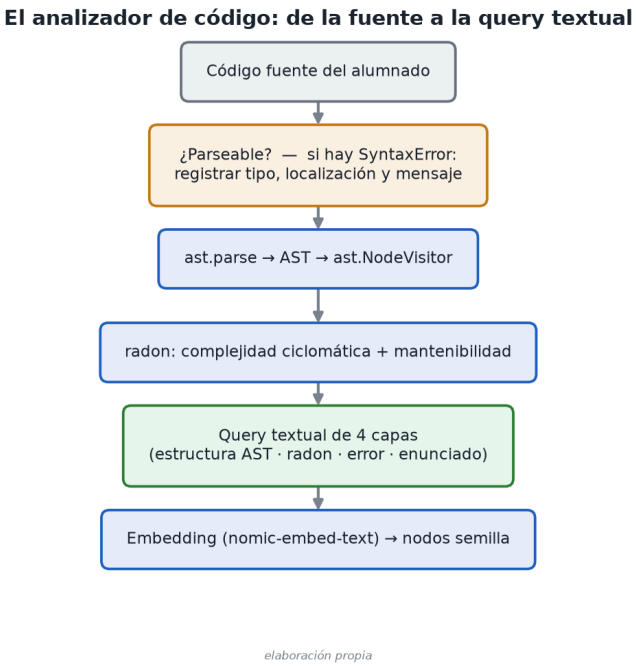

Figura 39. Del código fuente a la query textual de cuatro capas.

El resultado de combinar estas capas es una consulta textual que describe simultáneamente *qué intentaba hacer el código*, *qué hace en realidad* y *en qué falla*, formulada en un registro próximo al de los conceptos del EKG. Sobre esta consulta se calcula el embedding mediante el modelo `nomic-embed-text` servido localmente a través de Ollama, y ese vector se enfrenta por similitud a los embeddings de los conceptos y enunciados del grafo para seleccionar los nodos semilla, que luego se expanden mediante consultas SPARQL sobre GraphDB para recuperar el subgrafo de conceptos, prerrequisitos y relaciones implicado en el error. Esta etapa de recuperación y expansión se detalla en los capítulos correspondientes; aquí basta con señalar que la *query* textual es la interfaz que hace posible ese acoplamiento entre el código del estudiante y la recuperación semántica, y que su diseño condiciona de forma directa la calidad de los subgrafos recuperados y, en cascada, la del feedback generado.

Debo dar cuenta de una desviación respecto del anteproyecto que afecta a esta parte de la arquitectura. El planteamiento inicial contemplaba el uso de `all-MiniLM-L6-v2` como modelo de embeddings, FAISS como índice vectorial y Neo4j como almacén del grafo. La implementación finalmente adoptada emplea `nomic-embed-text` servido por Ollama para los embeddings y `rdflib` junto con GraphDB (Ontotext, s.f.) para la gestión y consulta del grafo en RDF. Esta decisión obedece a tres motivos. El primero es la coherencia de la pila de despliegue local, pues al servir el modelo de embeddings y el LLM a través del mismo runtime de Ollama se simplifica la infraestructura y se refuerza el principio de soberanía de los datos que vertebra el proyecto, al mantener todo el procesamiento dentro del entorno del docente sin dependencias de servicios externos. El segundo es la alineación con los estándares de la Web Semántica, dado que trabajar sobre RDF con GraphDB permite explotar SPARQL (W3C, 2013) y el razonamiento OWL 2 RL (W3C, 2012) de forma nativa sobre el EKG, capacidades que resultan más naturales en un *triple store* que en una base de datos de propiedades como Neo4j. El tercero es de orden práctico, ya que `nomic-embed-text` ofrece un espacio de representación de mayor dimensión y un rendimiento adecuado para mi volumen, sin que el índice vectorial requiera la complejidad de FAISS dado el tamaño manejable del corpus de conceptos. Dejo constancia de este cambio porque documentar las decisiones de ingeniería forma parte del rigor que esta memoria pretende mantener.

18.4 Síntesis y limitaciones

El analizador de código estudiantil cumple, en suma, el papel de transformar un envío crudo en una representación apta para la recuperación semántica, y articula tres operaciones complementarias. La primera es la extracción sintáctica con `ast`, que descompone el programa en un árbol estructurado y maneja con elegancia tanto el código bien formado como el sintácticamente inválido. La segunda es el cálculo de métricas de complejidad con `radon`, que añade una caracterización cuantitativa del código mediante la complejidad ciclomática y el índice de mantenibilidad. La tercera es la construcción de la *query* textual, que abstrae y verbaliza toda esa información en una descripción del código y su error formulada en el registro conceptual del EKG. Las tres operaciones comparten la naturaleza estática del análisis, que es a la vez su principal virtud (seguridad, determinismo, alineación con conceptos del grafo) y su principal límite, pues deja fuera del alcance directo del analizador los errores que solo se manifiestan en ejecución y que deben aportarse como señal externa.

Asumo varias limitaciones. La dependencia de la parseabilidad reduce la riqueza del análisis ante código gravemente malformado, aunque la información del error sintáctico mitiga parcialmente esa carencia. La verbalización del código en la *query* introduce inevitablemente una pérdida de información y un sesgo de diseño, porque la calidad de la recuperación depende de cuán acertadamente se traduzcan las construcciones del código al vocabulario conceptual, y no descarto que ciertas categorías de error queden infrarrepresentadas en esa traducción. Por último, todo el análisis se ha desarrollado y probado sobre Python y sobre envíos sintéticos generados por plantillas, por lo que su generalización a otros lenguajes o a la variabilidad del código real de aula constituye una extensión natural que dejo apuntada como trabajo futuro. Estas cautelas no merman, a mi juicio, el valor del componente, pero sí delimitan lo que puede y lo que no puede esperarse de él dentro de la arquitectura global.

19 Recuperación de subgrafos contextuales

El núcleo de cualquier arquitectura de Generación Aumentada por Recuperación reside en el componente que decide *qué* conocimiento se inyecta en el contexto del modelo de lenguaje (Lewis et al., 2020). En las formulaciones canónicas de RAG ese conocimiento adopta la forma de fragmentos de texto plano recuperados de un corpus documental mediante similitud densa. Mi propuesta en este Trabajo de Fin de Máster se aparta de ese patrón en un aspecto sustantivo, pues el conocimiento que se recupera no es texto libre, sino un fragmento estructurado del Grafo de Conocimiento Educativo (EKG) (un subgrafo) que conserva la semántica formal del dominio. Esta decisión sitúa el trabajo en la familia de los enfoques GraphRAG (Edge et al., 2024; Peng et al., 2024), donde la recuperación explota explícitamente la topología relacional del grafo en lugar de tratar cada hecho como una unidad aislada. En este capítulo describo cómo materializo esta idea en el motor de recuperación de subgrafos contextuales, que constituye el componente del objetivo O2 encargado de tender el puente entre el análisis del código del estudiante y la generación del feedback.

La intuición pedagógica que motiva el diseño es sencilla pero exigente. Cuando un docente experto detecta un error en el código de un estudiante, no lo interpreta de forma aislada, sino que lo ubica en una red de conceptos relacionados (el error con índices fuera de rango remite a la indexación de secuencias, que a su vez se conecta con el modelo de iteración, con la noción de longitud de una colección y con los patrones idiomáticos para recorrerla) y desde esa red elabora una explicación situada en el nivel del aprendiz. Reproducir esa capacidad exige que el sistema, dado un código y un diagnóstico preliminar, no se limite a localizar el concepto más cercano, sino que recupere la *vecindad conceptual* relevante. Se trata del subgrafo que un docente tendría implícitamente presente al razonar sobre ese error. Detallo a continuación los dos mecanismos que, encadenados, producen ese subgrafo. El primero es una fase de *anclaje semántico* basada en embeddings y similitud; el segundo, una fase de *expansión estructural* basada en consultas SPARQL sobre el grafo.

19.1 Embeddings de conceptos con nomic-embed-text

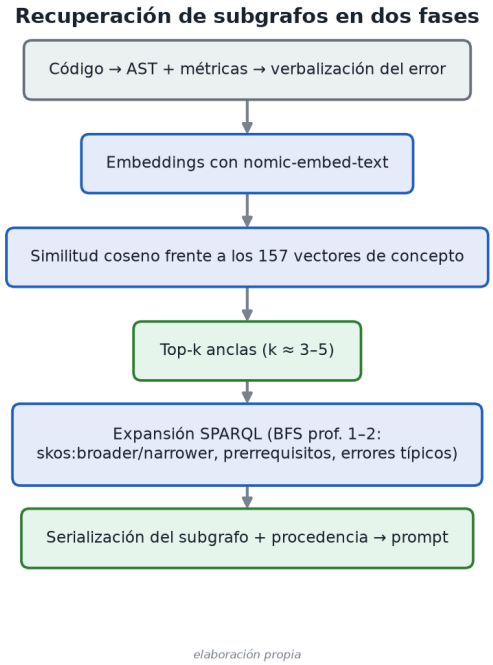

Figura 40. Recuperación de subgrafos en dos fases.

El primer eslabón de la recuperación traduce el espacio simbólico del grafo a un espacio vectorial en el que la proximidad geométrica aproxime la afinidad semántica. Con esa finalidad, cada concepto del EKG se proyecta a un vector denso mediante un modelo de embeddings, y la consulta (derivada del

análisis del código del estudiante) se proyecta al mismo espacio para poder compararla con el catálogo de conceptos. La Figura 40 resume el flujo completo, desde el anclaje semántico de esta fase hasta la expansión estructural posterior.

La elección del modelo de embeddings merece una justificación explícita, porque constituye una de las desviaciones declaradas respecto del anteproyecto. La planificación inicial contemplaba el uso de `all-MiniLM-L6-v2` indexado con FAISS, una combinación habitual y eficaz en escenarios RAG de propósito general. Durante la implementación opté, sin embargo, por `nomic-embed-text` servido localmente a través de Ollama. Varias razones convergen en esta decisión. La primera es de coherencia arquitectónica. Todo el procesamiento de lenguaje del sistema (tanto la generación con `llama3.1:8b` como el juez de evaluación) se apoya en Ollama como capa de servicio local, de modo que incorporar el cálculo de embeddings en la misma infraestructura reduce dependencias, simplifica el despliegue y refuerza el principio de soberanía de los datos que vertebra el proyecto, según el cual ningún texto del estudiante ni del grafo abandona la máquina. La segunda es de calidad representacional. `nomic-embed-text` ofrece una dimensionalidad y una capacidad contextual superiores a las de `all-MiniLM-L6-v2`, con una ventana de contexto holgada que permite codificar la etiqueta de un concepto y también su descripción y sus relaciones textualizadas sin truncamiento. La tercera es de pragmatismo en la escala del problema. El catálogo conceptual del EKG comprende 157 conceptos propios, un volumen para el que el coste de una búsqueda exacta por fuerza bruta sobre los vectores es despreciable. Esto vuelve innecesario un índice aproximado de vecinos como FAISS, cuya complejidad de integración y mantenimiento solo se justifica con corpus de cientos de miles o millones de vectores. Descarté FAISS, por tanto, no por una limitación técnica, sino por un principio de parsimonia. Introducir una dependencia pesada para resolver un problema que una comparación lineal resuelve con holgura habría contravenido la sencillez que persigo en el componente.

La construcción del índice de embeddings procede textualizando cada concepto antes de vectorizarlo. No se codifica únicamente el identificador o la etiqueta `rdfs:label`, sino una representación enriquecida que combina la etiqueta, la definición `skos:definition` cuando existe, y un breve resumen de sus relaciones salientes más significativas. Esta verbalización es deliberada, porque el embedding de un concepto como la indexación de listas resulta semánticamente más informativo si incorpora que «se relaciona con el acceso por posición, está subordinado al concepto de secuencia y admite índices negativos» que si se limita a la cadena «indexación de listas». La textualización de relaciones convierte parte de la estructura del grafo en señal aprovechable por el modelo de embeddings y mitiga la conocida limitación de los modelos densos para capturar relaciones explícitas. El vector resultante se almacena asociado al URI del concepto, de manera que la recuperación devuelva siempre identificadores del grafo (no texto) sobre los que la fase posterior pueda operar mediante SPARQL.

La consulta se construye de forma simétrica. A partir del análisis del código del estudiante mediante árboles de sintaxis abstracta y métricas de complejidad (descrito en el capítulo correspondiente), el sistema produce una descripción textual del error o de la situación detectada (por ejemplo, «acceso a un índice igual a la longitud de la lista en un bucle `for` que recorre `range(len(lista))`») y la proyecta al espacio de embeddings con el mismo modelo `nomic-embed-text`. Emplear idéntico modelo para conceptos y consulta es condición necesaria para que la comparación sea coherente, ya que ambos vectores han de habitar el mismo espacio métrico.

19.2 Similitud y selección top-k

Con la consulta y el catálogo de conceptos representados como vectores, la recuperación de los anclas semánticas se reduce a un problema de vecinos más próximos. La medida de afinidad empleada es la similitud del coseno, que cuantifica la proximidad direccional entre dos vectores con independencia de su magnitud:

$$\operatorname{sim}(\mathbf{q}, \mathbf{c}_i) = \frac{\mathbf{q} \cdot \mathbf{c}_i}{\lVert \mathbf{q} \rVert \, \lVert \mathbf{c}_i \rVert}$$

donde $\mathbf{q}$ es el vector de la consulta y $\mathbf{c}_i$ el del concepto i-ésimo del catálogo. La similitud del coseno es la elección habitual en recuperación densa porque la información semántica de los embeddings tiende a codificarse en la *dirección* del vector más que en su norma; normalizar por la magnitud evita que conceptos con descripciones más largas (y por tanto vectores de mayor norma tras la agregación) dominen artificialmente el ranking.

El sistema calcula la similitud de la consulta frente a los 157 vectores conceptuales, ordena los resultados de forma descendente y selecciona los k primeros, en lo que constituye la operación *top-k*. El valor de k es un hiperparámetro de recuperación con un compromiso claro y que se conecta directamente con el objetivo O5 de análisis de trade-offs sobre la profundidad de recuperación. Un k demasiado pequeño corre el riesgo de no anclar el subgrafo en todos los conceptos pertinentes;[27] un k demasiado grande introduce conceptos espurios que diluyen la señal y, al expandirse después por SPARQL, pueden hacer crecer el subgrafo hasta saturar la ventana de contexto del LLM con ruido. En la configuración empleada se trabaja con valores reducidos de k (del orden de tres a cinco conceptos ancla), partiendo de la observación de que la posterior expansión estructural ya aporta el grueso de la vecindad relevante, de modo que el anclaje denso solo necesita acertar con los *puntos de entrada* al grafo y no con la totalidad de los conceptos finalmente recuperados. Esta división del trabajo (embeddings para localizar la entrada, SPARQL para reconstruir la vecindad) es lo que distingue a un enfoque GraphRAG de un RAG denso convencional, y lo que permite que un anclaje deliberadamente conservador no penalice la cobertura final.

La fase de anclaje tiene un límite metodológico que conviene reconocer. La calidad del anclaje depende de la fidelidad con que la verbalización del error capture la intención del estudiante y de que el embedding sitúe esa verbalización cerca de los conceptos correctos. En errores idiomáticos o sutiles, la descripción textual generada a partir del AST puede ser ambigua, y un anclaje impreciso se propaga a la expansión. Este es uno de los puntos donde la evaluación automática sobre datos sintéticos, con su muestra reducida, no permite todavía cuantificar con solidez la tasa de aciertos del anclaje; caracterizarla con rigor queda señalado como trabajo futuro[28].

19.3 Expansión del subgrafo mediante SPARQL

Los conceptos ancla obtenidos por similitud son, como se ha dicho, *puntos de entrada* y no el contexto completo. La segunda fase los toma como semillas y reconstruye su vecindad conceptual recorriendo las relaciones del grafo mediante consultas SPARQL (W3C, 2013). Esta es la fase donde la naturaleza estructurada del conocimiento se vuelve operativa. En lugar de recuperar hechos independientes, se recupera un fragmento conexo del grafo que preserva las aristas entre conceptos, de manera que el LLM reciba no una lista de nociones sino una red con sus relaciones tipadas.

La expansión se modela como un recorrido en anchura acotado por un parámetro de profundidad. Partiendo de cada concepto semilla, una consulta SPARQL recupera sus vecinos a distancia uno siguiendo las propiedades semánticamente significativas del EKG (las jerarquías `skos:broader` y `skos:narrower`, las relaciones de prerrequisito y dependencia conceptual, las asociaciones temáticas y los vínculos con los tipos de error catalogados). El proceso puede repetirse para alcanzar vecinos a distancia dos e incorpora así los conceptos inmediatamente conectados con el error y, además, su contexto de segundo orden. La profundidad de expansión es, junto con k, el segundo gran hiperparámetro de recuperación analizado bajo el objetivo O5. Una profundidad de uno produce subgrafos compactos y muy centrados, adecuados para errores localizados; una profundidad de dos enriquece el contexto a costa de un crecimiento potencialmente combinatorio del número de nodos, que obliga a aplicar criterios de poda para no desbordar la ventana del modelo. Entre los criterios de control del tamaño se incluyen el filtrado por tipos de relación relevantes, la priorización de los vecinos más próximos a las semillas y la deduplicación de nodos alcanzados por múltiples caminos.

[27] si el error involucra dos nociones distintas, anclar solo en una empobrece el contexto

[28] idealmente con anotación humana del subgrafo «de oro» que un docente recuperaría

Una propiedad de diseño deliberada es que el mecanismo de expansión resulta *idéntico con independencia del motor de grafo subyacente*. El sistema puede ejecutar las mismas consultas SPARQL contra dos backends intercambiables, una instancia local de `rdflib` que carga el grafo serializado en Turtle en memoria o un servidor GraphDB (Ontotext, s.f.) accedido por su endpoint SPARQL. Esta intercambiabilidad no es una mera comodidad de implementación, sino una consecuencia de haberme atenido al estándar SPARQL como interfaz. El componente, al no acoplar la lógica de recuperación a las particularidades de ningún almacén concreto, queda protegido frente a la elección de infraestructura. En desarrollo y en la evaluación con muestras reducidas, `rdflib` resulta más práctico (arranca sin servicios externos, carga el grafo canónico directamente desde el fichero Turtle y facilita la reproducibilidad); en un despliegue con mayor volumen de consultas concurrentes o con necesidad de razonamiento materializado persistente, GraphDB ofrece prestaciones superiores. Que ambas rutas compartan el mismo código de consulta es también lo que permitió descartar la planificación inicial basada en Neo4j, que habría exigido expresar la recuperación en Cypher y atar el componente a un motor propietario y a un lenguaje no estándar, mientras que la adhesión a SPARQL preserva la portabilidad y el alineamiento con la pila de la Web Semántica (Hogan et al., 2021) sobre la que se construye todo el EKG.

Una sutileza relevante de la expansión es su interacción con la inferencia OWL-RL. El subgrafo se recupera sobre el grafo materializado tras el razonamiento, lo que significa que la expansión puede atravesar aristas que no fueron afirmadas explícitamente sino derivadas por el motor de inferencia (por ejemplo, transitividades de `skos:broader` o conceptos alineados vía `skos:exactMatch` con Wikidata). El efecto cuantitativo de esta materialización es el mismo que se documenta para el grafo en su conjunto. Una consulta de conceptos que sin razonamiento devuelve cero resultados alcanza 157 al activarlo, correspondientes a los 157 conceptos propios del EKG. Las 30 entidades de Wikidata enlazadas no engrosan esa cifra, pues se vinculan mediante `skos:exactMatch` (y no mediante `owl:sameAs`), una elección deliberada para no imponer la fusión lógica de individuos que el perfil OWL-RL derivaría de `owl:sameAs`; al no propagar `skos:exactMatch` el tipo de la entidad, dichas entidades no se infieren como `pyedu:Concepto` y permanecen como referencias externas. La recuperación no opera, por tanto, sobre el esqueleto afirmado del grafo sino sobre su clausura inferida. Ello amplía el horizonte de conceptos alcanzables y enriquece el contexto disponible para la generación.

19.4 Serialización del subgrafo como contexto

Una vez recuperado, el subgrafo debe transformarse en una representación textual que el modelo de lenguaje pueda consumir como parte de su prompt. Este paso de serialización es más delicado de lo que aparenta, porque media entre dos formalismos de naturaleza distinta, un grafo de tripletas RDF, riguroso y orientado a máquina, y una secuencia de tokens que un LLM ha de interpretar para razonar pedagógicamente. La serialización es, en buena medida, donde se decide cuánto del valor estructural del subgrafo logra trasladarse efectivamente al razonamiento del modelo.

La estrategia adoptada huye de volcar Turtle crudo en el prompt. Aunque los LLM contemporáneos toleran sintaxis RDF, una serialización Turtle literal abruma el contexto con prefijos, URIs largos y ruido sintáctico que el modelo debe descodificar antes de poder razonar, consumiendo tokens y atención sin aportar señal pedagógica. En su lugar, el subgrafo se *verbaliza*. Cada concepto se presenta con su etiqueta legible y su definición, y cada relación se expresa en lenguaje natural conciso («la indexación de listas es un caso particular de acceso a secuencias», «este error es característico de la confusión entre longitud e índice máximo»). La verbalización preserva la información relacional, que es lo verdaderamente valioso del subgrafo frente a una mera lista de conceptos, pero la presenta en un formato que el modelo procesa sin fricción. Junto a la verbalización se conservan, de forma compacta, los identificadores de procedencia de cada afirmación, de modo que el módulo de explicabilidad pueda después rastrear qué nodo del grafo sustenta cada parte del feedback generado y marcar su origen. Esta conservación de la procedencia es la que habilita la propiedad de trazabilidad que el sistema busca y que la evaluación mide explícitamente; sin ella, el grafo aportaría fundamentación pero no rendición de cuentas verificable.

El subgrafo serializado se ensambla finalmente en la plantilla de prompt junto con el código del estudiante, su análisis estático y las instrucciones de tarea, conformando el contexto completo que se entrega al LLM. El orden y el formato de este ensamblaje no son neutros, pero su tratamiento corresponde al capítulo de generación de prompts; aquí basta subrayar que la salida del motor de recuperación es un bloque de contexto autocontenido, legible y trazable, y no un artefacto sintáctico que el modelo deba interpretar como datos en bruto.

19.5 Un ejemplo ilustrativo: el error *off-by-one*

Quiero aterrizar la cadena completa en un caso concreto y reconocible, el clásico error *off-by-one*, paradigma de los fallos que un análisis puramente sintáctico difícilmente diagnostica y que en cambio un grafo conceptual contextualiza con naturalidad. Considérese el fragmento siguiente, en el que un estudiante pretende recorrer e imprimir todos los elementos de una lista:

lista = [10, 20, 30]

for i in range(len(lista) + 1):

print(lista[i])

El programa es sintácticamente correcto y supera la compilación sin objeciones, pero falla en tiempo de ejecución con un `IndexError` en la última iteración. Cuando `i` alcanza el valor 3, la expresión `lista[3]` accede a una posición inexistente, pues los índices válidos de una lista de tres elementos van de 0 a 2. El `+ 1` superfluo en `range` provoca una sola iteración de más, el desplazamiento por uno que da nombre al error.

Siguiendo la arquitectura descrita, el analizador AST detecta el patrón (un bucle `for` que itera sobre `range(len(…))` con una desviación aritmética en el límite y un acceso indexado en el cuerpo) y produce una verbalización del problema. Esa descripción se proyecta con `nomic-embed-text` al espacio de embeddings, y la búsqueda top-k recupera como conceptos ancla, con alta similitud del coseno, nociones como la indexación de secuencias, el rango de índices válidos y el patrón de iteración sobre `range(len(…))`. A partir de esas semillas, la expansión SPARQL (ejecutada de forma idéntica tanto si el grafo reside en `rdflib` como en GraphDB) recorre el subgrafo y recupera la vecindad conceptual, formada por el concepto de longitud de una colección, su relación con el índice máximo válido (que es la longitud menos uno), el tipo de error catalogado «confusión entre longitud e índice», la jerarquía que sitúa estos conceptos bajo la noción general de acceso a colecciones, y el patrón idiomático preferente de iterar directamente sobre los elementos (`for elemento in lista`) en lugar de hacerlo por índice. La inferencia OWL-RL contribuye aquí enlaces que no estaban afirmados explícitamente, como la subordinación transitiva de estos conceptos a categorías más abstractas. La Figura 41 reconstruye ese subgrafo conceptual y distingue las anclas semánticas (top-k) de la vecindad expandida por SPARQL.

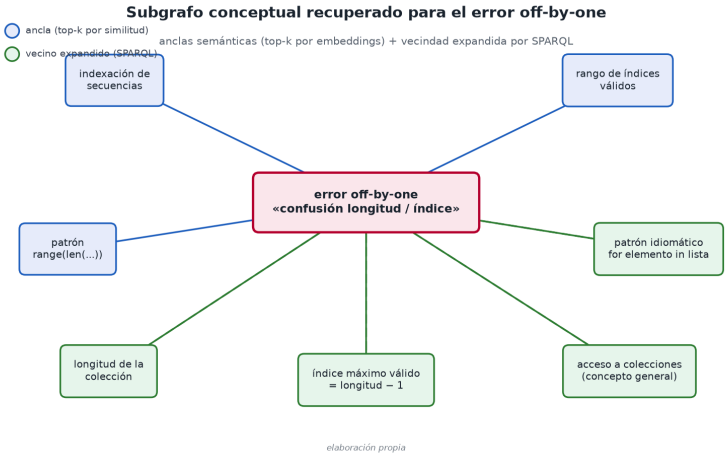

Figura 41. Subgrafo recuperado para el error off-by-one: anclas + vecindad expandida. *Fuente: elaboración propia.*

El subgrafo resultante se serializa verbalizándolo, de manera que no se entrega al modelo la tripleta `pyr:errorOffByOne pyedu:relacionadoCon pyr:indiceMaximo`, sino una explicación legible de que el error nace de confundir la longitud de la lista con su índice máximo accesible, que ambos difieren en uno, y que el límite correcto del `range` es `len(lista)` sin incremento. Acompañan a esta verbalización los marcadores de procedencia que vinculan cada afirmación con su nodo de origen en el grafo. Con este contexto, el LLM no se limita a corregir el síntoma («cambia `+ 1` por nada») sino que dispone del andamiaje conceptual para explicar *por qué* falla, conectar el fallo con el modelo mental erróneo que lo subyace y sugerir el idioma de iteración más seguro. Ofrece así un feedback formativo en el sentido de Hattie y Timperley (2007) en lugar de una mera enmienda mecánica.

Este ejemplo permite apreciar el valor diferencial de la recuperación de subgrafos respecto de un RAG textual convencional, y conecta directamente con los resultados reportados en el capítulo de validación. Allí se observa que la configuración con GraphRAG (Sistema C) mejora de forma apreciable la identificación del concepto subyacente (acierto de concepto de 0,18 en el modelo base frente a 0,48 con GraphRAG) y la trazabilidad del feedback (1,72 frente a 2,92 sobre 5), las dimensiones que el subgrafo está diseñado para nutrir. Que estas ganancias sean complementarias a las que aporta el fine-tuning en la clasificación del error (donde el Sistema B destaca) es lo que justifica la exploración del Sistema D híbrido, que combina *fine tuning* y GraphRAG. La evaluación de ese sistema híbrido ya se ha completado con la muestra ampliada y confirma la hipótesis. El Sistema D sintetiza la complementariedad entre B y C, pues gana o empata en las siete dimensiones evaluadas (es el mejor en seis) y se sitúa por delante de B y C, que a su vez superan al modelo base (D > {B, C} > A). En concreto, el Sistema D alcanza un acierto de concepto de 0,54 y una trazabilidad de 3,16 sobre 5, y recoge así tanto la identificación conceptual que aporta el grafo como la clasificación que aporta el *fine tuning*. Procede, no obstante, mantener la cautela metodológica que recorre toda la memoria. Estos resultados proceden de una evaluación automática sobre 50 casos held-out con un juez LLM y, aun siendo el Sistema D el mejor de los cuatro, su mejor acierto de categoría se queda en 0,76, de modo que ninguno de los objetivos cuantitativos del anteproyecto (que fijaban umbrales como un 85 % de acierto de categoría) se da por verificado; la validación por un panel docente del subgrafo recuperado constituye trabajo futuro.

20 Afinado eficiente del Sistema B con QLoRA

El segundo de los cuatro sistemas que se contrastan en este trabajo (el que en la nomenclatura del estudio aparece designado como Sistema B) responde a una pregunta distinta de la que motivó la arquitectura RAG. Mientras que los sistemas augmentados con el grafo (C y D) apuestan por inyectar conocimiento estructurado en el contexto del modelo en tiempo de inferencia, el Sistema B explora la vía complementaria. En lugar de aportar el conocimiento desde fuera, modifico los pesos del propio modelo de lenguaje para que aprenda a producir, por sí mismo, el tipo de feedback diagnóstico que el dominio educativo requiere. En este capítulo narro cómo he llevado a cabo ese *fine tuning*, qué técnicas lo han hecho viable en una única tarjeta gráfica de consumo, qué dificultades (singularmente el sobreajuste) surgieron por el camino y cómo las mitigué y, sobre todo, con qué cautelas debe leerse el resultado. El *fine tuning* funcionó, mejoró de forma medible la capacidad del modelo para clasificar errores de programación, pero lo hizo sobre un dataset sintético y con una muestra de validación pequeña, y prefiero no confundir una mejora real y acotada con una validación robusta y generalizable.

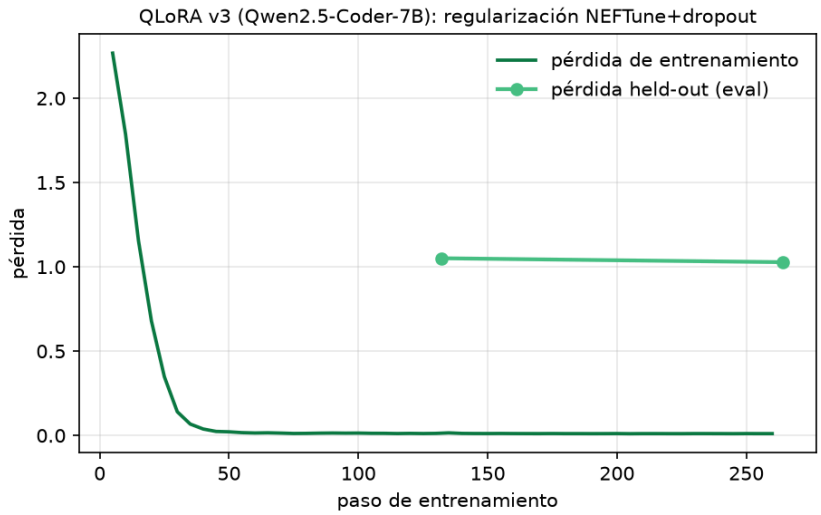

Figura 42. Curvas de pérdida de entrenamiento y held-out (QLoRA v3): la regularización invierte el sobreajuste (1,051 → 1,028). *Fuente: elaboración propia a partir de `loss_history.json` (entrenamiento QLoRA v3).*

20.1 Por qué afinar y por qué con QLoRA

El *fine tuning* completo de un modelo de lenguaje masivo (reajustar todos sus parámetros mediante descenso de gradiente sobre un corpus específico) es, para un modelo de siete mil millones de parámetros, una operación que excede con holgura los recursos de un trabajo de fin de máster realizado sobre hardware de consumo. Almacenar el modelo en precisión completa, junto con los estados del optimizador y los gradientes, exige decenas de gigabytes de memoria de vídeo que ninguna tarjeta doméstica ofrece. Durante años, esta barrera relegó el *fine tuning* de modelos grandes a quienes disponían de clústeres de aceleradores, y dejó a la inmensa mayoría de investigadores y desarrolladores la única opción de consumir modelos preentrenados tal cual venían. La irrupción de las técnicas de *fine tuning* eficiente en parámetros (y, dentro de ellas, de LoRA y su variante cuantizada QLoRA) cambió ese panorama de manera sustancial. Es esa posibilidad la que el Sistema B aprovecha.

La idea de LoRA (*Low-Rank Adaptation*), formulada por Hu et al. (2021), parte de una observación tan sencilla como fértil. Cuando se adapta un modelo preentrenado a una tarea nueva, la actualización que sufren sus matrices de pesos tiende a tener un rango intrínseco bajo. Dicho de otro modo, no hace falta modificar libremente cada uno de los millones de parámetros de una capa para especializarla; basta con aprender una corrección de rango reducido, expresable como el producto de dos matrices pequeñas, que se suma a la matriz original. Durante el *fine tuning*, los pesos preentrenados permanecen congelados y solo se entrenan esas matrices de baja dimensión, lo que reduce el número de parámetros entrenables en varios órdenes de magnitud. El resultado es un *fine tuning* mucho más ligero en memoria y en

cómputo, y unos adaptadores compactos que pueden almacenarse y combinarse con el modelo base sin alterarlo de forma irreversible.

QLoRA (Dettmers et al., 2023) lleva esta economía un paso más allá al combinar LoRA con la cuantización del modelo base a cuatro bits. En lugar de mantener los pesos congelados en precisión de dieciséis o treinta y dos bits, QLoRA los almacena en un formato cuantizado de cuatro bits[29] y realiza el cómputo de los gradientes solo a través de los adaptadores LoRA, que sí se mantienen en precisión superior. La cuantización a nf4, junto con técnicas auxiliares como la doble cuantización de las constantes de escala y la paginación de los estados del optimizador, comprime drásticamente la huella de memoria del modelo base sin que la calidad del *fine tuning* se resienta de forma apreciable. La consecuencia práctica, demostrada por Dettmers et al. (2023), es que modelos que antes requerían infraestructura de servidor pueden afinarse en una sola tarjeta gráfica. Es esa democratización la que hace posible el Sistema B de este trabajo.

La configuración concreta adoptada sigue de cerca las recomendaciones de ambos trabajos. Como modelo base se eligió Qwen2.5-Coder-7B-Instruct, una variante de siete mil millones de parámetros especializada en código y ya instruida para seguir indicaciones, lo que la convierte en un punto de partida natural para un sistema cuyo cometido es razonar sobre fragmentos de programas. La cuantización se fijó en cuatro bits con el tipo nf4. Los adaptadores LoRA se configuraron con un rango r=16 y un factor de escala alpha=32 (una relación alpha/r de dos que es habitual en la práctica y que equilibra la capacidad de adaptación con la estabilidad del entrenamiento), aplicados a las proyecciones de las capas de atención y de las redes de avance. Todo el *fine tuning* se ejecutó sobre una única tarjeta NVIDIA RTX 5090, un hardware de consumo de gama alta cuya memoria de vídeo, holgada para los estándares domésticos pero modesta frente a un acelerador de centro de datos, resultó más que suficiente gracias a la economía de QLoRA. Subrayo este punto porque encaja con la filosofía general del proyecto. Igual que el sistema de inferencia se despliega íntegramente en local para preservar la soberanía sobre los datos del estudiante, también el *fine tuning* se realiza sin recurrir a infraestructura externa, lo que mantiene todo el ciclo (datos, entrenamiento e inferencia) bajo control propio.

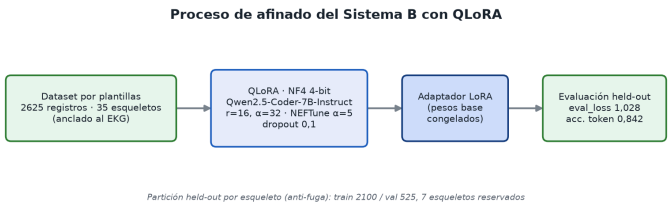

Figura 43. Proceso de afinado del Sistema B. El dataset por plantillas anclado al EKG (2625 registros, 35 esqueletos) alimenta un afinado QLoRA NF4 de 4 bits sobre Qwen2.5-Coder-7B-Instruct (r=16, α=32, NEFTune α=5, dropout 0,1); el adaptador resultante se evalúa sobre el conjunto held-out (eval_loss 1,028 y precisión de token 0,842). *Fuente: elaboración propia a partir de este capítulo y de `eval_final.json` (qlora-feedback-v3).*

20.2 El dataset sintético: plantillas ancladas al grafo

Un *fine tuning* solo es tan bueno como los datos sobre los que se entrena, y aquí afronté una dificultad inherente al dominio. No se dispone de un corpus amplio y etiquetado de código de estudiantes con diagnósticos de error validados por docentes. Reunir tal corpus habría exigido recopilar entregas reales, anonimizarlas, anotarlas con expertos y resolver los problemas éticos y de privacidad que ello acarrea (un esfuerzo que excede el alcance de este trabajo y que, de hecho, forma parte de su trabajo futuro). Ante esa carencia, opté por generar un dataset sintético mediante plantillas, una decisión pragmática que condiciona la interpretación de todos los resultados posteriores. La composición del corpus resultante se resume en la Figura 44.

[29] el tipo de datos NormalFloat 4-bit, nf4, diseñado específicamente para pesos cuyos valores siguen una distribución aproximadamente normal

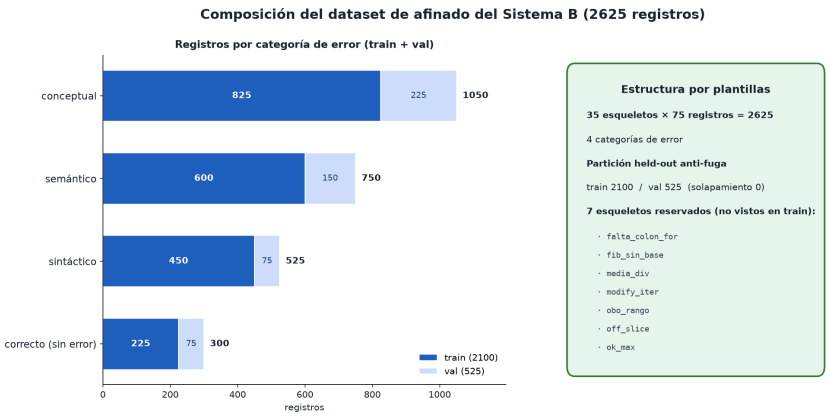

Figura 44. Composición del dataset de feedback (2625 registros, 35 esqueletos, 4 categorías).

La generación sintética no se hizo, sin embargo, al azar ni desconectada del resto del sistema. La clave del enfoque, y lo que a mi juicio le da legitimidad metodológica dentro de sus límites, es que las plantillas están ancladas al grafo de conocimiento educativo. Cada ejemplo de entrenamiento se construye a partir de un concepto y un error típico representados en el EKG, de modo que el repertorio de fragmentos de código defectuosos y de diagnósticos asociados no es una invención arbitraria, sino una proyección del conocimiento que el grafo ya modela de forma estructurada. Un error de confusión entre asignación y comparación, una iteración con índice fuera de rango, una recursión sin caso base, una mutación inadvertida de una estructura compartida; cada uno de estos patrones, presentes en el grafo como errores conceptuales recurrentes vinculados a sus conceptos correspondientes, da lugar a una familia de ejemplos sintéticos en los que el código manifiesta el error y el diagnóstico de referencia lo nombra, lo clasifica y lo explica. Así, el dataset hereda del EKG una cobertura controlada del dominio y una coherencia entre lo que el modelo aprende a diagnosticar y lo que el grafo afirma sobre la enseñanza de la programación.

Este anclaje tiene una virtud adicional que quiero resaltar, y es que alinea el Sistema B con el resto de la arquitectura. Como el dataset de *fine tuning* y el grafo que alimenta los sistemas RAG comparten la misma ontología de conceptos y errores, los cuatro sistemas del estudio operan sobre un vocabulario diagnóstico común, lo que hace que su comparación sea más limpia y que las diferencias observadas reflejen las virtudes de cada aproximación (conocimiento internalizado en los pesos frente a conocimiento recuperado en contexto) y no discrepancias en la taxonomía de partida. Aun así, no quiero sobrevender el anclaje. Que las plantillas deriven del grafo garantiza coherencia y pertinencia, pero no convierte a los ejemplos sintéticos en código real de estudiantes, al que les falta la variabilidad, el ruido, las soluciones parciales y las idiosincrasias que caracterizan las entregas auténticas. El modelo afinado aprende a diagnosticar el tipo de errores que las plantillas instancian, en las formas en que las plantillas los instancian, y esa es una limitación de fondo que ninguna sofisticación del entrenamiento puede subsanar y que solo una validación con datos reales podría despejar.

Quiero dejar constancia explícita de una decisión que considero capital para interpretar correctamente todo lo que sigue, y es que los ejemplos de entrenamiento se generaron con plantillas deterministas ancladas al grafo, y en ningún momento mediante un modelo de lenguaje generativo. No recurrí a un modelo grande para fabricar el código defectuoso ni los diagnósticos de referencia. Cada ejemplo es la instanciación reglada de una plantilla que parte de un concepto y un error tipificados en el EKG, sin ningún paso de generación neuronal por el medio. Tomé esta vía por tres razones que se refuerzan entre sí. La primera es el control. Una plantilla determinista me garantiza que cada caso ilustra el error que pretendo cubrir y que su diagnóstico de referencia es correcto por construcción, sin el riesgo de que un generador introduzca etiquetas equivocadas, errores espurios o explicaciones plausibles pero infundadas. La segunda es la trazabilidad. Al derivar del grafo, cada ejemplo conserva un vínculo nítido con el concepto y el error del EKG de los que procede, lo que permite saber en todo momento qué parte

del dominio cubre el dataset y auditar su composición; un dataset destilado de un modelo generativo perdería esa procedencia y mezclaría el conocimiento del grafo con el conocimiento, no siempre fiable, del modelo que lo genera. La tercera es la reproducibilidad. Una generación determinista a partir del grafo y de un conjunto fijo de plantillas reconstruye exactamente el mismo dataset cuantas veces se ejecute, mientras que un dataset producido por un modelo generativo dependería de su versión, de su temperatura de muestreo y de un comportamiento que no es estable en el tiempo, lo que comprometería la replicabilidad del experimento. La contrapartida de esta elección es conocida y la asumo sin matices. Un dataset así corre el riesgo de contener regularidades superficiales que el modelo podría aprender a reproducir en lugar de la competencia diagnóstica subyacente.[30] Es una limitación real, pero acotable, y la mitigo con la partición held-out por esqueleto que describo a continuación, que reserva para validación plantillas estructurales nunca vistas en entrenamiento y obliga así a que las métricas midan transferencia a estructuras nuevas y no mera reproducción de las regularidades aprendidas.

20.3 Enmascarado del prompt y separación por esqueleto

Dos decisiones técnicas en la preparación de los datos merecen una explicación detenida, porque atañen directamente a la validez de la evaluación. La primera es el enmascarado del prompt. Cada ejemplo de entrenamiento consta de dos partes, una entrada (el fragmento de código y la solicitud de diagnóstico) y una salida (el diagnóstico de referencia que se desea que el modelo aprenda a generar). Si se entrenara el modelo para predecir indistintamente todos los tokens de la secuencia, incluidos los del prompt, una fracción del gradiente se gastaría en enseñarle a reproducir el enunciado de la tarea, algo que no aporta nada al objetivo y que diluye la señal de aprendizaje. Apliqué, por ello, el enmascarado del prompt. En el cálculo de la pérdida solo se computan los tokens de la respuesta, mientras que los del prompt se marcan para ser ignorados. Así, todo el esfuerzo de optimización se concentra en lo que de verdad importa, que es producir el diagnóstico correcto a partir del código dado. Es una práctica estándar en el *fine tuning* de modelos instruidos, pero la declaro porque no es neutra respecto de los resultados. Sin ella, las métricas de pérdida y de precisión de token no serían comparables ni interpretables del mismo modo.

La segunda decisión, y la más importante para la integridad del estudio, es la separación de los datos en entrenamiento y validación mediante un *split* mantenido aparte (*held-out*) por esqueleto. Aquí el término «esqueleto» designa la plantilla estructural subyacente de la que se derivan los ejemplos concretos. El riesgo evidente de un dataset generado por plantillas es la fuga de información. Si ejemplos derivados de la misma plantilla aparecen tanto en entrenamiento como en validación, el modelo puede memorizar la estructura de la plantilla durante el entrenamiento y obtener después una puntuación engañosamente alta sobre ejemplos de validación que, pese a diferir en los detalles superficiales, comparten su esqueleto. Una evaluación así mediría la capacidad de memorización, no la de generalización, y sería radicalmente engañosa. Para evitarlo, la partición se hizo por esqueleto, de modo que los esqueletos de plantilla reservados para validación no aparecen en absoluto durante el entrenamiento, de modo que cuando el modelo se evalúa lo hace sobre estructuras de error que nunca ha visto en esa forma. Esta separación anti-fuga es la única manera de que las métricas de validación signifiquen algo, y su adopción condiciona la lectura de todas las cifras que siguen. Cuando más abajo se hable de pérdida o de precisión sobre datos mantenidos aparte, debe entenderse que se trata de esqueletos genuinamente nuevos para el modelo.

20.4 El sobreajuste y su diagnóstico

El primer run de *fine tuning*, ejecutado sin medidas explícitas de regularización más allá de las que la propia configuración de LoRA introduce, ofreció una lección clásica y temprana sobre los peligros del *fine tuning* con datos escasos. La señal fue inequívoca y aparece en la evolución de la pérdida sobre el conjunto mantenido aparte a lo largo de las épocas, con 1.297 en la primera época, 1.380 en la segunda y 1.410 en la tercera. La interpretación de esta secuencia no admite mucha ambigüedad. Que

[30] patrones léxicos y estructurales repetidos que el conjunto finito de plantillas imprime en todos sus ejemplos

la pérdida de validación, lejos de descender o estabilizarse, crezca de forma monótona época tras época es la firma característica del sobreajuste. El modelo mejora su ajuste a los ejemplos de entrenamiento (memorizándolos, en buena medida) a costa de empeorar su comportamiento sobre los ejemplos que no ha visto. Cada época adicional, en lugar de hacerlo más competente, lo hacía más rígido y peor adaptado a esqueletos nuevos.

Este resultado, aunque indeseable, fue valioso por dos razones. La primera es que el hecho mismo de poder detectarlo confirma que la separación por esqueleto cumplía su función. Si la validación hubiera compartido plantillas con el entrenamiento, el sobreajuste se habría disfrazado de mejora y habría pasado inadvertido. Ver crecer la pérdida de validación es, paradójicamente, una buena noticia metodológica, porque significa que la métrica estaba midiendo generalización y no memorización. La segunda razón es que el diagnóstico era accionable. Un sobreajuste tan claro, con una muestra de entrenamiento pequeña como la disponible, apunta directamente hacia la regularización y hacia el control del número de épocas como remedios. No se trataba de un fallo de la arquitectura ni de una incapacidad del modelo para la tarea, sino de un exceso de ajuste que las técnicas adecuadas podían contener. Quiero ser explícito sobre la causa de fondo. Con un dataset sintético y una muestra de validación reducida, la tendencia al sobreajuste no es una anomalía, sino el comportamiento esperable, y combatirla no lo elimina como riesgo latente, solo lo atenúa hasta donde los datos permiten. La homogeneidad interna de cada esqueleto, muy superior a la diversidad entre esqueletos distintos, refuerza este diagnóstico (Figura 45).

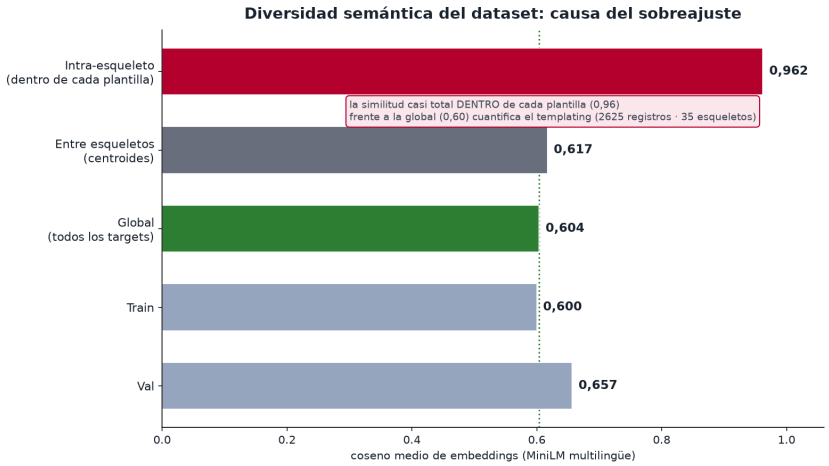

Figura 45. Diversidad por coseno: intra-esqueleto 0,962 vs global 0,604.

20.5 Mitigación mediante regularización

A la vista del diagnóstico, el segundo run incorporó dos mecanismos de regularización complementarios. El primero es NEFTune, una técnica que inyecta ruido aleatorio controlado en los *embeddings* de entrada durante el *fine tuning*, con un parámetro de intensidad fijado en alpha=5. La intuición que la sustenta es que perturbar ligeramente las representaciones de entrada impide que el modelo se ancle en los detalles superficiales y exactos de los ejemplos de entrenamiento y lo fuerza a aprender regularidades más robustas que toleran pequeñas variaciones; en términos de generalización, el ruido actúa como un suavizado que reduce la tendencia a memorizar. El segundo mecanismo es el *dropout* aplicado a los propios adaptadores LoRA, con una tasa de 0.1, que durante el entrenamiento desactiva aleatoriamente una fracción de las conexiones de los adaptadores y obliga así a que la adaptación no dependa en exceso de ninguna ruta concreta. Ambas técnicas son baratas, bien establecidas y especialmente indicadas para escenarios de datos limitados como el que aquí nos ocupa. La configuración completa de este *fine tuning* regularizado se recoge en la Figura 46.

Configuración de afinado QLoRA del Sistema B (v3)

Parámetro	Valor
Modelo base	Qwen2.5-Coder-7B-Instruct
Técnica	QLoRA · cuantización NF4 4-bit
LoRA · rango (r)	16
LoRA · alpha (α)	32
LoRA · dropout	0,100
Regularización ruido	NEFTune α = 5
Épocas	2
GPU	NVIDIA RTX 5090 (Blackwell, sm_120)
Partición	held-out por esqueleto · train 2100 / val 525
eval_loss (ép. 1 → ép. 2)	1,051 → 1,028
Precisión de token (held-out)	0,842

Fuente: adapter_config.json · eval_final.json · loss_history.json (qlora-feedback-v3) — elaboración propia

Figura 46. Configuración del fine-tuning QLoRA v3.

El efecto de esta regularización sobre la pérdida de validación fue claro y cuantificable, y queda recogido en la Figura 42 y en la comparación de la Figura 47. Frente a la secuencia ascendente del primer run, el run regularizado mostró una pérdida sobre datos mantenidos aparte de 1.051 que descendía hasta 1.028 e invirtió por completo la dinámica patológica anterior. Donde antes la validación empeoraba época tras época, ahora mejoraba, lo que indica que el modelo estaba aprendiendo regularidades transferibles a esqueletos no vistos en lugar de memorizar los de entrenamiento. La magnitud absoluta de la mejora (de un entorno de 1.3 a 1.4 a un entorno de 1.03 a 1.05) y, más aún, el cambio de signo de la tendencia confirman que la regularización atacó la raíz del problema. No quiero, sin embargo, presentar estos números como una victoria rotunda. Una pérdida de validación en torno a 1.03 es un valor razonable para esta tarea y esta escala, pero no es un indicador de excelencia absoluta, y su interpretación está acotada por el carácter sintético de los datos y por el tamaño reducido del conjunto de validación. Lo que las cifras permiten afirmar es que la regularización transformó un *fine tuning* que se deterioraba en uno que generalizaba, no que el modelo resultante alcanzara un dominio definitivo de la tarea.

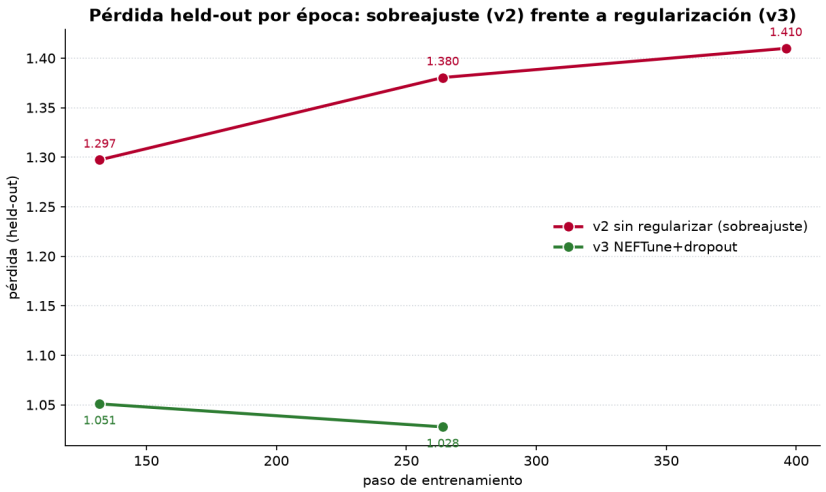

Figura 47. Pérdida held-out por época en los dos runs de afinado. El run v2 sin regularizar asciende de 1,297 a 1,410 (sobreajuste); el run v3 con NEFTune y dropout desciende de 1,051 a 1,028. *Fuente: elaboración propia a partir de los `loss_history.json` y `trainer_state.json` de qlora-feedback-v2 y qlora-feedback-v3.*

Como métrica complementaria a la pérdida, se midió también la precisión de token sobre el conjunto mantenido aparte, que se situó en 0.842. Esta cifra indica que, sobre esqueletos nuevos, alrededor del ochenta y cuatro por ciento de los tokens del diagnóstico de referencia se predicen correctamente. Es una métrica útil porque ofrece una lectura más interpretable que la pérdida (habla directamente de

aciertos en la generación) pero la leo con la misma cautela. La precisión de token premia la coincidencia literal token a token y, en un dataset derivado de plantillas, una parte de esa coincidencia procede de la estructura compartida del lenguaje diagnóstico, no necesariamente de un razonamiento profundo sobre el error. Un valor de 0.842 es consistente con un modelo que ha aprendido a producir diagnósticos bien formados y mayoritariamente correctos sobre el tipo de errores que el grafo modela, pero no debe extrapolarse a una precisión diagnóstica sobre código real ni equipararse con las métricas pedagógicas que la evaluación A/B/C examina por separado.

20.6 Lectura del resultado y cautelas

Llegados a este punto, es imprescindible situar el *fine tuning* del Sistema B dentro del marco de cautela que preside todo el trabajo. El anteproyecto fijó como expectativa una precisión diagnóstica superior al ochenta y cinco por ciento, y sería tentador presentar la precisión de token de 0.842 como una cifra que casi alcanza ese umbral. Sería, además, profundamente engañoso. La precisión de token no es la precisión diagnóstica, sino que mide coincidencia léxica sobre datos sintéticos, no acierto pedagógico sobre entregas reales evaluadas por docentes. El target del ochenta y cinco por ciento era una hipótesis a contrastar, y este *fine tuning* no lo contrasta ni lo cumple; simplemente reporta una métrica técnica del entrenamiento que no debe confundirse con aquel objetivo. Mantener esta distinción no es una cautela retórica, sino una exigencia de integridad, porque confundir ambas cosas inflaría artificialmente los logros del sistema.

Dicho lo anterior, el *fine tuning* sí produjo una mejora real y verificable, que se manifiesta no en las métricas internas del entrenamiento sino en la evaluación comparada con juez automático sobre cincuenta casos mantenidos aparte. En esa evaluación, el Sistema B destaca con claridad en el acierto de categoría del error, donde alcanza 0.70 frente al 0.26 del modelo base sin augmentar (Sistema A), y en la identificación del problema en la escala de uno a cinco, donde obtiene 3.80 frente al 2.04 del modelo base. Es decir, el *fine tuning* dotó al modelo de una competencia genuina para clasificar el tipo de error de programación y para explicar técnicamente qué ocurre en el código, y superó con holgura al modelo de partida en esas dimensiones concretas. Esa es la contribución real del Sistema B, y es coherente con lo que cabía esperar de internalizar en los pesos del modelo el repertorio de errores del grafo. El modelo aprendió a reconocer y nombrar esos errores con notable solvencia.

Esa misma evaluación revela, sin embargo, los límites del enfoque y motiva el resto de la arquitectura. El Sistema B no mejora la trazabilidad del feedback (donde queda por detrás de los sistemas augmentados con el grafo), porque el conocimiento internalizado en los pesos no deja, por su naturaleza, marcadores explícitos de procedencia que el estudiante o el docente puedan inspeccionar. Y en el acierto de concepto subyacente queda por debajo del sistema RAG, que recupera del grafo el concepto pertinente de forma explícita. Esta complementariedad (el Sistema B sobresale en clasificar y explicar el error, el Sistema C sobresale en identificar el concepto y en aportar trazabilidad) es lo que justifica el Sistema D híbrido, que combina el modelo afinado con la augmentación por grafo para reunir las virtudes de ambos. El *fine tuning*, en suma, no es un fin en sí mismo, sino una de las dos piezas cuya conjunción se hipotetiza como superior.

Resta enumerar las cautelas que cualquier lector debe tener presentes al interpretar este capítulo. El *fine tuning* se realizó sobre un dataset sintético generado por plantillas, lo que significa que el modelo ha sido entrenado y evaluado sobre una proyección controlada del dominio, no sobre código auténtico de estudiantes con toda su variabilidad. La validación se hizo sobre una muestra pequeña, tanto en el conjunto mantenido aparte del entrenamiento como en los cincuenta casos de la evaluación comparada, de modo que las cifras tienen el valor de indicios consistentes, no de resultados estadísticamente robustos. La separación por esqueleto protege contra la fuga de información dentro del marco sintético, pero no garantiza la transferencia a esqueletos de error que las plantillas no contemplan en absoluto. Y el sobreajuste, aunque mitigado con éxito por la regularización, sigue siendo un riesgo latente inherente al *fine tuning* con datos escasos, que solo una ampliación del corpus (idealmente con datos reales)

podría conjurar de manera definitiva. La validación con docentes humanos, el cálculo de la concordancia inter-juez y la evaluación sobre entregas reales que el anteproyecto contemplaba quedan, también para el Sistema B, como trabajo futuro. Lo que este capítulo afirma, y solo eso, es que mediante QLoRA fue posible afinar un modelo de siete mil millones de parámetros en una sola tarjeta de consumo, que ese *fine tuning* sufrió un sobreajuste claramente diagnosticado y eficazmente mitigado con regularización, y que el modelo resultante adquirió una competencia real, aunque acotada, para clasificar y explicar errores de programación. Ni más, ni menos.

21 Los cuatro sistemas comparados (A/B/C/D)

La pregunta de investigación que vertebra este Trabajo de Fin de Máster (si la integración de un grafo de conocimiento educativo con un modelo de lenguaje masivo mediante una arquitectura RAG mejora la calidad pedagógica de la retroalimentación frente al modelo sin augmentación, hipótesis H1) no admite una respuesta válida si se compara únicamente un sistema completo contra un modelo base. Una comparación de ese tipo confundiría dos intervenciones distintas que la arquitectura propuesta entrelaza. La primera es la adaptación del propio modelo de lenguaje al dominio mediante *fine tuning* supervisado; la segunda, el enriquecimiento de su contexto con conocimiento estructurado recuperado del grafo. Si midiera el sistema final contra el modelo base y observara una mejora, no podría atribuirla con fundamento a una u otra causa, ni discernir si ambas contribuyen, si una de ellas es prescindible o si interactúan. Por esta razón organicé la fase de validación experimental (Fase III de la metodología) como un **diseño factorial 2×2** que cruza dos factores binarios (la presencia o ausencia de fine-tuning y la presencia o ausencia de recuperación sobre el grafo) y da lugar a cuatro configuraciones que denomino Sistema A, B, C y D. Este capítulo define con precisión cada una de ellas, explicita qué variable aísla cada celda del diseño y justifica por qué esta disposición factorial es la condición necesaria para atribuir efectos a sus causas.

21.1 El problema de la atribución y la lógica factorial

Un diseño factorial no es un mero refinamiento estadístico, sino una exigencia derivada de la propia naturaleza de la hipótesis. La arquitectura que propongo combina dos mecanismos heterogéneos que actúan sobre el modelo de lenguaje por vías completamente distintas. El **fine-tuning** modifica los pesos del modelo (o, en mi caso, un conjunto reducido de parámetros añadidos mediante adaptadores de bajo rango) para reorientar su comportamiento hacia el dominio y la tarea, con lo que el conocimiento adaptado queda *internalizado* en el modelo. La **augmentación por recuperación** (RAG), en cambio, no altera el modelo en absoluto. Deja intactos sus pesos y se limita a inyectar conocimiento *externo* dentro del contexto de cada consulta, recuperado dinámicamente del grafo en función del código concreto que se evalúa (Lewis et al., 2020). Son, por tanto, dos formas radicalmente diferentes de aportar conocimiento de dominio a un LLM, una que lo cose en la red y otra que lo presenta como contexto. Que persigan un fin parecido (mejorar el feedback) no implica que actúen sobre las mismas dimensiones de la calidad ni que sus efectos se sumen de manera trivial.

Ante dos intervenciones de esta índole, la inferencia causal exige separar sus contribuciones. La lógica del diseño factorial consiste en manipular cada factor de forma independiente y observar el sistema en todas las combinaciones posibles de sus niveles, de manera que el efecto de cada factor pueda estimarse comparando las celdas que solo difieren en ese factor (efecto principal) y, lo que resulta especialmente relevante aquí, pueda detectarse si la presencia de un factor modifica el efecto del otro (interacción). En mi caso, con dos factores binarios, las combinaciones son cuatro: ninguno de los dos mecanismos (A), solo fine-tuning (B), solo grafo (C) y ambos (D). Si únicamente dispusiera de A y D (el extremo desnudo y el extremo completo) estaría ante un *diseño confundido* en el sentido técnico del término, porque las dos variables cambiarían a la vez entre ambos sistemas y ningún razonamiento posterior podría desenmarañar cuánta de la diferencia observada corresponde a cada una. La inclusión deliberada de las celdas intermedias B y C es lo que convierte una comparación anecdótica en un experimento interpretable.

En forma de cuadro, el diseño se dispone así, con las cuatro celdas que resultan de cruzar los dos factores binarios:

	Sin grafo (RAG ausente)	Con grafo (GraphRAG)
Sin fine-tuning	**Sistema A** — modelo base, línea de fondo	**Sistema C** — GraphRAG, conocimiento externo
Con fine-tuning	**Sistema B** — modelo afinado (QLoRA)	**Sistema D** — híbrido (afinado + grafo)

Fuente: elaboración propia (diseño factorial 2×2 descrito en esta sección).

Esta decisión tiene además una consecuencia que afecta a la integridad científica del trabajo. El anteproyecto formuló la hipótesis H1 desglosada en cuatro sub-hipótesis: precisión diagnóstica (H1a), reducción de alucinaciones (H1b), utilidad formativa (H1c) y trazabilidad (H1d), con la expectativa, ambiciosa, de que la integración EKG+LLM superara holgadamente al modelo base en todas ellas. El diseño factorial permite contrastar esa hipótesis de forma matizada en lugar de aceptarla o rechazarla en bloque, pues es perfectamente posible que un factor mejore unas dimensiones y otro factor mejore otras distintas, con lo que la respuesta a H1 no sería un sí o un no rotundos, sino un mapa de qué interviene sobre qué. Como se verá, eso es lo que la evidencia recogida sugiere, y solo un diseño que aísle los factores permite leerlo así sin caer en la tentación de atribuir todo el mérito a la arquitectura completa.

21.2 Sistema A: el modelo base como línea de fondo

El Sistema A constituye la **línea de fondo** (*baseline*) contra la que se mide todo lo demás. Consiste en el modelo de lenguaje `llama3.1:8b` (Grattafiori et al., 2024) desplegado localmente mediante Ollama, sin *fine tuning* alguno y sin acceso al grafo de conocimiento. Recibe el código del estudiante (tras el análisis con árboles de sintaxis abstracta y métricas de complejidad que comparte con el resto de configuraciones) y genera la retroalimentación apoyándose exclusivamente en el conocimiento que adquirió durante su preentrenamiento general. No hay, en este sistema, ningún componente específico del dominio educativo de la programación en Python más allá de lo que un modelo de propósito general de ocho mil millones de parámetros pueda haber absorbido de la web.

El Sistema A aísla, por tanto, la **capacidad del modelo de lenguaje desnudo**. Su función en el diseño es doble. En primer lugar, representa la alternativa que un docente o una institución podrían adoptar con el mínimo esfuerzo de ingeniería, esto es, descargar un modelo abierto y pedirle feedback directamente. Si las demás configuraciones no mejoraran apreciablemente sobre A, todo el aparato de grafo, razonamiento y *fine tuning* quedaría injustificado por su coste. En segundo lugar, A materializa el extremo de referencia del factor fine-tuning (nivel «ausente») y del factor grafo (nivel «ausente») simultáneamente; es la celda 00 del diseño factorial. Las limitaciones que la literatura atribuye a los LLM en este escenario (alucinaciones, ausencia de fundamentación en un modelo pedagógico explícito y escasa trazabilidad de sus afirmaciones; Bender et al., 2021) deberían manifestarse aquí con mayor crudeza, y los resultados confirman esa expectativa. A obtiene los valores más bajos en prácticamente todas las dimensiones medidas, con un acierto de categoría del error de 0,26, un acierto de concepto de 0,18, una identificación del problema de 2,04 sobre 5 y una trazabilidad de 1,72 sobre 5.

21.3 Sistema B: el modelo afinado, conocimiento internalizado

El Sistema B introduce el primer factor en su nivel «presente», el **fine-tuning**, y mantiene ausente el grafo. A diferencia de A, no emplea `llama3.1:8b` sino `Qwen2.5-Coder-7B-Instruct`, un modelo especializado en código que se afinó sobre el dominio mediante QLoRA, la técnica de *fine tuning* eficiente en parámetros que combina cuantización de cuatro bits del modelo base con adaptadores de bajo rango entrenables (Dettmers et al., 2023; Hu et al., 2021). La configuración del *fine tuning* (adaptadores LoRA de rango 16 y factor de escala alpha 32, cuantización nf4 de cuatro bits, ejecutado sobre una RTX 5090) y, sobre todo, las decisiones tomadas para combatir el sobreajuste se documentan en detalle en el capítulo dedicado al entrenamiento; aquí basta con señalar que el run sin regularizar mostró una clara degradación de la pérdida en el conjunto held-out a lo largo de las épocas (1,297, 1,380, 1,410), que la incorporación de regularización mediante NEFTune (alpha 5) y *dropout* en los adaptadores (0,1) la corrigió hasta valores de 1,051 y 1,028, y que la precisión de token sobre datos no vistos se situó en 0,842. Toda la evaluación se realizó exclusivamente sobre esqueletos held-out para evitar fugas entre entrenamiento y prueba.

El uso de un modelo distinto en B (Qwen2.5-Coder) respecto del modelo de A (Llama 3.1) introduce una asimetría que conviene aclarar. En rigor, B difiere de A en dos cosas, el modelo de partida y el *fine tuning*. Esto significa que el efecto que B muestra sobre A no es atribuible al fine-tuning en estado

puro, sino a la combinación de un modelo base especializado en código y su posterior adaptación al dominio. La elección respondió a una consideración práctica (Qwen2.5-Coder es un punto de partida mucho más sólido para tareas de programación que un modelo generalista), pero obliga a moderar la interpretación. Cuando atribuyo a B su perfil de mejoras, hablo del *tratamiento de fine tuning tal como se implementó*, no de un factor abstracto perfectamente aislado. Es una limitación del diseño que la evaluación ampliada, con un protocolo más controlado, busca acotar mejor.

Pese a esa cautela, el Sistema B aísla con claridad suficiente el efecto de **internalizar el conocimiento del dominio en los pesos del modelo**. Y su perfil de resultados es revelador. B sobresale en aquello que el *fine tuning* supervisado sobre ejemplos del dominio cabría esperar que mejorase: la *clasificación del error* y la calidad de la explicación técnica. Su acierto de categoría asciende a 0,70 (muy por encima de A) y su capacidad de identificación del problema alcanza 3,80 sobre 5, el valor más alto entre A, B y C. En cambio, su acierto de concepto (0,50) y su trazabilidad (3,00) progresan de forma más moderada en relación con su salto diagnóstico, lo cual es coherente con su naturaleza, ya que el conocimiento internalizado no aporta, por sí mismo, referencias verificables ni vínculos explícitos a un cuerpo de conceptos estructurado.

21.4 Sistema C: GraphRAG, conocimiento externo recuperado

El Sistema C invierte la combinación de B, ya que activa el factor grafo y mantiene ausente el fine--tuning. Parte del mismo modelo base que A (`llama3.1:8b` sin afinar) y le añade la cadena completa de **GraphRAG**. Ante el código del estudiante, el sistema computa los embeddings con `nomic-embed--text`, recupera del EKG el subgrafo más relevante combinando similitud semántica y expansión mediante consultas SPARQL, y construye un prompt que presenta al modelo, como contexto, los conceptos, relaciones jerárquicas y procedencias recuperados del grafo canónico (157 conceptos propios, que la inferencia OWL-RL materializa como 157 entidades `pyedu:Concepto` consultables, con sus 30 puentes a Wikidata declarados mediante `skos:exactMatch`[31] y sus relaciones N-arias reificadas). El modelo, cuyos pesos permanecen intactos, genera entonces la retroalimentación apoyándose en ese conocimiento externo estructurado en lugar de en su memoria paramétrica (Lewis et al., 2020; Edge et al., 2024).

El Sistema C aísla, así, el efecto de **fundamentar la generación en conocimiento externo verificable** sin tocar el modelo. Su valor en el diseño es central, porque encarna la tesis principal del trabajo (que un grafo educativo bien construido puede anclar y dar trazabilidad al feedback de un LLM) en su forma más pura, despojada de la interferencia del *fine tuning*. Y, de nuevo, su perfil de resultados es coherente con su mecanismo. C destaca justamente en las dimensiones que el grafo está diseñado para servir. Su acierto de concepto alcanza 0,48 (muy por encima de A y a la par del *fine tuning* B), porque la recuperación sobre el grafo le proporciona la nomenclatura precisa de los conceptos del dominio; y su trazabilidad se sitúa en 2,92 sobre 5, en el rango alto de las configuraciones sin híbrido, porque los marcadores de procedencia recuperados permiten al modelo referenciar de dónde procede cada afirmación. En contrapartida, su acierto de categoría del error (0,36) apenas supera al de A y queda muy lejos del de B, porque anclar la generación en conceptos no mejora, por sí solo, la habilidad de clasificar correctamente el tipo de fallo, que es competencia del modelo. C, en suma, mejora concepto, trazabilidad y calidad pedagógica del anclaje, pero no la destreza diagnóstica fina.

21.5 Sistema D: el híbrido, y por qué el diseño lo motiva

El Sistema D es la celda 11 del diseño factorial y combina **fine-tuning y GraphRAG** simultáneamente. Toma el modelo afinado de B (`Qwen2.5-Coder-7B` adaptado con QLoRA) y lo dota de la misma cadena de recuperación sobre el grafo de C, con lo que el modelo recibe a la vez el conocimiento internalizado en sus adaptadores y el conocimiento externo inyectado en su contexto. Es, conceptualmente, la

[31] que enlazan sin fusionar y, a diferencia de `owl:sameAs`, no propagan el tipo, de modo que las 30 entidades de Wikidata no se infieren como `pyedu:Concepto`

arquitectura completa que el anteproyecto perseguía, un modelo experto en el dominio que, además, fundamenta cada respuesta en el grafo educativo y expone su procedencia.

La justificación de D no es un mero afán de completitud del cuadro factorial, sino que emerge de manera natural de la lectura de los tres sistemas anteriores. Al separar los factores, B y C revelaron un patrón de **complementariedad**. B mejora la clasificación del error y la explicación técnica (las dimensiones del *qué* falla y *por qué* en términos de código) mientras que C mejora la identificación del concepto, la trazabilidad y la fundamentación pedagógica (las dimensiones del *qué concepto del currículo* está implicado y *de dónde* procede la afirmación). Ambos superan a A, pero en frentes distintos y en buena medida disjuntos. Esta observación, que solo es posible porque el diseño aisló cada factor, conduce a una hipótesis precisa. Si las fortalezas de B y de C residen en dimensiones diferentes, un sistema que reúna ambos mecanismos podría heredar lo mejor de cada uno. El Sistema D es la verificación experimental de esa hipótesis de complementariedad, y la tanda ampliada la confirma, pues D sintetiza las fortalezas de B y de C y resulta el mejor sistema del cuadro. Sin las celdas intermedias, D habría sido una apuesta a ciegas; con ellas, fue una predicción razonada que el experimento ha confirmado.

Aquí es donde el diseño factorial muestra su otra virtud, la capacidad de detectar **interacción**. El efecto conjunto de dos factores no tiene por qué ser la suma de sus efectos individuales. Puede ocurrir que se refuercen (sinergia, una interacción positiva), que uno sature el margen de mejora del otro (interacción negativa o subaditividad) o incluso que entren en conflicto.[32] La comparación de D contra lo que cabría esperar de la mera adición de los efectos de B y C es la estimación del término de interacción, y constituye uno de los hallazgos más informativos del trabajo. Esta cuarta celda, que en una primera tanda preliminar no se había evaluado, se midió en la tanda ampliada de 50 casos held-out con el juez `qwen2.5:32b` (Yang et al., 2024), junto a A, B y C. Los resultados confirman la hipótesis de complementariedad. El Sistema D híbrido gana o empata en las siete dimensiones medidas (y es el mejor en seis de ellas) y dibuja una ordenación clara, D por encima de B y C, y estos a su vez por encima de A. En las métricas objetivas, D alcanza un acierto de categoría del error de 0,76 y un acierto de concepto de 0,54; en las puntuaciones del juez, una identificación del problema de 4,04 sobre 5 y una trazabilidad de 3,16 sobre 5. Recoge así la destreza diagnóstica del *fine tuning* y el anclaje del grafo en un mismo sistema. El juez puntuó además tres dimensiones de estilo. En la calidad divulgativa D encabeza con 3,66 (frente a 3,52 de A, 3,38 de B y 3,20 de C) y en la calidad técnica también lidera con 3,62 (frente a 3,02 de A, 3,44 de B y 2,44 de C), mientras que la dimensión de sugerencia de mejora se reparte de forma prácticamente empatada entre los cuatro sistemas, en torno a 2,9. La interacción, lejos del conflicto que cabía temer entre el conocimiento internalizado y el contexto recuperado, resultó positiva.

La tabla siguiente reúne en un solo cuadro los valores ya comentados para las cuatro configuraciones, de modo que el patrón de complementariedad entre B y C, y la posición del híbrido D, resulten visibles de un vistazo:

[32] por ejemplo, que un modelo fuertemente afinado tienda a confiar en su conocimiento internalizado e ignore parcialmente el contexto recuperado, fenómeno plausible que la propia estructura del experimento permite vigilar

Dimensión	A	B	C	D
Fine-tuning	ausente	presente	ausente	presente
Grafo (GraphRAG)	ausente	ausente	presente	presente
Acierto de categoría del error	0,26	0,70	0,36	0,76
Acierto de concepto	0,18	0,50	0,48	0,54
Identificación del problema (sobre 5)	2,04	3,80	—	4,04
Trazabilidad (sobre 5)	1,72	3,00	2,92	3,16
Calidad divulgativa (sobre 5)	3,52	3,38	3,20	3,66
Calidad técnica (sobre 5)	3,02	3,44	2,44	3,62
Sugerencia de mejora (sobre 5)	≈2,9	≈2,9	≈2,9	≈2,9

Fuente: elaboración propia a partir de las cifras citadas en este capítulo; el guion (—) marca un valor que el texto no consigna para C (solo indica que el 3,80 de B es el más alto entre A, B y C), y «≈2,9» recoge el reparto prácticamente empatado de la sugerencia de mejora entre los cuatro sistemas.

21.6 El marco común que hace comparables las cuatro celdas

Para que un diseño factorial sea interpretable, todo lo que no es factor debe permanecer constante entre las celdas; de lo contrario, las variables ajenas se confundirían con los efectos de interés. Por ello, las cuatro configuraciones comparten un mismo armazón experimental. Todas reciben la misma entrada (el código del estudiante procesado de idéntico modo mediante el analizador AST y las métricas de complejidad de `radon`); todas se evalúan sobre los mismos casos held-out, generados sintéticamente a partir de plantillas y separados con rigor del material de entrenamiento para evitar fugas; todas se juzgan con el mismo protocolo automático, métricas objetivas y un juez LLM (`qwen2.5:32b`) que puntúa con la misma rúbrica de dimensiones; y todas se despliegan localmente, en coherencia con el compromiso de soberanía de los datos que recorre el proyecto. Las únicas dos cosas que varían deliberadamente entre celdas son los dos factores del diseño: si el modelo está afinado y si tiene acceso al grafo.

Sobre esta arquitectura experimental pesan limitaciones que conviene reiterar y que acotan el alcance de cualquier conclusión. La evaluación es **automática**, no humana. El juez es un LLM, no el panel de tres a cinco docentes que el anteproyecto preveía, panel que (junto con el cálculo del acuerdo inter-juez mediante ICC y el estudio sobre las 300 submissions reales) permanece como trabajo futuro. Los datos son **sintéticos** y la muestra es **pequeña** (50 casos held-out en la tanda completada, frente a los 18 de la tanda preliminar), lo que impide cualquier pretensión de significación estadística robusta o de generalización al aula real. Y, como se señaló, el Sistema B introduce una asimetría de modelo base respecto de A que enturbia parcialmente el aislamiento del factor fine-tuning. Ninguno de los objetivos cuantitativos ambiciosos del anteproyecto (precisión diagnóstica del 85% o superior, alucinaciones del 5% o inferiores, utilidad formativa de 4,0 sobre 5, trazabilidad del 100%) se da por cumplido; el mejor sistema, el híbrido D, llega a un acierto de categoría de 0,76, lejos aún del umbral del 85%, y ninguna configuración alcanza los demás targets. Eran hipótesis a contrastar, y lo que esta comparación de cuatro sistemas ofrece no es su confirmación sino una medición estructurada de manera que cada efecto observado pueda atribuirse, dentro de sus límites, a su causa probable. El mérito del diseño A/B/C/D no reside en producir cifras halagüeñas, sino en producir cifras *interpretables*, en garantizar que, sean cuales sean los resultados, sepa a qué se deben.

22 Módulo de explicabilidad e interfaz web (O4)

La arquitectura descrita en los capítulos precedentes recupera subgrafos del grafo de conocimiento educativo (EKG), los inyecta en el prompt y genera un feedback formativo apoyado en el conocimiento estructurado del dominio. Todo ese andamiaje, sin embargo, permanecería opaco para sus destinatarios (el docente que supervisa el sistema y, finalmente, el estudiante que recibe la corrección) si el feedback se presentara como un bloque de texto sin más, indistinguible del que produciría cualquier modelo de lenguaje sin augmentación. El cuarto objetivo del proyecto (O4) aborda esa carencia. Se trata de dotar al sistema de un módulo de explicabilidad que haga visible *por qué* el sistema dice lo que dice, *de dónde* procede cada afirmación y *qué* fragmento del conocimiento del dominio ha movilizado para diagnosticar el error. La hipótesis subyacente, alineada con la sub-hipótesis H1d sobre trazabilidad, es que un feedback cuyo fundamento puede inspeccionarse resulta más fiable y también más útil pedagógicamente, porque convierte la corrección en un objeto de estudio que profesor y aprendiz pueden interrogar.

En este capítulo describo la materialización de ese objetivo en dos planos complementarios. El primero es técnico y consiste en una interfaz web construida sobre FastAPI en el lado del servidor y sobre Cytoscape.js en el lado del cliente, que visualiza el subgrafo recuperado para cada envío con un código de color por tipo de nodo y que incorpora marcadores de procedencia sobre cada afirmación del feedback. El segundo plano es de fundamentación. Argumento que la explicabilidad no es en este sistema un añadido cosmético ni una mera funcionalidad de depuración, sino la expresión de un principio ético (el derecho de quien es evaluado a comprender el juicio que recae sobre él) que recorre el conjunto del proyecto. Cierro el capítulo con la descripción de la publicación de la propia memoria como sitio MyST, decisión que prolonga, también en el plano de la comunicación científica, la misma vocación de transparencia y reproducibilidad que anima al módulo de explicabilidad.

22.1 La explicabilidad como principio ético: el derecho a comprender

Empiezo por la justificación, porque es ella la que confiere sentido a las decisiones técnicas que vendrán después. En el contexto educativo, un sistema que evalúa el trabajo de un estudiante y le devuelve un diagnóstico ejerce, aunque sea de forma mediada y subordinada al docente, una forma de autoridad sobre el aprendiz. Esa autoridad conlleva una responsabilidad correlativa, la de poder dar cuenta de sus juicios. Cuando quien evalúa es un ser humano, esperamos que pueda explicar por qué considera incorrecto un planteamiento, qué concepto no se ha comprendido y cómo podría corregirse; un docente que se limitara a emitir veredictos inapelables sin justificación alguna estaría faltando a una norma básica de la práctica pedagógica. No veo razón para rebajar esa exigencia cuando el agente que media en la evaluación es un sistema automático. Al contrario, la opacidad característica de los modelos de lenguaje masivos, capaces de producir prosa fluida y persuasiva con independencia de su fundamento real, agrava el problema en lugar de atenuarlo, pues un feedback erróneo formulado con aplomo puede inducir a error con mayor facilidad que una corrección humana titubeante.

Esta preocupación entronca con un debate más amplio sobre los riesgos de los grandes modelos de lenguaje. Bender et al. (2021) advirtieron de que estos sistemas operan, en lo esencial, como reproductores estadísticos de patrones lingüísticos sin acceso a un modelo del significado de lo que enuncian, lo que los hace propensos a generar texto plausible pero infundado. En un contexto de evaluación formativa esa propensión es especialmente delicada, porque el estudiante carece, por definición, del conocimiento experto que le permitiría detectar un diagnóstico equivocado. Si supiera distinguir una corrección correcta de una incorrecta, probablemente no habría cometido el error en primer lugar. El derecho a comprender se convierte así en una salvaguarda, pues no basta con que el sistema acierte la mayor parte de las veces; es preciso que ofrezca al destinatario los medios para escrutar cada juicio concreto y, llegado el caso, para discrepar de él de forma fundamentada. La explicabilidad es la condición que permite mantener al estudiante (y sobre todo al docente que supervisa) como sujeto crítico frente al sistema, y no como receptor pasivo de sus veredictos.

El anteproyecto situaba la transparencia entre sus compromisos éticos explícitos, junto con la soberanía de los datos garantizada por el despliegue local, la atención a los sesgos y la preservación del rol docente. El módulo de explicabilidad es la traducción operativa de ese compromiso. Su finalidad no es persuadir al usuario de que confíe en el sistema, sino justo la contraria, darle los elementos para *desconfiar de manera informada*, para verificar si la cadena que va del código del estudiante al concepto del grafo y de éste al feedback se sostiene en cada eslabón. Por eso la arquitectura RAG sobre la que se asienta el proyecto no es solo una estrategia para mejorar la calidad del feedback, que reduce las alucinaciones al anclar la generación en conocimiento estructurado según la lógica que Lewis et al. (2020) formalizaron para la recuperación aumentada, sino también, e indisociablemente, una estrategia de explicabilidad, ya que el subgrafo recuperado constituye al mismo tiempo el insumo de la generación y la evidencia que la justifica. Hacer visible ese subgrafo equivale, así, a hacer visible el propio razonamiento del sistema.

Hay además una dimensión propiamente formativa en este planteamiento que no quiero pasar por alto. La literatura sobre feedback efectivo, y muy en particular el marco de Hattie y Timperley (2007), subraya que el feedback más valioso no es el que se limita a señalar el error, sino el que ayuda al aprendiz a situar ese error en una estructura de conocimiento: a entender qué concepto está implicado, cómo se relaciona con lo que ya sabe y qué camino conduce a la comprensión correcta. La visualización del subgrafo recuperado responde a esa lógica. Al mostrar tanto el concepto en el que reside el error como sus prerrequisitos y sus relaciones con conceptos vecinos, el sistema convierte el feedback en una pequeña cartografía del territorio conceptual que el estudiante está transitando. La explicabilidad deja entonces de ser únicamente una garantía ética para convertirse, ella misma, en un recurso pedagógico. Comprender por qué se ha cometido un error es ya parte de aprender a no cometerlo.

22.2 Arquitectura de la interfaz: FastAPI y Cytoscape.js

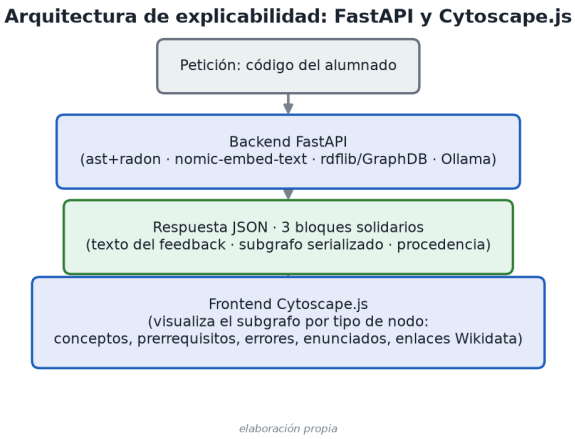

Figura 48. Cliente-servidor: FastAPI + Cytoscape.js.

En el plano técnico, el módulo se organiza según la separación habitual entre un servicio de backend, responsable de la lógica y de la exposición de datos, y un cliente de frontend, responsable de la presentación e interacción, como resume la Figura 48. Para el backend se ha optado por FastAPI, un framework de Python para la construcción de interfaces de programación de aplicaciones (API) sobre el protocolo HTTP. La elección no es casual ni meramente acomodaticia. En primer lugar, mantener el servidor en Python preserva la homogeneidad de la pila tecnológica del proyecto. El analizador de código basado en `ast` y `radon`, el cálculo de embeddings con `nomic-embed-text`, la gestión del grafo con `rdflib` y las consultas a GraphDB, y la invocación del LLM local a través de Ollama son todos componentes Python. Exponerlos mediante FastAPI evita, por tanto, introducir una frontera de lenguaje innecesaria y reduce la fricción entre la canalización de recuperación y su capa de presentación. En segundo lugar, FastAPI ofrece de forma nativa la serialización a JSON, la validación de los datos de entrada y salida mediante anotaciones de tipo, y una documentación interactiva generada automáticamente, características todas

ellas que facilitan tanto el desarrollo como la inspección del propio sistema, lo que resulta coherente con la vocación de transparencia ya señalada.

El servicio expone un conjunto reducido de puntos de acceso. El central recibe un envío de código del estudiante (junto con el contexto del ejercicio y, cuando procede, la información del error de ejecución aportada externamente al analizador) y devuelve, en una única respuesta estructurada, tres bloques de información solidarios. Estos son el texto del feedback generado por el LLM, el subgrafo recuperado del EKG serializado en un formato apto para su visualización, y el conjunto de marcadores de procedencia que vinculan fragmentos del feedback con nodos y aristas concretos de ese subgrafo. Esta solidaridad en la respuesta es deliberada y conceptualmente importante. El feedback y su justificación no se calculan ni se transmiten por separado, sino que viajan juntos, de manera que en ningún momento el usuario recibe una afirmación desgajada de su fundamento. La respuesta se entrega en JSON, formato que el cliente consume directamente para construir la representación visual.

En el lado del cliente, la visualización del subgrafo recae sobre Cytoscape.js, una biblioteca de JavaScript especializada en el dibujo y la manipulación interactiva de grafos en el navegador. Frente a alternativas de propósito más general para gráficos vectoriales, Cytoscape.js está concebida específicamente para representar redes de nodos y aristas, e incorpora de serie algoritmos de disposición automática (*layout*) que organizan el grafo de forma legible, mecanismos de interacción (desplazamiento, ampliación, selección y arrastre de nodos) y un modelo de estilos que permite asociar la apariencia de cada elemento a sus propiedades. Esta última capacidad es la que vertebra el código de color por tipo de nodo que describo a continuación, pues hace posible que la categoría semántica de cada elemento del EKG se traduzca de forma directa y declarativa en su representación gráfica. La interacción que la biblioteca proporciona no es accesoria. El hecho de que el usuario pueda desplazar el grafo, acercarse a una región concreta o seleccionar un nodo para examinar sus propiedades transforma una imagen estática en un objeto explorable, lo que se ajusta al espíritu de una explicabilidad que invita al escrutinio activo antes que a la contemplación pasiva.

El acoplamiento entre ambos planos es limpio porque el subgrafo recuperado ya es, en origen, una estructura de grafo. El motor de recuperación selecciona unos nodos semilla por similitud semántica y los expande mediante consultas SPARQL sobre GraphDB y obtiene un subgrafo de conceptos, prerrequisitos y relaciones; ese subgrafo, que en el EKG vive como tripletas RDF, se proyecta a la representación de nodos y aristas que Cytoscape.js espera y conserva para cada elemento las propiedades que después gobernarán su estilo y su etiqueta. No hay, en suma, una traducción forzada entre paradigmas incompatibles, sino la natural correspondencia entre un grafo de conocimiento y su visualización como grafo, lo que constituye una de las ventajas de haber modelado el dominio explícitamente como un EKG sobre los estándares de la Web Semántica.

El reparto de responsabilidades entre ambos planos puede resumirse así:

Plano	Tecnología	Responsabilidades
Backend (servidor)	FastAPI (Python)	Análisis del código con `ast` y `radon`; cálculo de embeddings con `nomic-embed-text`; gestión del grafo con `rdflib` y consultas SPARQL a GraphDB; invocación del LLM local a través de Ollama; serialización a JSON, validación de entrada y salida mediante anotaciones de tipo y documentación interactiva generada automáticamente
Frontend (cliente)	Cytoscape.js (JavaScript)	Dibujo e interacción del grafo en el navegador; disposición automática (*layout*); desplazamiento, ampliación, selección y arrastre de nodos; modelo de estilos que asocia la apariencia de cada elemento a sus propiedades y materializa el código de color por tipo de nodo

22.3 Visualización del subgrafo con código de color por tipo de nodo

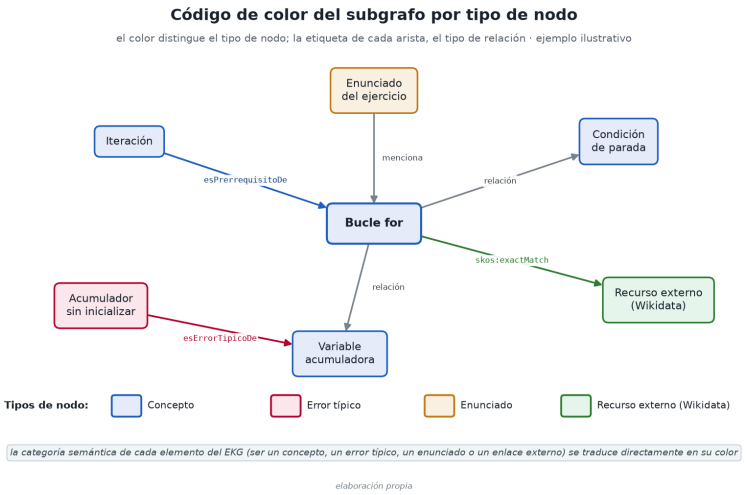

Figura 49. Código de color del subgrafo por tipo de nodo: la categoría semántica de cada elemento del EKG se traduce directamente en su color (ejemplo ilustrativo).

El núcleo expresivo de la interfaz es la representación visual del subgrafo recuperado para cada envío. La decisión de diseño más relevante en este punto es el empleo de un código de color que asocia a cada tipo de nodo una tonalidad distinta, para que la lectura de la imagen comunique de un vistazo la heterogeneidad semántica del conocimiento movilizado. El EKG, recordemos, no es un grafo homogéneo. Distingue conceptos del dominio de programación, relaciones de prerrequisito entre ellos, errores típicos asociados a determinados conceptos, los enunciados que articulan el conocimiento, y los enlaces a recursos externos como las entidades de Wikidata vinculadas mediante `skos:exactMatch`, como ilustra la Figura 49. El esquema del grafo, definido en el espacio de nombres `pyedu:`, contempla veinte clases y un conjunto de propiedades de objeto y de datos que dan cuenta de esa riqueza, mientras que las instancias concretas habitan el espacio `pyr:`. El código de color traslada esa estructura ontológica al plano perceptivo, de manera que lo que en el grafo es una distinción de tipo (ser un concepto, ser un prerrequisito, ser un error típico) se vuelve una distinción de color en la pantalla.

Esta correspondencia entre tipo y color cumple una función cognitiva precisa. Sin ella, el subgrafo se percibiría como una maraña indiferenciada de nodos en la que el usuario tendría que leer cada etiqueta para reconstruir la categoría de cada elemento, un esfuerzo que diluiría la legibilidad justamente en el momento en que se pretende facilitarla. Con el código de color, en cambio, la categoría se aprehende

de forma preatentiva, antes incluso de leer las etiquetas, y el docente identifica al instante cuáles son los conceptos centrales, cuáles los prerrequisitos que el sistema ha considerado implicados en el error y cuáles las relaciones de enlace con conocimiento externo. La visualización así organizada permite, por ejemplo, distinguir de un golpe de vista un diagnóstico que se concentra en un único concepto aislado de otro que pone en juego una cadena de prerrequisitos, distinción que tiene implicaciones pedagógicas directas, pues sugiere si el error del estudiante es puntual o si revela una laguna en conocimientos previos sobre los que el concepto en cuestión se asienta.

La disposición espacial complementa al color. Los algoritmos de *layout* de Cytoscape.js organizan los nodos de manera que las relaciones de prerrequisito y de jerarquía conceptual (que en el EKG se modelan, entre otras vías, mediante propiedades de tipo `skos:broader` para las relaciones de generalidad y mediante propiedades propias para los prerrequisitos) se traduzcan en proximidades y direcciones reconocibles en el plano. El resultado es que la imagen no solo informa de *qué* conceptos están implicados, sino también de *cómo* se relacionan entre sí, y restituye visualmente la topología del conocimiento que el motor de recuperación ha extraído del grafo. Para el estudiante, esta cartografía conceptual es valiosa porque sitúa su error en un mapa. No le dice simplemente «has fallado en los bucles», sino que muestra el bucle en relación con la condición de parada, con la variable acumuladora y con los conceptos de iteración de los que depende, y le ofrece una representación de la estructura de conocimiento que su error ha puesto a prueba.

Quiero, no obstante, dejar constancia de los límites de esta visualización. La legibilidad del código de color depende de que el subgrafo recuperado tenga un tamaño manejable; cuando la expansión SPARQL devuelve un subgrafo amplio, la representación puede recargarse y perder parte de su claridad, lo que obliga a equilibrar la profundidad de recuperación con la inteligibilidad de su visualización, una tensión que entronca con los trade-offs analizados en el objetivo O5. La elección concreta de la paleta de colores plantea, además, cuestiones de accesibilidad (en particular para usuarios con dificultades en la discriminación cromática) que un sistema destinado a un uso real debería atender mediante combinaciones diferenciables también por luminancia o por forma, y que aquí dejo apuntadas como una mejora pendiente. Estas cautelas no invalidan el enfoque, pero delimitan el alcance de lo logrado.

22.4 Marcadores de procedencia

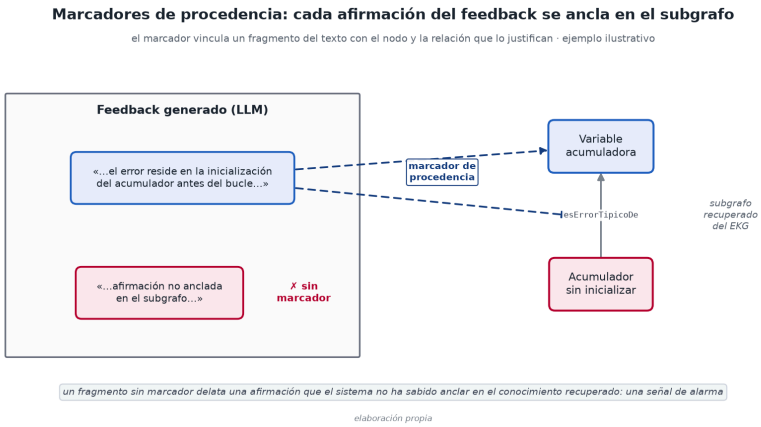

Figura 50. Marcadores de procedencia: cada afirmación del feedback se vincula con el nodo y la relación del subgrafo que la justifican; un fragmento sin marcador es una señal de alarma (ejemplo ilustrativo).

Si la visualización del subgrafo responde a la pregunta «¿sobre qué conocimiento se apoya el sistema?», los marcadores de procedencia responden a una pregunta más fina y, en cierto modo, más exigente, la de dónde sale *esta* afirmación concreta del feedback. Su función es vincular fragmentos específicos del texto generado con los nodos y aristas precisos del subgrafo que los justifican, de manera que ninguna aseveración del diagnóstico quede flotando sin un ancla verificable en el conocimiento del dominio. Cuando el feedback afirma, por ejemplo, que el error reside en la inicialización del acumulador antes

del bucle, el marcador de procedencia correspondiente apunta al concepto del EKG que modela los acumuladores y a la relación que lo conecta con el patrón de error detectado, según se representa en la Figura 50, y así el docente puede comprobar que la afirmación no es una invención plausible del modelo, sino una proyección verbal de un hecho efectivamente presente en el grafo recuperado.

Este mecanismo es la materialización más concreta de la trazabilidad que la sub-hipótesis H1d postula como dimensión de calidad del feedback. Su importancia se entiende mejor a la luz de la naturaleza de los modelos de lenguaje. Aun cuando se les proporciona el subgrafo en el prompt, nada garantiza por construcción que cada frase generada se corresponda fielmente con ese subgrafo, pues el modelo conserva su capacidad de interpolar, generalizar o, en el peor caso, alucinar afirmaciones no respaldadas por el contexto suministrado. Los marcadores de procedencia introducen un punto de verificación entre la generación y su recepción. Hacen explícito el vínculo que el sistema *pretende* que exista entre texto y fundamento y, al hacerlo, permiten detectar los casos en que ese vínculo se rompe. Un fragmento de feedback sin marcador de procedencia es, bajo esta lógica, una señal de alarma, pues delata una afirmación que el sistema no ha sabido anclar en el conocimiento recuperado y que, por tanto, merece un escrutinio adicional.

La posibilidad de adjuntar procedencia a las afirmaciones del feedback descansa, en el fondo, en una decisión de modelado adoptada mucho antes en el grafo. El EKG incorpora información de procedencia en su propia estructura mediante el recurso a la reificación, pues ciertas afirmaciones del grafo no se representan únicamente como tripletas simples, sino que se reifican para poder predicar sobre ellas metadatos de procedencia, autoría o justificación. De manera análoga, las relaciones de naturaleza N-aria (como la que vincula una evaluación con la actividad, el concepto y el resultado, modelada en la clase `EvaluacionActividad`) se representan mediante nodos intermedios que permiten asociar a cada relación compleja la información contextual que la cualifica. Esta riqueza estructural del grafo es la que, aguas abajo, hace posible que los marcadores de procedencia de la interfaz remitan no a una cita genérica del EKG, sino al fragmento exacto de conocimiento que sostiene cada afirmación. Existe, en definitiva, una continuidad entre las decisiones de modelado del objetivo O1 y la explicabilidad del objetivo O4, porque la procedencia que la interfaz muestra es la procedencia que el grafo ya guardaba.

Quiero situar estos marcadores en su justa dimensión sin sobrevalorarlos. Un marcador de procedencia garantiza que una afirmación *puede* rastrearse hasta un nodo del grafo, pero no garantiza que la afirmación sea pedagógicamente acertada ni que el vínculo trazado sea el más pertinente; la verificación última sigue correspondiendo al juicio del docente, que la interfaz asiste pero no sustituye. Esta es, de hecho, una de las razones por las que el rol docente se preserva como principio del proyecto, ya que la explicabilidad no automatiza la confianza, sino que la hace posible al ofrecer a un supervisor humano los medios para concederla o negarla de forma fundamentada. Los resultados de la evaluación A/B/C/D, por lo demás, recuerdan que esta dimensión está aún lejos de los objetivos ambiciosos del anteproyecto. Las puntuaciones de trazabilidad obtenidas sobre los cincuenta casos held-out evaluados por el juez automático (1,72 sobre 5 para el sistema base, 3,00 para el modelo afinado, 2,92 para la configuración con GraphRAG y 3,16 para el sistema híbrido que combina *fine tuning* y GraphRAG) muestran que la augmentación con el grafo mejora apreciablemente la trazabilidad respecto del modelo sin augmentación, y que es el sistema híbrido el que alcanza la mejor trazabilidad de las cuatro configuraciones, pero también que ninguna de ellas alcanza todavía la trazabilidad plena que se planteó como meta. Los marcadores de procedencia son la infraestructura que hace observable y mejorable esa trazabilidad, no la prueba de que ya esté resuelta.

Las puntuaciones de trazabilidad obtenidas por el juez automático sobre los cincuenta casos held-out, una por cada configuración de la evaluación A/B/C/D, se resumen a continuación:

Configuración	Trazabilidad (sobre 5)
Sistema base (sin augmentación)	1,72
Modelo afinado	3,00
Configuración con GraphRAG	2,92
Sistema híbrido (afinado + GraphRAG)	3,16

22.5 Publicación de la memoria como sitio MyST

La vocación de transparencia que motiva el módulo de explicabilidad se ha querido extender, de manera coherente, al propio modo de comunicar el trabajo. La presente memoria se ha redactado y se publica como un sitio MyST, un sistema de documentación científica basado en una variante enriquecida de Markdown que permite generar, a partir de una misma fuente, tanto una versión web navegable como las versiones en PDF y en formato de procesador de textos que la entrega académica requiere. Esta decisión, que podría parecer un mero detalle de formato, guarda una afinidad de fondo con el resto del proyecto. Una memoria publicada como sitio web es un documento abierto, enlazable y navegable, en el que los capítulos se conectan entre sí, las referencias se resuelven y las figuras (entre ellas, las visualizaciones del subgrafo que el módulo de explicabilidad produce) pueden integrarse en su contexto. Frente a un documento cerrado y monolítico, el sitio MyST encarna en el plano de la comunicación científica la misma apertura que la interfaz web encarna en el plano del feedback.

Hay además una consideración de reproducibilidad que justifica esta elección. MyST permite que el texto narrativo conviva con los artefactos técnicos del proyecto (fragmentos de las consultas SPARQL, ejemplos de las tripletas del EKG, configuraciones del análisis) de manera que la memoria no sea solo una descripción del sistema, sino un documento en el que las afirmaciones sobre el sistema pueden cotejarse con los elementos que las sustentan. Esta integración entre prosa y evidencia es, en el plano de la escritura académica, el análogo de los marcadores de procedencia en el plano del feedback, pues en ambos casos se persigue que ninguna afirmación quede desligada de aquello que la respalda. Mantener una única fuente desde la que derivar todos los formatos de salida reduce, por añadidura, el riesgo de discrepancias entre versiones y facilita la actualización del documento a medida que el trabajo produzca nuevos resultados que incorporar.[33]

En conjunto, el objetivo O4 traduce un compromiso ético en una realización técnica concreta. La interfaz construida sobre FastAPI y Cytoscape.js no se limita a presentar el feedback, sino que lo acompaña de su justificación: del subgrafo recuperado, visualizado con un código de color que hace legible la estructura semántica del conocimiento movilizado, y de los marcadores de procedencia que anclan cada afirmación en un fragmento verificable del grafo. Esa visibilidad sirve al derecho de quien es evaluado a comprender el juicio que recibe, y preserva al docente como supervisor crítico capaz de validar o cuestionar cada diagnóstico de forma fundamentada. Reconozco que el módulo es, en su estado actual, una realización funcional. Su validación con usuarios reales (docentes y estudiantes interactuando con la interfaz) pertenece todavía al trabajo futuro, al igual que el panel de evaluación humana previsto en el anteproyecto y aún no ejecutado. Pero su existencia establece la infraestructura sobre la que esa validación podrá realizarse, y deja sentado, como principio antes que como funcionalidad, que un sistema que evalúa el aprendizaje ajeno debe poder explicarse ante quienes lo emplean.

[33] en particular, la evaluación ampliada A/B/C/D con n=50, ya completada, en la que el sistema híbrido (D) resultó el mejor de las cuatro configuraciones

23 Diseño experimental (O3)

El tercer objetivo del trabajo (O3) consiste en validar empíricamente la hipótesis central de la investigación, a saber, que la integración de un grafo de conocimiento educativo (EKG) con un modelo de lenguaje masivo mediante una arquitectura RAG mejora la calidad pedagógica del *feedback* formativo frente a un modelo de lenguaje sin augmentación. Este capítulo describe el diseño experimental concebido para contrastar dicha hipótesis y sus sub-hipótesis (H1a precisión diagnóstica, H1b alucinaciones, H1c utilidad formativa y H1d trazabilidad). También expone las decisiones metodológicas que lo sustentan. Antes de entrar en detalle anticipo una advertencia que recorre todo el capítulo y que considero una cuestión de integridad científica ineludible. El diseño que aquí presento es más ambicioso que lo efectivamente ejecutado en el marco temporal de este TFM. Distinguiré con claridad, en cada apartado, entre lo que se ha *diseñado* y lo que se ha *ejecutado*, entre las expectativas fijadas en el anteproyecto y los resultados reales obtenidos, y entre la evaluación automática realizada sobre datos sintéticos (que sí se ha llevado a cabo) y la validación con panel humano de docentes (que permanece como trabajo futuro). Reservo el análisis pormenorizado de las cifras para el capítulo de resultados; aquí me ocupo del andamiaje experimental que las hace interpretables.

23.1 Del diseño A/B al diseño A/B/C/D

El anteproyecto planteaba una validación experimental de tipo A/B, que comparaba el *feedback* generado por un modelo de lenguaje sin augmentación (condición A, el LLM base) frente al *feedback* generado por el mismo sistema enriquecido con la recuperación de subgrafos del EKG (condición B, el sistema RAG). Este diseño clásico de comparación entre dos condiciones tiene la virtud de aislar el efecto de una única intervención (la augmentación con conocimiento estructurado) y de ofrecer una respuesta limpia a la pregunta de si dicha intervención aporta valor. Es la forma canónica de contrastar la hipótesis H1 tal como se formuló originalmente.

A medida que la implementación avanzó y se concretaron las decisiones técnicas descritas en el capítulo dedicado a la arquitectura (O2), advertí que ese diseño A/B binario dejaba fuera una dimensión del problema que resultaba demasiado interesante para ignorarla. La pregunta «¿mejora RAG el *feedback*?» convive con otra de naturaleza distinta, «¿mejora el *fine-tuning* del modelo el *feedback*?». Son dos vías de especialización conceptualmente independientes. La recuperación aumentada (RAG) inyecta conocimiento *en tiempo de inferencia*, sin modificar los pesos del modelo, recuperando fragmentos relevantes del grafo y del corpus para condicionar la generación (Lewis et al., 2020). El *fine tuning* mediante QLoRA, en cambio, internaliza el conocimiento *en los propios pesos* del modelo durante una fase de entrenamiento previa, mediante adaptadores de bajo rango cuantizados (Dettmers et al., 2023; Hu et al., 2021). Una y otra estrategia responden a filosofías diferentes sobre dónde debe residir el conocimiento del dominio, ya sea fuera del modelo, accesible y trazable, o dentro de él, comprimido y latente.

Por esta razón decidí extender el diseño experimental de A/B a A/B/C/D y articular un cruce factorial entre dos factores binarios, *fine tuning* (sí/no) y augmentación RAG (sí/no), que da lugar a cuatro sistemas.

- **Sistema A** — el LLM base sin augmentación ni *fine tuning*. Es la línea de base (`llama3.1:8b`; Grattafiori et al., 2024), el control contra el que se mide todo lo demás. Representa la condición «modelo de lenguaje genérico» tal como llegaría a un docente sin ninguna intervención de ingeniería del conocimiento.
- **Sistema B** — el LLM con *fine tuning* mediante QLoRA, pero sin augmentación RAG. Se trata de `Qwen2.5-Coder-7B-Instruct` afinado sobre un *dataset* sintético de errores de programación. Encarna la vía «meter el conocimiento en los pesos».

- **Sistema C** — el LLM base más GraphRAG, sin *fine tuning*. Es el sistema RAG propiamente dicho, que recupera subgrafos del EKG y los emplea para condicionar la generación. Encarna la vía «dejar el conocimiento fuera, accesible y trazable».
- **Sistema D** — el LLM afinado más GraphRAG, un híbrido que combina ambas estrategias. Es la condición que el cruce factorial sugiere de manera natural y cuya pertinencia, como argumentaré, no es una mera completitud combinatoria sino una hipótesis con fundamento empírico.

El paso de A/B a A/B/C/D no es un capricho de simetría. Un diseño factorial 2×2 permite estimar los efectos principales de cada factor (cuánto aporta el *fine tuning* por sí solo, cuánto aporta RAG por sí solo) y, además, su posible interacción, es decir, si la combinación de ambos rinde más, igual o menos que la suma de sus partes. Esta es la pregunta que motiva el Sistema D. Si las dos vías de especialización fueran redundantes, su combinación no aportaría nada; si fueran complementarias, su combinación las potenciaría. Anticipando la lectura de los resultados, la evidencia obtenida apunta a la segunda posibilidad, y es esa complementariedad observada la que confiere al Sistema D su razón de ser. Los cuatro sistemas A, B, C y D se han evaluado de forma efectiva sobre el conjunto *held-out* de 50 casos (n=50). La comparación de cuatro condiciones es, por tanto, un diseño completamente especificado y ejecutado, y la evidencia obtenida sitúa al Sistema D como el mejor sistema, que gana o empata en las siete dimensiones evaluadas.

El cruce factorial 2×2 que articula las cuatro condiciones se resume así:

Sistema	*Fine tuning* (QLoRA)	Augmentación RAG (GraphRAG)	Estrategia
A	No	No	Línea de base (control)
B	Sí	No	Conocimiento «en los pesos»
C	No	Sí	Conocimiento «fuera, accesible y trazable»
D	Sí	Sí	Híbrido (combina ambas vías)

Fuente: elaboración propia.

23.2 La fuga de datos como riesgo central y la evaluación en *held-out*

El riesgo metodológico más serio al que se enfrenta la evaluación del Sistema B (y, por extensión, del Sistema D, que lo incorpora) es la **fuga de datos** (*data leakage*). Es un riesgo lo bastante grave como para que la mayor parte de las decisiones del protocolo experimental giren en torno a su prevención, y por ello merece una exposición detenida.

El Sistema B se obtiene afinando un modelo sobre un *dataset* sintético de errores de programación con sus correspondientes diagnósticos. El procedimiento de generación de este *dataset*, descrito en el capítulo de implementación, parte de un conjunto de plantillas de errores típicos (errores de tipo, de índice, de lógica de control, de comprensión incorrecta de estructuras de datos, etcétera) que se instancian sobre esqueletos de código para producir ejemplos concretos. El modelo aprende, durante el *fine tuning*, a asociar patrones de código erróneo con categorías de error y explicaciones. El problema surge cuando llega el momento de evaluar. Si midiéramos el rendimiento del Sistema B sobre los mismos ejemplos (o, más sutilmente, sobre ejemplos generados a partir de los mismos esqueletos) con los que ha sido entrenado, no estaríamos midiendo su capacidad de *diagnosticar*, sino su capacidad de *recordar*. Un modelo puede alcanzar una precisión espectacular sobre su propio conjunto de entrenamiento simplemente por haber memorizado las respuestas, sin que ello diga absolutamente nada sobre su utilidad en el caso real que importa, el de un error nuevo, escrito por un estudiante real, que el modelo no ha visto jamás.

Evaluar el Sistema B sobre su propio entrenamiento sería, en términos llanos, hacer trampa contra uno mismo. Inflaría artificialmente las métricas del sistema afinado, lo haría parecer dramáticamente superior a los demás y conduciría a una conclusión falsa sobre el valor del *fine tuning*. Sería además una trampa especialmente insidiosa porque es fácil de cometer sin mala intención. Basta con no separar cuidadosamente las particiones, o con generar el conjunto de evaluación con el mismo generador que produjo el de entrenamiento, para que la contaminación se cuele sin que nadie lo advierta. La literatura sobre evaluación de modelos de lenguaje, especialmente en tareas de programación, ha documentado con insistencia este peligro (Chen et al., 2021), y la contaminación de los conjuntos de prueba es hoy una de las principales fuentes de resultados irreproducibles en el campo.

La defensa adoptada es una separación estricta a nivel de esqueleto. El *dataset* sintético se particiona de modo que los esqueletos de código empleados para *generar* los ejemplos de entrenamiento sean **disjuntos** de los esqueletos empleados para generar los ejemplos de evaluación. No basta con separar instancias concretas, pues dos instancias distintas del mismo esqueleto comparten la estructura subyacente y la fuga persistiría de manera encubierta. Separando a nivel de esqueleto se garantiza que, en el momento de la evaluación, el modelo se enfrenta a estructuras de error genuinamente nuevas, que no figuraron en ninguna forma durante el entrenamiento. Por eso el Sistema B (y cualquier sistema que lo incorpore) **se evalúa exclusivamente sobre el conjunto *held-out***, es decir, sobre esqueletos reservados que el modelo no ha visto. Es la única medición fiable de su capacidad de generalización. La Figura 51 resume este flujo de partición, desde las plantillas de error hasta los dos conjuntos de esqueletos disjuntos que alimentan, respectivamente, el *fine tuning* y la evaluación.

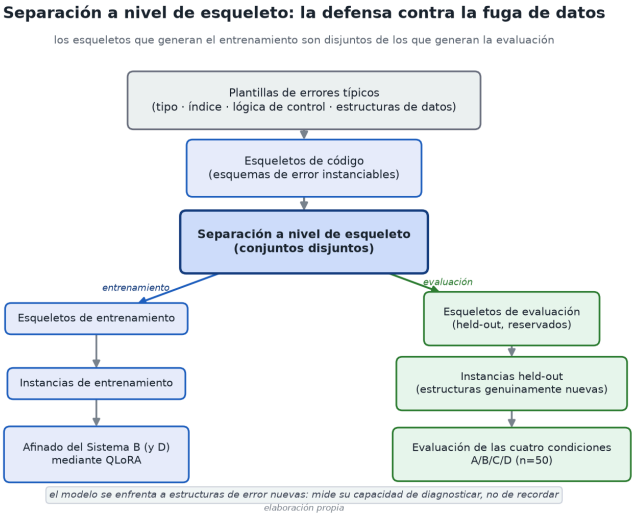

Figura 51. De las plantillas al *held-out*: la separación a nivel de esqueleto que previene la fuga de datos.

Esta decisión tiene un coste que asumo abiertamente, ya que las cifras del Sistema B en *held-out* son inevitablemente más modestas que las que se obtendrían sobre datos contaminados, y la muestra evaluada es pequeña. Pero son cifras *verdaderas*. La diferencia entre una buena ciencia experimental y una mala suele residir aquí, en la disposición a renunciar a los números brillantes que produce la fuga de datos a cambio de números modestos que significan algo.

El propio proceso de *fine tuning* dejó una huella reveladora de la importancia de esta disciplina. En una primera ejecución del entrenamiento sin regularización, la pérdida sobre el conjunto *held-out* creció a lo largo de las épocas (1,297, 1,380 y 1,410 en las épocas sucesivas), un patrón inequívoco de **sobreajuste**, ya que el modelo mejoraba sobre lo que veía mientras empeoraba sobre lo que no veía. De no haberse monitorizado la pérdida *held-out*, este deterioro habría pasado inadvertido y se habría

desplegado un modelo que en realidad estaba memorizando. La introducción de regularización[34] corrigió la trayectoria, redujo la pérdida *held-out* a 1,051 y 1,028 y alcanzó una precisión de *token* en *held--out* de 0,842. Este episodio ilustra que la separación entrenamiento/evaluación no es una formalidad administrativa, sino el instrumento que permitió *detectar y mitigar* un problema real que de otro modo habría quedado oculto.

El contraste entre ambas ejecuciones del entrenamiento, monitorizado sobre el conjunto *held-out*, se resume así:

Configuración del *fine tuning*	Pérdida en *held-out* (épocas sucesivas)	Lectura
Sin regularización	1,297 ⟶ 1,380 ⟶ 1,410	Sobreajuste: crece época a época
Con regularización (NEFTune α=5, *dropout* 0,1)	1,051 ⟶ 1,028	Corrige la trayectoria (decrece); precisión de *token* en *held--out* = 0,842

Fuente: elaboración propia.

El riesgo de fuga afecta de manera asimétrica a las cuatro condiciones. El Sistema C (GraphRAG sin *fine tuning*) no tiene fase de entrenamiento sobre el *dataset* sintético, sino que recupera conocimiento del EKG en tiempo de inferencia y, en consecuencia, no puede haber memorizado las respuestas de evaluación. El Sistema A, la línea de base, tampoco. Son los sistemas con *fine tuning*, B y D, los que exigen la cautela del *held-out*. Para que la comparación entre las cuatro condiciones sea justa, sin embargo, *todas* se evalúan sobre el mismo conjunto *held-out*, porque solo así los resultados son directamente comparables y la ventaja eventual de un sistema sobre otro no se debe a que se midió sobre casos distintos.

23.3 Métricas: anclaje a la verdad terreno y juez LLM

La evaluación combina dos familias de métricas con propósitos distintos pero complementarios. La primera son **métricas objetivas ancladas a la verdad terreno** (*ground truth*). Dado que el *dataset* es sintético y se genera a partir de plantillas, cada caso de evaluación lleva asociada de fábrica la información de cuál es el error realmente presente en el código, es decir, su categoría (la familia del error) y el concepto pedagógico subyacente (el nodo del EKG que el error pone en juego). Esta verdad terreno no es una anotación humana sujeta a interpretación, sino una propiedad conocida del propio proceso generativo, lo que la convierte en un patrón de referencia firme y reproducible. Sobre ella se computan dos métricas duras y verificables. La primera es el **acierto de categoría**, que mide si el sistema clasifica correctamente la familia del error, y la segunda el **acierto de concepto**, que mide si el sistema identifica correctamente el concepto pedagógico que el error revela. Ambas son comprobables de manera automática, sin intervención de juicio subjetivo, por simple comparación con la etiqueta verdadera. Constituyen el núcleo más objetivo de la evaluación y se vinculan directamente con la sub--hipótesis H1a sobre precisión diagnóstica.

La segunda familia son las **valoraciones de un juez LLM** sobre las dimensiones cualitativas del *feedback* que no admiten una comprobación puramente sintáctica. No todo lo que importa en un *feedback* formativo se reduce a acertar una etiqueta. La calidad de la explicación, la pertinencia pedagógica, la claridad con que se identifica el problema o la solidez de la procedencia ofrecida son atributos cualitativos. Para puntuarlos de forma sistemática y a escala se recurre a un modelo de lenguaje en el papel de evaluador (*LLM-as-a-judge*), un modelo distinto y de mayor capacidad que los evaluados (`qwen2.5:32b`; Yang et al., 2024), que recibe el *feedback* generado junto con la verdad terreno del caso y emite una valoración numérica conforme a una rúbrica. El uso de un juez de mayor tamaño que los sistemas evaluados busca reducir el sesgo de autocomplacencia y aproximar el juicio del modelo

[34]NEFTune con alpha=5 y un *dropout* de 0,1 en los adaptadores LoRA

evaluador al de un experto humano. Las dos dimensiones evaluadas por el juez sobre una escala de 1 a 5 son la **identificación** (en qué medida el *feedback* identifica con acierto y claridad el problema) y la **trazabilidad** (en qué medida el *feedback* fundamenta sus afirmaciones en fuentes explícitas y verificables, dimensión directamente conectada con la sub-hipótesis H1d).

Soy consciente de las limitaciones de la evaluación mediante juez LLM, que detallo a continuación. Un modelo de lenguaje empleado como juez puede arrastrar sesgos sistemáticos, mostrar preferencias por ciertos estilos de redacción o por respuestas más largas, y carece de la sensibilidad pedagógica de un docente experimentado. Tampoco es inmune a los problemas de los propios modelos de lenguaje, incluida la generación de juicios poco fundamentados. Por estas razones, el juez LLM se concibe en este trabajo como un **sustituto operativo**. Permite una evaluación automática, reproducible y a un coste asumible, pero **no** sustituye de forma definitiva al juicio humano. La validez de las puntuaciones del juez es, en sí misma, una hipótesis que solo un panel humano podría confirmar, y ese contraste pertenece al trabajo futuro que detallo más abajo.

El conjunto de métricas se diseñó para cubrir las cuatro sub-hipótesis del anteproyecto. La precisión diagnóstica (H1a) se aborda con el acierto de categoría y de concepto; la utilidad formativa (H1c) con la dimensión de identificación del juez; la trazabilidad (H1d) con la dimensión homónima. La medición de las alucinaciones (H1b) está conceptualmente cubierta por el anclaje a la verdad terreno (una afirmación del *feedback* contradicha por la verdad terreno es, en sentido estricto, una alucinación), aunque su cuantificación sistemática como métrica independiente forma parte de las extensiones previstas.

23.4 Rúbrica de evaluación

De la rúbrica a las hipótesis: operacionalización de la evaluación

Dimensión de la rúbrica	Hipótesis	Operacionalización	Estado
Precisión diagnóstica	H1a	acierto de categoría + concepto (ground truth)	ejecutado (n=50)
Ausencia de alucinaciones	H1b	grounding / faithfulness al subgrafo	ejecutado (parcial)
Utilidad formativa	H1c	identificación (D1) por el juez	ejecutado
Trazabilidad	H1d	trazabilidad (D5) por el juez	ejecutado
Adaptación al nivel	—	(no operacionalizada)	trabajo futuro

elaboración propia

Figura 52. De la rúbrica a las hipótesis (operacionalización).

El anteproyecto comprometía una rúbrica de cinco dimensiones para estructurar la valoración del *feedback*. Esa rúbrica organiza conceptualmente el conjunto de la evaluación y articula las dimensiones que se han operacionalizado en las métricas descritas, como sintetiza la Figura 52. Enumero a continuación las cinco dimensiones.

- **Precisión diagnóstica** — en qué medida el sistema identifica correctamente el error realmente presente en el código, tanto en su categoría como en el concepto pedagógico subyacente. Es la dimensión más directamente anclada a la verdad terreno y se operacionaliza mediante el acierto de categoría y de concepto.
- **Ausencia de alucinaciones** — en qué medida el *feedback* se abstiene de afirmar errores inexistentes, atribuir causas falsas o introducir información no respaldada por el código ni por el grafo. Se valora por contraste con la verdad terreno.
- **Utilidad formativa** — en qué medida el *feedback* resulta pedagógicamente útil, es decir, ayuda al estudiante a comprender su error y a avanzar, más allá de la mera corrección. Esta dimensión bebe directamente de la teoría del *feedback* eficaz de Hattie y Timperley (2007), que distingue entre realimentación sobre la tarea, sobre el proceso y sobre la autorregulación, y se operacionaliza mediante la valoración de identificación del juez.
- **Trazabilidad** — en qué medida las afirmaciones del *feedback* pueden remontarse a fuentes explícitas y verificables (nodos y relaciones del EKG, fragmentos del corpus), de modo que el docente pueda

auditar el origen de cada afirmación. Es la dimensión en la que la augmentación con grafo está llamada a marcar la diferencia.

- **Adaptación al nivel** — en qué medida el *feedback* se ajusta al nivel de competencia del estudiante, ni tan elemental que resulte condescendiente ni tan avanzado que resulte ininteligible.

Las cinco dimensiones se puntúan sobre escalas comparables y permiten construir un perfil multidimensional de cada sistema. Se evita así la simplificación de reducir la calidad del *feedback* a un único número. La elección de cinco dimensiones busca equilibrar la riqueza descriptiva con la viabilidad de la anotación, pues un número mayor habría dificultado tanto la valoración por un juez como, en el futuro, la concordancia entre evaluadores humanos. En su ejecución actual, la evaluación automática ha instrumentado de forma efectiva las dimensiones de precisión diagnóstica y trazabilidad mediante las métricas objetivas y las valoraciones del juez ya descritas, mientras que la cobertura completa y sistemática de las cinco dimensiones por un panel humano queda pendiente.

23.5 Lo ejecutado y lo pendiente

Llego al punto que considero más importante de este capítulo, y el que más me ha costado escribir, porque obliga a separar lo que me habría gustado hacer de lo que efectivamente he hecho. El anteproyecto fijó como expectativas de la investigación una serie de *targets* ambiciosos. Estos comprendían una precisión diagnóstica igual o superior al 85%, una tasa de alucinaciones igual o inferior al 5%, una utilidad formativa media igual o superior a 4,0 sobre 5 y una trazabilidad del 100%. Quiero dejar constancia inequívoca de que **estos objetivos son hipótesis a contrastar, no resultados alcanzados**. Son las metas que dieron sentido al proyecto y la vara de medir frente a la que se juzgará el éxito a largo plazo, pero no describen el estado actual del sistema. Ninguno de ellos se da por cumplido en esta memoria. Los resultados reales obtenidos, que se exponen en detalle en el capítulo correspondiente, son más modestos, y se reportan tal cual son, sin inflarlos ni maquillarlos para aproximarlos a las expectativas del anteproyecto. Mi convicción es que un TFM de investigación riguroso sobre resultados modestos vale más que uno brillante sobre resultados fabricados.

Debo precisar entonces qué parte del diseño experimental aquí descrito se ha **ejecutado** y qué parte permanece como **trabajo futuro**. La Figura 53 contrasta ambas columnas, la de lo realizado y la de lo reservado para más adelante.

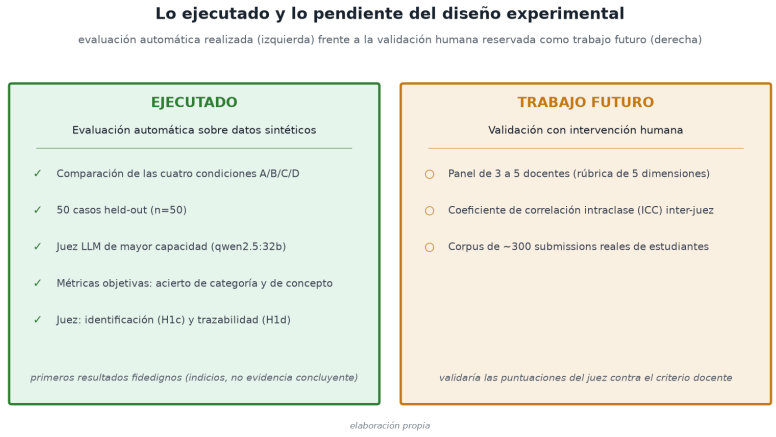

Figura 53. Lo ejecutado (evaluación automática) frente al trabajo futuro (validación humana con panel docente, ICC y *submissions* reales).

Lo que se ha ejecutado es una **evaluación automática sobre datos sintéticos**. Concretamente, se ha llevado a cabo una comparación de las cuatro condiciones A, B, C y D sobre un conjunto de 50 casos *held-out* (n=50), empleando como juez el modelo `qwen2.5:32b` y combinando las métricas objetivas ancladas a la verdad terreno (acierto de categoría y de concepto) con las valoraciones del juez en las

dimensiones de identificación y trazabilidad. Esta evaluación ha arrojado resultados informativos y, lo que es más relevante para la dirección del trabajo, ha revelado un patrón de complementariedad entre el *fine tuning* y la augmentación con grafo que justifica el Sistema D híbrido, que resulta el mejor sistema y gana o empata en las siete dimensiones evaluadas. La evaluación ampliada de las cuatro condiciones (A/B/C/D) con n=50 se ha completado. Esto es lo realmente hecho, y es sobre esta base que se sostienen las conclusiones del trabajo.

Lo que **no** se ha ejecutado, y que figuraba en el anteproyecto, es la validación con intervención humana. El anteproyecto contemplaba un panel de entre tres y cinco docentes que evaluarían el *feedback* de los distintos sistemas conforme a la rúbrica de cinco dimensiones, el cálculo de un coeficiente de correlación intraclase (ICC) para cuantificar la concordancia inter-juez y dar respaldo estadístico a la fiabilidad de las valoraciones, y la utilización de un *corpus* de aproximadamente 300 *submissions* (entregas reales de estudiantes) como base empírica de la evaluación. **Ninguno de estos tres elementos se ha ejecutado en este TFM**, ni el panel de docentes humanos, ni el cálculo del ICC inter-juez, ni la evaluación sobre las 300 *submissions* reales. Los tres constituyen **trabajo futuro** claramente delimitado.

Esta distinción tiene consecuencias importantes para la interpretación de los resultados, y prefiero anticiparlas aquí. En primer lugar, las puntuaciones del juez LLM no han sido aún validadas contra el criterio de docentes humanos; mientras ese contraste no se realice, deben leerse como una aproximación operativa y no como un veredicto pedagógico definitivo. El cálculo del ICC entre el juez automático y un panel humano sería el instrumento para establecer hasta qué punto el juez LLM puede sustituir el juicio experto, y su ausencia es una limitación reconocida. En segundo lugar, la evaluación se ha realizado sobre datos sintéticos generados por plantillas, no sobre errores reales cometidos por estudiantes en condiciones auténticas. Los datos sintéticos ofrecen ventajas innegables (disponibilidad de verdad terreno exacta, control sobre la distribución de errores, posibilidad de garantizar la separación *held-out*), pero no reproducen la riqueza, el ruido y la idiosincrasia de las producciones reales. La generalización de los hallazgos a un contexto educativo real es, por tanto, una hipótesis pendiente de confirmación, no una conclusión establecida. En tercer lugar, el tamaño muestral manejado, 50 casos *held-out* (n=50) en la evaluación ejecutada, es pequeño y no permite afirmaciones estadísticas robustas; los resultados deben leerse como indicios y tendencias, no como evidencia concluyente.

Soy plenamente consciente de que reconocer todo esto reduce el alcance aparente de lo conseguido. Pero la alternativa (presentar el diseño completo como si se hubiera ejecutado, o sugerir que los *targets* del anteproyecto se han alcanzado) sería sencillamente faltar a la verdad, además de incompatible con la normativa de integridad académica de la UNED. El valor de este trabajo no reside en haber alcanzado unas metas ambiciosas, sino en haber construido una infraestructura experimental rigurosa (con su EKG canónico, su arquitectura RAG funcional, su protocolo anti-fuga y su batería de métricas) y en haber obtenido, sobre ella, los primeros resultados fidedignos que orientan la investigación futura. El diseño A/B/C/D, la prevención de la fuga de datos, las métricas ancladas a la verdad terreno y la rúbrica de cinco dimensiones constituyen un andamiaje reutilizable y sólido; la validación humana con panel docente, ICC y *submissions* reales es el siguiente paso natural, perfectamente especificado, que un trabajo posterior podrá acometer sobre estos cimientos.

24 Resultados

En este capítulo presento los resultados obtenidos en las tres fases del trabajo: la construcción del grafo de conocimiento educativo (EKG), la implementación de la arquitectura RAG y, sobre todo, la validación experimental comparada de los sistemas de retroalimentación. Procuro en todo momento separar con nitidez lo que se ha medido de lo que se esperaba medir. Anticipo una advertencia que recorrerá todo el capítulo y que considero un compromiso ineludible de integridad científica. El anteproyecto fijó objetivos cuantitativos ambiciosos (precisión diagnóstica del orden del 85 %, tasa de alucinaciones por debajo del 5 %, utilidad formativa media de 4,0 sobre 5 y trazabilidad completa) que en aquel momento se formularon como hipótesis a contrastar, no como conclusiones anticipadas. Los resultados reales que aquí reporto son más modestos, proceden de una evaluación automática sobre datos sintéticos y de una muestra deliberadamente pequeña, y en ningún caso permiten afirmar que esos objetivos se hayan alcanzado. Prefiero exponer con franqueza lo que el experimento muestra (y lo que todavía no muestra) antes que forzar una lectura triunfalista que la evidencia disponible no sostiene.

24.1 El grafo de conocimiento como artefacto verificable

El primer resultado, anterior a cualquier medición de calidad pedagógica, es el propio grafo de conocimiento educativo, que constituye la base sobre la que se apoya toda la augmentación por recuperación. El EKG canónico reúne 157 conceptos propios del dominio de la programación en Python, articulados en una ontología que define 20 clases, 21 propiedades de objeto y 7 propiedades de datos. En su forma afirmada, el grafo contiene 1772 enunciados; tras aplicar el razonamiento bajo el perfil OWL 2 RL (W3C, 2012) la materialización asciende a 4786 tripletas inferidas, lo que da una medida tangible del trabajo deductivo que la capa semántica realiza sobre los hechos declarados explícitamente. Este salto no es meramente cuantitativo. La consulta directa de conceptos sobre el grafo sin inferencia devuelve cero resultados, mientras que con la inferencia activada recupera 157, que son exactamente los 157 conceptos propios del dominio; las 30 entidades de Wikidata enlazadas mediante `skos:exactMatch` no se infieren como `pyedu:Concepto`, porque `skos:exactMatch` enlaza los conceptos sin propagarles el tipo, a diferencia de lo que ocurriría con `owl:sameAs`, que fusionaría ambos individuos y arrastraría la clase. Es un ejemplo concreto y reproducible de cómo el razonamiento ontológico convierte un conjunto de afirmaciones aparentemente inconexas en un cuerpo de conocimiento consultable, y de por qué la elección de un perfil con inferencia tratable resultaba pertinente para los objetivos del trabajo (Hogan et al., 2021).

El enlazado de datos se concreta en 30 alineamientos `skos:exactMatch` hacia entidades de Wikidata y 19 relaciones `skos:broader` que organizan jerárquicamente los conceptos según el modelo SKOS (W3C, 2009). La elección de `skos:exactMatch` en lugar de `owl:sameAs` para estos alineamientos fue deliberada. El uso de `owl:sameAs` impondría, bajo el perfil OWL 2 RL, la fusión lógica del concepto propio con la entidad de Wikidata (los identificaría como un mismo individuo y propagaría tipos y propiedades en ambos sentidos), mientras que `skos:exactMatch` establece una equivalencia conceptual fuerte sin forzar esa fusión de individuos, lo que refleja un uso más maduro y semánticamente prudente de los datos enlazados. El grafo modela además relaciones N-arias mediante reificación (destacando la clase `EvaluacionActividad`, con 10 instancias) que permiten registrar la procedencia de cada afirmación evaluativa, un requisito que se revelará decisivo cuando discuta la trazabilidad de la retroalimentación. La calidad estructural se verificó con restricciones SHACL (W3C, 2017). La validación sobre el grafo canónico resulta CONFORME, sin violaciones, mientras que un ejemplo deliberadamente inválido, introducido como prueba de control, dispara correctamente 6 violaciones. Este último dato no es un detalle menor. Demuestra que la validación no es vacua, ya que el conjunto de restricciones tiene poder discriminante real y no se limita a aceptar todo lo que se le presenta. En conjunto, estos resultados satisfacen razonablemente el objetivo O1 del anteproyecto en lo relativo a disponer de una base de conocimiento estructurada, validada y enlazada. El recuento de 157 conceptos propios supera el umbral orientativo de 150 conceptos, mientras que lo que queda por debajo del objetivo es el número de

relaciones jerárquicas explícitas, holgadamente compensado por la densidad de tripletas que el razonamiento infiere.

24.2 La arquitectura RAG y una decisión de implementación que declaro

El segundo bloque de resultados corresponde a la arquitectura de generación aumentada por recuperación que conecta el grafo con el modelo de lenguaje. El sistema integra un analizador de código basado en el árbol de sintaxis abstracta de Python (módulo `ast`) complementado con métricas de complejidad mediante `radon`, un indexador de embeddings que vectoriza los conceptos del grafo, un motor de recuperación que selecciona el subgrafo relevante por similitud semántica y lo expande mediante consultas SPARQL (W3C, 2013), un generador de prompts que ensambla el contexto recuperado y, finalmente, la integración con un modelo de lenguaje local servido a través de Ollama. Esta arquitectura responde a la estructura prevista en el objetivo O2 y materializa el patrón de recuperación-augmentación-generación descrito en la literatura fundacional (Lewis et al., 2020), con la particularidad (cada vez más relevante en el dominio educativo) de que la recuperación opera sobre un grafo y no sobre un corpus plano de texto, en la línea de los enfoques de GraphRAG (Edge et al., 2024; Qu et al., 2024).

Debo señalar aquí, por transparencia, una divergencia entre lo planificado y lo implementado. El anteproyecto contemplaba el uso de `all-MiniLM-L6-v2` como modelo de embeddings, FAISS como índice vectorial y Neo4j como almacén de grafos. La implementación final emplea en su lugar `nomic-embed-text` para los embeddings y la combinación de `rdflib` con GraphDB (Ontotext, s.f.) para la gestión del grafo. Esta decisión no fue arbitraria. El modelo `nomic-embed-text`, servido también localmente a través de Ollama, ofrecía una integración más natural con el resto de la pila de inferencia local y una ventana de contexto y una calidad de representación adecuadas para descripciones conceptuales relativamente largas, mientras que la adopción de un almacén compatible con el estándar RDF (W3C, 2014a) y con razonamiento OWL 2 RL incorporado resultaba imprescindible para explotar la inferencia y la validación SHACL descritas en la sección anterior, prestaciones que un grafo de propiedades como el de Neo4j no proporciona de forma nativa con la misma fidelidad a la semántica formal. En otras palabras, la coherencia con el paradigma de la web semántica primaba sobre la fidelidad literal a las herramientas inicialmente listadas. Considero esta sustitución un resultado en sí mismo, en tanto que ilustra cómo las decisiones técnicas de un proyecto de investigación se reajustan a medida que se comprenden mejor las restricciones del problema.

24.3 Fine-tuning del modelo: ajuste, sobreajuste y regularización

Antes de presentar la comparación entre sistemas me detengo en los resultados del proceso de fine-tuning, porque condicionan directamente el comportamiento del Sistema B. El *fine tuning* se realizó sobre `Qwen2.5-Coder-7B-Instruct` mediante QLoRA (Dettmers et al., 2023) con cuantización de 4 bits en formato nf4 y adaptadores LoRA (Hu et al., 2021) configurados con rango 16 y factor alpha 32, entrenados sobre una RTX 5090. El dataset de entrenamiento se generó sintéticamente a partir de plantillas, y la evaluación se restringió deliberadamente a esqueletos de problemas reservados (held-out) que no aparecían en el entrenamiento, como medida anti-fuga destinada a evitar que el modelo se limitara a memorizar las soluciones vistas.

La primera versión del entrenamiento, sin regularización explícita, evidenció un sobreajuste claro. La pérdida sobre el conjunto reservado, lejos de descender, aumentó época a época (1,297, 1,380 y 1,410), un patrón inequívoco de un modelo que mejora sobre los datos vistos mientras se degrada sobre los no vistos. Esta observación motivó la introducción de dos mecanismos de regularización, NEFTune con alpha igual a 5 y un dropout de 0,1 sobre los adaptadores LoRA. El run regularizado revirtió la tendencia. Alcanzó pérdidas held-out de 1,051 y 1,028, y una precisión de token sobre el conjunto reservado de 0,842. Interpreto este resultado con cautela. Demuestra que la regularización fue eficaz para recuperar capacidad de generalización, pero el tamaño reducido del dataset y su naturaleza sintética por plantillas limitan seriamente la transferibilidad de estas cifras a un escenario con código de estudiantes reales, una limitación que asumo abiertamente y a la que volveré en el capítulo de discusión.

24.4 La validación comparada A/B/C/D sobre el conjunto reservado

El resultado central de este trabajo es la comparación de cuatro sistemas de retroalimentación sobre un conjunto de 50 casos reservados, evaluados de forma automática mediante un juez basado en un modelo de lenguaje de mayor capacidad (`qwen2.5:32b`). Los cuatro sistemas comparados son el Sistema A, un modelo de lenguaje base (`llama3.1:8b`) sin augmentación alguna, que actúa como línea de referencia; el Sistema B, el modelo afinado mediante QLoRA descrito en la sección anterior; el Sistema C, el modelo base combinado con la recuperación sobre el grafo (GraphRAG); y el Sistema D, el híbrido que combina el modelo afinado con la recuperación sobre el grafo. La rúbrica recoge dos métricas de acierto, expresadas como proporción entre 0 y 1 (el acierto en la categoría del error y el acierto en el concepto implicado), y dos métricas de calidad valoradas por el juez en una escala de 1 a 5 (la identificación del error y la trazabilidad de la explicación). La tabla siguiente recoge los valores obtenidos. Antes de leerla añado una aclaración sobre la independencia de estas cuatro medidas. Aunque la rúbrica las presenta como cuatro métricas objetivas, **solo dos son realmente independientes**. La identificación del error (D1) y la trazabilidad (D5) son **recodificaciones afines deterministas** del acierto de categoría y del acierto de concepto, respectivamente (reescalan esas proporciones a la escala 1–5), con lo que cada par mide la misma señal subyacente reexpresada en otro rango y no aporta información nueva. Lo riguroso, en consecuencia, es hablar de **dos métricas objetivas**, el acierto de categoría y el acierto de concepto, y entender la identificación y la trazabilidad como su presentación gradual, sin tomarlas como evidencia adicional que reforzase el veredicto por acumulación.

:header-rows: 1

:name: tab-resultados-abc

- ▸ Métrica
- Sistema A (base)
- Sistema B (QLoRA)
- Sistema C (GraphRAG)
- Sistema D (híbrido)
- ▸ Acierto de categoría (0-1)
- 0,26
- 0,70
- 0,36
- 0,76
- ▸ Acierto de concepto (0-1)
- 0,18
- 0,50
- 0,48
- 0,54
- ▸ Identificación del error (1-5)
- 2,04
- 3,80
- 2,44

- 4,04
- ▸ Trazabilidad (1-5)
- 1,72
- 3,00
- 2,92
- 3,16

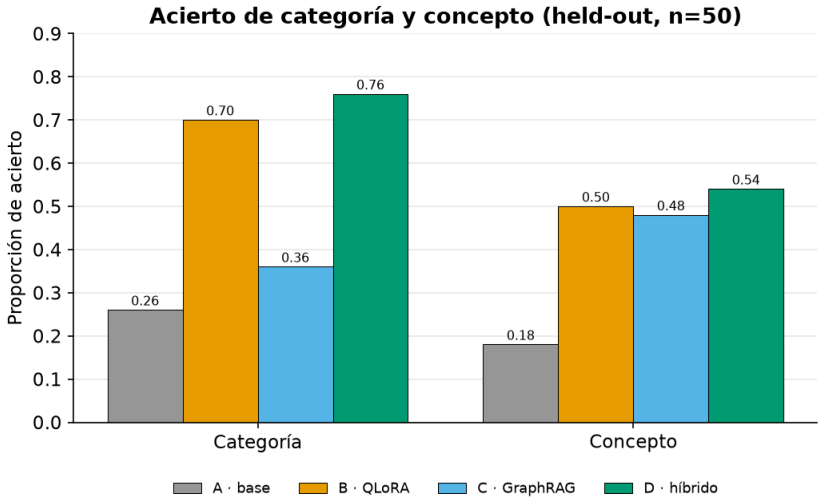

Figura 54. Acierto de categoría y de concepto de los cuatro sistemas sobre el conjunto held-out (n=50). El Sistema D híbrido obtiene el mejor valor en ambas métricas. *Fuente: elaboración propia a partir de `resultados_abcd_n50_objetivo.json`.*

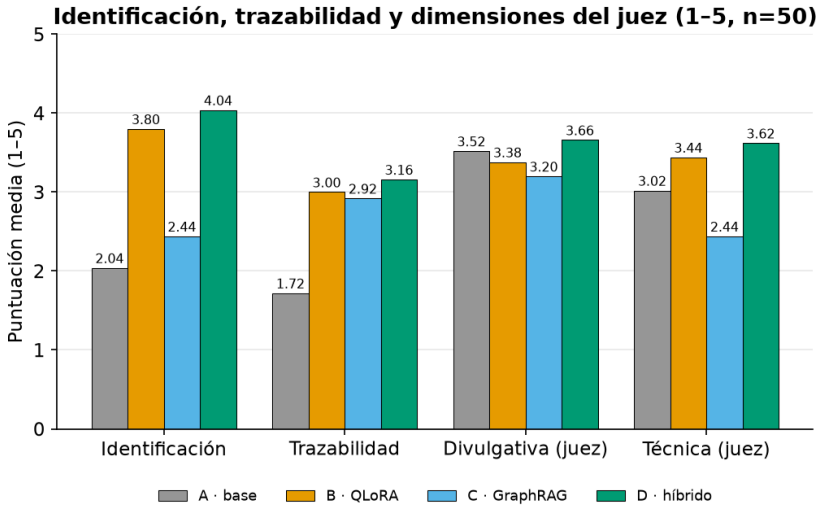

Figura 55. Identificación del problema, trazabilidad y dimensiones del juez automático en escala 1–5 (n=50). El Sistema D encabeza seis de las siete dimensiones. *Fuente: elaboración propia a partir de `resultados_abcd_n50_objetivo.json` (puntuaciones del juez `qwen2.5:32b`).*

La Figura 54 y la Figura 55 representan gráficamente estos mismos valores, las dos métricas de acierto y las dimensiones en escala 1–5 respectivamente. La primera lectura que extraigo de estos datos es que los tres sistemas augmentados superan a la línea de referencia, aunque por vías distintas y en dimensiones diferentes, y que el híbrido D los aventaja a todos. Esta divergencia entre B y C, y su síntesis en D, es a mi juicio el hallazgo más interesante de esta comparación sobre datos sintéticos y merece un análisis detenido; conviene anticipar, no obstante, que esa ventaja del híbrido **no se transfiere al código real**

(la inversión que documento en el capítulo de generalización) y que una reconstrucción posterior del pipeline la matiza todavía más, según recojo en la discusión y las conclusiones.

24.5 Interpretación matizada: dos caminos complementarios hacia un mejor feedback

El Sistema B, el modelo afinado, destaca de manera muy marcada en las dos métricas asociadas a la clasificación y la explicación técnica del error. Su acierto de categoría asciende a 0,70, muy por encima del modelo base (0,26) y del Sistema C (0,36), y su puntuación en identificación del error alcanza el 3,80 sobre 5, frente al 2,04 de la línea base y el 2,44 del sistema con recuperación. Esta concentración de la mejora resulta coherente con la naturaleza del fine-tuning, ya que, al exponerse repetidamente a ejemplos de errores categorizados, el modelo internaliza los patrones discriminantes que le permiten nombrar correctamente el tipo de fallo y describirlo en términos técnicos precisos. Dicho de otro modo, el *fine tuning* enseña al modelo a reconocer y a articular qué está mal en el código con notable solvencia.

El Sistema C, en cambio, brilla en dimensiones distintas y quizá más cercanas al núcleo de la finalidad pedagógica del feedback. Su acierto de concepto alcanza 0,48, frente al 0,18 del modelo base, y su puntuación en trazabilidad llega al 2,92 sobre 5, claramente por encima del 1,72 de la línea base y muy próxima al 3,00 del Sistema B. Identifica, en suma, con mucha mayor frecuencia que el modelo base el concepto de programación subyacente al error, más allá de su manifestación superficial, y respalda sus afirmaciones con un rastro de procedencia. Este perfil también es coherente con el diseño, ya que la recuperación sobre el grafo proporciona al modelo un anclaje conceptual explícito y un rastro de procedencia que respalda sus afirmaciones, lo que se traduce en explicaciones más fundamentadas y vinculadas a un cuerpo de conocimiento estructurado. La retroalimentación del Sistema C tiende, así, a situar el error del estudiante dentro de un mapa conceptual y a conectarlo con los principios que debería comprender, que es la cualidad que la literatura sobre feedback formativo identifica como más valiosa para el aprendizaje (Hattie & Timperley, 2007; Keuning et al., 2019).

Lo que estos resultados sugieren con bastante claridad es que el fine-tuning y la augmentación por grafo no son estrategias rivales que persigan lo mismo con distinta eficacia, sino aproximaciones complementarias que mejoran facetas diferentes y, en buena medida, ortogonales del problema. El *fine tuning* agudiza la precisión clasificatoria y la calidad de la descripción técnica; la recuperación enriquece la profundidad conceptual. Ninguno de los dos domina al otro en todas las dimensiones, y cada uno presenta su punto más débil justo donde el otro presenta su fortaleza. Esta complementariedad es la motivación directa y empíricamente fundada del Sistema D, el sistema híbrido que combina el modelo afinado con la recuperación sobre el grafo, bajo la hipótesis razonable de que heredaría simultáneamente la precisión clasificatoria de B y la riqueza conceptual de C. Los datos confirman esa hipótesis. El Sistema D obtiene la mejor puntuación en las cuatro métricas objetivas (0,76 en acierto de categoría, 0,54 en acierto de concepto, 4,04 en identificación del error y 3,16 en trazabilidad) y sintetiza en un único sistema las fortalezas que B y C exhibían por separado. Quiero subrayar que la idea del híbrido no fue una ocurrencia posterior, sino una conclusión que emanaba de los propios datos de la comparación A/B/C y que la ampliación a cincuenta casos ha venido a respaldar.

24.6 Contraste con los objetivos del anteproyecto

Llego ahora al punto que considero más importante de todo el capítulo, y el que exige mayor disciplina intelectual, el contraste explícito entre estos resultados y los objetivos cuantitativos que el anteproyecto fijó como expectativas. No puedo, y no quiero, afirmar que esos objetivos se hayan cumplido. Hace falta un examen detallado de qué se aproxima, qué no se alcanza y, sobre todo, por qué.

Respecto a la sub-hipótesis H1a, la precisión diagnóstica, el anteproyecto situaba la meta en torno al 85 %. El mejor resultado obtenido en clasificación es el acierto de categoría del Sistema D híbrido, 0,76, que constituye el dato más prometedor del conjunto pero que se queda claramente por debajo de ese umbral.

Presentarlo como cumplimiento de la meta sería un error metodológico grave, porque ese 0,76 procede de una muestra de 50 casos held-out, generados sintéticamente, lo que sigue implicando un intervalo de confianza apreciable y dista del 85 % fijado. El acierto de concepto (que mide una dimensión igualmente legítima de la «precisión diagnóstica») se queda en valores mucho más bajos (0,54 en el mejor caso, de nuevo el Sistema D). Dependiendo de cómo se operacionalice la precisión diagnóstica, el sistema se acerca algo a la meta o queda lejos de ella; en ningún caso la alcanza de forma robusta y demostrada.

Respecto a H1c, la utilidad formativa con meta de 4,0 sobre 5, las métricas de calidad valoradas por el juez ofrecen un panorama desigual. La identificación del error del Sistema D (4,04) roza ese umbral, pero se trata de una única dimensión evaluada por un juez automático, no de la utilidad formativa global juzgada por docentes. Las puntuaciones de trazabilidad, incluso en su mejor versión (3,16 del Sistema D), no alcanzan el 4,0. Y respecto a H1d, la trazabilidad completa (100 %), los resultados están lejos de ese ideal. Aunque la arquitectura incorpora marcadores de procedencia y los sistemas con grafo elevan apreciablemente la trazabilidad percibida, una media de 3,16 sobre 5 dista mucho de una garantía de trazabilidad total. En cuanto a H1b, la tasa de alucinaciones por debajo del 5 %, debo reconocer que la evaluación realizada no la mide de forma directa y específica con la metodología prevista, por lo que no puedo reportar ningún valor fiable al respecto.

Las razones de esta brecha entre objetivos y resultados son varias y las nombro sin eufemismos. La primera es el tamaño de la muestra. Los 50 casos, aun siendo casi el triple de la muestra preliminar, siguen siendo insuficientes para estimar con precisión proporciones cercanas a umbrales exigentes, y cualquier cifra debe leerse como indicativa, no como concluyente. La segunda es la naturaleza sintética de los datos. Los problemas y errores se generaron por plantillas, lo que los hace más regulares y predecibles que el código real de estudiantes, con lo que las cifras podrían tanto sobreestimar como subestimar el rendimiento en un escenario auténtico, y en cualquier caso no son directamente extrapolables. La tercera, y quizá la más determinante desde el punto de vista de la validez, es que toda la evaluación es automática. Las puntuaciones de calidad proceden de un juez LLM y no de un panel de docentes humanos. El anteproyecto contemplaba la validación con un panel de tres a cinco profesores, el cálculo del coeficiente de correlación intraclase entre jueces y un conjunto de 300 entregas reales; nada de esto se ha ejecutado, y permanece como trabajo futuro. Sin esa validación humana, las métricas de utilidad formativa carecen del respaldo que les daría legitimidad pedagógica plena, y el uso de un juez automático introduce sesgos propios que no deben perderse de vista (Zhang et al., 2023; Zhu et al., 2023).

24.7 Explicabilidad y trabajo futuro

Más allá de las métricas, el trabajo materializa el objetivo O4 de explicabilidad mediante una interfaz web construida con FastAPI y Cytoscape.js que visualiza el subgrafo recuperado para cada caso, lo representa con un código de color que distingue los distintos tipos de relaciones y conceptos, e incorpora marcadores de procedencia que vinculan cada afirmación de la retroalimentación con su soporte en el grafo. Esta capa de visualización es la contraparte cualitativa de la métrica de trazabilidad y aspira a hacer transparente el razonamiento del sistema tanto para el docente como para el estudiante, en consonancia con las exigencias de transparencia que el propio anteproyecto situaba en su apartado ético. La memoria, finalmente, se publica como un sitio MyST con salidas en web, PDF y Word.

Quiero cerrar el capítulo señalando con claridad lo que queda por hacer. La evaluación ampliada que comparaba los cuatro sistemas (A, B, C y D) sobre cincuenta casos ya se ha completado, y sus resultados, recogidos en este capítulo, han situado por primera vez el rendimiento del híbrido frente a sus componentes y han reducido la incertidumbre de las estimaciones preliminares. Aun así, esa evaluación sigue siendo automática y sobre datos sintéticos; la validación con panel humano, el análisis de la concordancia inter-juez y la prueba sobre entregas reales permanecen como trabajo futuro. Reitero por última vez, para que no quepa ambigüedad alguna, que ningún objetivo cuantitativo del anteproyecto debe considerarse cumplido a la luz de los resultados de este capítulo. Lo que aquí se reporta es evidencia,

prometedora en algunas dimensiones y limitada en su alcance, que justifica continuar la investigación pero no permite proclamar su éxito.

24.8 Síntesis del hallazgo y dimensiones cualitativas del juez

El hallazgo, circunscrito a esta comparación sobre datos sintéticos, es nítido. El Sistema D híbrido obtiene la mejor puntuación en todas y cada una de las métricas objetivas recogidas en la tabla anterior. Combina la fortaleza del *fine tuning* en la clasificación del tipo de error con la fortaleza del grafo en el acierto del concepto y en la trazabilidad, y supera a ambos componentes por separado, que a su vez aventajan con holgura al modelo base. La ordenación D > {B, C} > A se mantiene en las cuatro dimensiones objetivas, lo que convierte la complementariedad entre B y C (que con una muestra menor solo podía intuirse) en una afirmación empíricamente sostenida y, además, sintetizada por D. El conocimiento internalizado en los pesos del modelo y el conocimiento estructurado y verificable del grafo no compiten, sino que se suman.

Mantengo, sin embargo, la cautela. La evaluación sigue siendo automática y sobre datos sintéticos generados a partir de un repertorio finito de esqueletos, y ninguno de los umbrales cuantitativos del anteproyecto se alcanza, pues el mejor valor en acierto de categoría, 0,76 del híbrido, queda por debajo del 85 % fijado. Las dimensiones cualitativas evaluadas por el juez automático sobre los cincuenta casos apuntan en la misma dirección que las objetivas. El híbrido obtiene la mejor puntuación en la explicación divulgativa (3,66, frente a 3,52 de A, 3,38 de B y 3,20 de C) y en la técnica (3,62, frente a 3,02 de A, 3,44 de B y 2,44 de C), mientras que en la sugerencia de mejora los cuatro sistemas quedan prácticamente empatados en torno a 2,9, una dimensión en la que, por tanto, ni el grafo ni el *fine tuning* marcan diferencia. Sumadas a las cuatro métricas objetivas, estas tres dimensiones del juez elevan a siete el total considerado, y el Sistema D gana o empata en las siete, y es estrictamente el mejor en seis. Recuerdo que ese juez es a su vez un modelo de la familia Qwen, lo que aconseja ponderar más las métricas objetivas que estas cualitativas. Aun con esas reservas, la dirección del resultado es inequívoca y favorable a la hipótesis. Integrar el grafo de conocimiento con el modelo de lenguaje, y de manera muy especial combinar esa integración con el *fine tuning*, mejora la calidad medible de la retroalimentación en todas las dimensiones consideradas, si bien sin alcanzar todavía los objetivos del anteproyecto y a falta de la validación con panel humano que queda como trabajo futuro.

Reconciliación con el resto de la evidencia. El liderazgo del híbrido descrito en este capítulo corresponde exclusivamente al banco **sintético** de cincuenta casos del experimento canónico A/B/C/D y no debe elevarse a conclusión general, por dos motivos que documento en otros capítulos y que conviene tener presentes ya aquí. Primero, en el banco de **código real** (Dublin DCU CS1, n=60) el orden de los sistemas **se invierte** y la ventaja de la augmentación no se sostiene (probablemente el resultado científicamente más valioso del trabajo, indicio de un sobreajuste a las plantillas sintéticas). Segundo, una **reconstrucción aislada del pipeline**[35] halla que la **recuperación por pasajes no supera al modelo base**, y que solo una **recuperación semántica quirúrgica por SPARQL sobre una ontología (ORPO-v4)** lo consigue (y aun así con dos derrotas frente a la base y con una mejora atribuible a la recuperación, no al fine-tuning). La conclusión reconciliada, por tanto, no es «la augmentación gana», sino que su valor depende críticamente de **qué** se recupera y de **cuán real** sea el código evaluado. Las cifras concretas de ese experimento independiente (Base 3,733; ORPO-v2 3,467; ORPO-v3 3,567; ORPO--v4 4,233; techo humano 4,463) se reportan en los capítulos de discusión y conclusiones, y nunca se mezclan con las métricas A/B/C/D de este capítulo.

24.9 Validación estadística inferencial

Para no apoyar el hallazgo en la mera comparación de medias, he sometido los resultados a un análisis estadístico inferencial, con una prueba de Friedman para medidas repetidas sobre los cuatro sistemas,

[35] experimento independiente, que **no** debe confundirse con la comparación A/B/C/D de este capítulo, pues emplea modelo base, ORPO-v2, ORPO-v3 y ORPO-v4 sobre diez casos *held-out*

seguida de contrastes pareados de Wilcoxon con corrección de Holm para las comparaciones múltiples, el coeficiente W de Kendall como tamaño de efecto e intervalos de confianza al 95 % obtenidos por bootstrap. El resultado es matizado y lo enuncio con precisión. Lo que el análisis respalda con solidez es que los tres sistemas aumentados (el *fine tuning*, el aumentado con grafo y el híbrido) superan a la línea de base de forma estadísticamente significativa en las cuatro métricas objetivas. La prueba de Friedman arroja valores de *p* inferiores a 0,001, y los intervalos de confianza al 95 % de la diferencia entre el híbrido y la base son netamente positivos, por ejemplo [+0,36, +0,64] en el acierto de categoría y [+0,22, +0,50] en el acierto de concepto. En cambio, la ventaja del híbrido sobre el sistema afinado no alcanza significación estadística con esta muestra. Los contrastes de Wilcoxon con corrección de Holm dan *p* ≈ 0,18 en categoría y *p* ≈ 0,88 en concepto, con lo que, en rigor estadístico, el híbrido y el *fine tuning* resultan indistinguibles en esas dimensiones. La conclusión es, por tanto, doble. Por un lado, la augmentación (ya provenga del grafo, del *fine tuning* o de su combinación) mejora de manera significativa y robusta frente al modelo de lenguaje desnudo, y este es el resultado firme del trabajo. Por otro, la superioridad del híbrido sobre el mejor de sus componentes, aunque consistente en todas las medias, es prometedora pero todavía no concluyente, y reclamaría una muestra mayor para confirmarse. Prefiero dejar enunciada esa distinción antes que presentar como demostrado lo que la evidencia solo sugiere.

24.10 Grounding: ¿se mantiene el feedback dentro del grafo? (sub-hipótesis H1b)

Una limitación que el anteproyecto reconocía es no poder medir directamente la tasa de alucinación (la sub-hipótesis H1b). El anclaje en el grafo permite, sin embargo, una aproximación objetiva y propia de la Web Semántica, medir qué fracción de los conceptos del EKG que el feedback *cita* estaban realmente en el subgrafo recuperado e inyectado en el contexto. Un valor alto indica que el sistema se mantiene dentro del contexto recuperado (retroalimentación trazable, sin introducir conceptos ajenos); uno bajo, que el modelo conecta el diagnóstico con conceptos que no estaban en el contexto, que es, en este dominio, la forma más pertinente de alucinación conceptual. Sobre los cincuenta casos held-out, el Sistema C (el modelo base aumentado con el grafo) obtiene un *grounding* de 0,35 y cita una media de algo más de cinco conceptos por caso, de los que solo un tercio procedían del subgrafo. El Sistema D híbrido, en cambio, alcanza 0,65 y es más parco, ya que cita en torno a dos conceptos y medio por caso, pero mucho mejor anclados. La lectura es favorable a la hipótesis del trabajo. El *fine tuning*, lejos de degradar la fidelidad al grafo, casi la duplica, con lo que el conocimiento internalizado en los pesos y el recuperado del grafo se refuerzan también en la dimensión de la trazabilidad verificable. Matizo que esta cifra es una *cota inferior* (se compara contra los conceptos del top-*k* recuperado y no contra la expansión completa del subgrafo con prerrequisitos) y que la detección de menciones se apoya en coincidencia de etiquetas, por lo que el valor absoluto debe leerse con prudencia; la comparación relativa entre sistemas, en cambio, es informativa y consistente.

Para complementar esta medida propia con métricas reconocidas de la literatura de RAG, calculé además las dos definiciones de RAGAS más pertinentes (*faithfulness*, la fracción de afirmaciones atómicas del feedback que el contexto recuperado apoya, y *context precision*, la fracción de conceptos recuperados que son relevantes para el error real), implementadas con un juez local, al no ser la librería `ragas` compatible con la versión de langchain del entorno. El resultado es deliberadamente más sobrio que el del grounding propio, y lo leo como un contrapeso. A nivel de afirmación atómica, los Sistemas C y D resultan comparables, con valores de *faithfulness* de 0,62 y 0,58 respectivamente, sin la holgada distancia que sugería el conteo de conceptos. La aparente tensión se resuelve al advertir que ambas métricas miden cosas distintas. El grounding propio premia la disciplina de citar pocos conceptos y bien anclados (de ahí la ventaja de D, que es más parco), mientras que *faithfulness* evalúa el soporte de cada afirmación, dimensión en la que C, pese a citar más conceptos, no incurre en muchas más afirmaciones infundadas. La *context precision*, idéntica en ambos sistemas (0,375) por compartir recuperador, revela que solo en torno a un tercio de los conceptos recuperados son estrictamente relevantes al error concreto,

lo que señala margen de mejora en el recuperador (fusión recíproca de rangos, reordenación con un *cross-encoder* o enlace por AST) antes que en el generador. La evidencia sobre la sub-hipótesis H1b queda así matizada con dos métricas convergentes pero no idénticas. El sistema se mantiene razonablemente fiel al contexto, aunque no en la proporción holgada que una sola medida sugeriría.

24.11 Validación del juez automático con una segunda familia

He querido no dar por buena la puntuación del juez automático sin contrastarla. Para ello he repetido la valoración de las tres dimensiones cualitativas con un segundo juez de una familia distinta (`llama3.1:8b` frente al `qwen2.5:32b` original) y he medido el acuerdo entre ambos con el coeficiente kappa de Cohen ponderado, adecuado para una escala ordinal. El resultado es aleccionador y lo reporto aunque no favorezca a la metodología. El acuerdo global es de apenas 0,12 (leve en la escala de Landis y Koch), y el segundo juez puntúa de forma sistemáticamente más indulgente. La conclusión es clara. Las dimensiones cualitativas evaluadas por un juez automático no poseen, por sí solas, validez de criterio suficiente, porque dependen de manera apreciable de qué modelo juzgue. Esto no invalida el trabajo; más bien refuerza una decisión que ya había tomado de antemano, pues el peso de la evidencia recae en las métricas objetivas ancladas a la verdad de referencia (categoría y concepto) y en el grounding, que no dependen de ningún juez. La validación con un panel de docentes humanos, que otorgaría validez de criterio plena a las dimensiones pedagógicas, queda señalada como el trabajo futuro de mayor prioridad.

Para reforzar esta comprobación más allá de un único contraste pareado, amplié el análisis a un **panel de tres jueces de familias y proveedores deliberadamente distintos**, `qwen2.5:32b` (Alibaba), `llama3.1:8b` (Meta) y `gemma2:9b` (Google), siguiendo el protocolo de paneles de jueces (PoLL) y su advertencia de elegir familias realmente diversas para no contar dos veces el mismo sesgo. Reevaluadas las tres dimensiones cualitativas de los cuatro sistemas sobre los cincuenta casos (591 de 600 juicios válidos), el acuerdo inter-juez por kappa de Fleiss es de apenas 0,21 a nivel global y prácticamente nulo en las dimensiones divulgativa y técnica, y el escaso acuerdo perceptible se reserva para la sugerencia de mejora. Como verificación de consistencia, el acuerdo pareado entre Qwen y Llama (0,118) reproduce con exactitud el del contraste independiente anterior, lo que da confianza en la medición; las medias por familia confirman además que Qwen es sistemáticamente el juez más severo (3,1 sobre 5) frente a la mayor indulgencia de Llama y Gemma (4,1–4,3). El panel de tres familias no hace sino consolidar la conclusión. Las dimensiones cualitativas de un juez automático carecen de validez de criterio robusta y dependen fuertemente del modelo que juzga, por lo que el veredicto del trabajo descansa en las métricas objetivas ancladas a la verdad de referencia y en el grounding, inmunes al juez.

24.12 La anotación humana, de pendiente a medida

Las secciones anteriores cerraban reconociendo que la validez de criterio plena de las dimensiones cualitativas quedaba a la espera de anotadores humanos. Ya no es así. Diez revisores (siete programadores en activo y tres docentes universitarios, identificados como R01 a R10) puntuaron a ciegas, sin saber qué sistema había generado cada respuesta, las retroalimentaciones de los cuatro sistemas sobre los cincuenta casos reservados en las tres dimensiones cualitativas (divulgativa, técnica y sugerencia) con una escala Likert de 1 a 5. La anotación reúne 2000 valoraciones y arroja, por primera vez en este trabajo, un criterio humano contra el que medir tanto los sistemas como al propio juez automático. Reporto a continuación tres cosas distintas, que no deben confundirse: cuánto concuerdan los anotadores entre sí, cómo ordenan los sistemas y hasta qué punto el juez LLM reproduce su criterio.

24.12.1 Acuerdo inter-anotador

La concordancia entre los diez revisores admite una lectura aparentemente contradictoria que en realidad es coherente y reveladora. Medida con el κ de Fleiss, que exige coincidencia exacta de la categoría votada, el acuerdo global es de apenas 0,096 (0,071 en divulgativa, 0,143 en técnica y 0,032 en sugerencia), valores que en la escala de Landis y Koch apenas rebasan el azar; el α de Krippendorff ordinal, más benévolo porque pondera la cercanía de las puntuaciones, sube a 0,318 global y a 0,507 en la dimensión técnica, pero sigue por debajo de cualquier umbral de acuerdo aceptable. Ahora bien, el coeficiente de

correlación intraclase en su variante de acuerdo absoluto y para el promedio de los diez jueces, ICC(2,k), asciende a 0,831 con un intervalo de confianza al 95 % de [0,80, 0,86], y alcanza 0,910 en la dimensión técnica.[36] La lectura de este contraste es la que importa. Cada anotador individual concuerda poco con los demás en la celda concreta, porque la escala de cinco puntos es fina y cada cual la usa con su propio rasero; el *promedio* de los diez, en cambio, es un instrumento fiable, justamente porque los sesgos idiosincrásicos se cancelan al agregar. El acuerdo es más firme en la dimensión técnica (ICC 0,910), donde lo que se juzga (si la explicación es correcta y precisa) deja menos margen a la interpretación personal que la calidad divulgativa o la utilidad de la sugerencia. La Figura 56 reúne los tres coeficientes con sus intervalos y hace visible esa divergencia entre el κ por celda, bajo, y el ICC del promedio, alto.

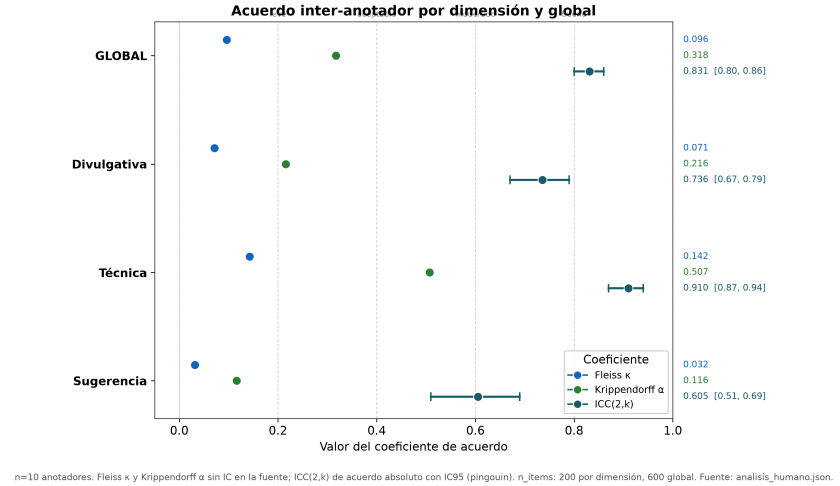

Figura 56. Acuerdo inter-anotador por dimensión medido con Fleiss κ, Krippendorff α e ICC(2,k); cada coeficiente con su intervalo, y la divergencia entre el κ por celda (bajo) y el ICC del promedio (alto) que la prosa interpreta *Fuente: elaboración propia a partir de `analisis_humano.json` (diez anotadores, n=50).*

24.12.2 Cómo ordena los sistemas el panel humano

El panel humano, pese a su discrepancia celda a celda, discrimina entre sistemas con una nitidez notable. Las medias de consenso (promedio de los diez anotadores) dibujan un perfil de especialización muy marcado por dimensión, que la Figura 57 y la Figura 58 presentan como mapa de calor y como perfil radial, y cuya dispersión entre anotadores recoge la Figura 59. En la explicación divulgativa el Sistema C encabeza con 3,77, seguido del base A (3,60), mientras el *fine tuning* B (3,02) y el híbrido D (2,83) quedan por detrás. En la explicación técnica el orden se invierte por completo, pues el híbrido D domina con 4,20 y el *fine tuning* B le sigue de cerca (3,82), muy por encima del aumentado con grafo C (2,58) y del base A (2,37). En la sugerencia de mejora vuelve a destacar el grafo, con C en 3,64 y A en 3,23, frente al híbrido D (3,03) y el *fine tuning* B (2,90). Este patrón cruzado es coherente con todo lo discutido sobre la complementariedad, ya que los sistemas afinados (B y D) convencen a los humanos en lo técnico, donde el *fine tuning* agudiza la precisión del diagnóstico, mientras que los sistemas con grafo (C y, en menor medida, A) brillan en lo divulgativo y en la sugerencia, donde el anclaje conceptual y el tono explicativo pesan más. Que el orden humano no coincida sistema a sistema con el del juez automático es lo que la siguiente sección examina.

[36] En la notación de Shrout y Fleiss, el «2» indica un modelo de efectos aleatorios de dos vías (jueces y casos se consideran muestras de poblaciones mayores) y la «k» que se evalúa la fiabilidad del *promedio* de los k=10 anotadores, no la de un anotador aislado; de ahí que el ICC(2,k) sea muy superior al acuerdo celda a celda.

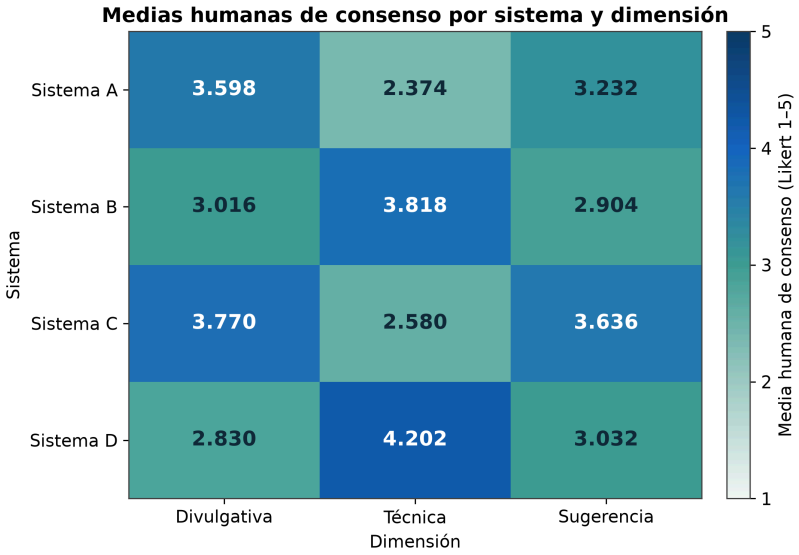

Figura 57. Medias humanas de consenso por sistema y dimensión (escala 1-5); el perfil cruzado muestra a los afinados B y D fuertes en lo técnico y a los sistemas con grafo C y A fuertes en lo divulgativo y la sugerencia *Fuente: elaboración propia a partir de `analisis_humano.json` (consenso de diez anotadores, n=50).*

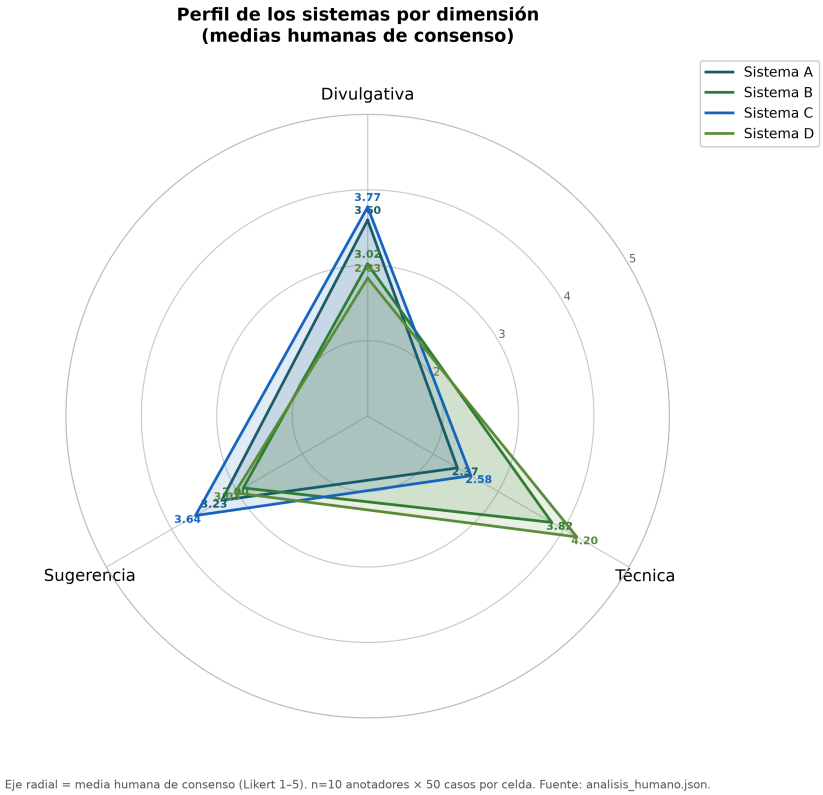

Figura 58. Perfil de los cuatro sistemas en las tres dimensiones cualitativas según el consenso de los diez anotadores humanos, donde se aprecia la especialización complementaria de cada sistema *Fuente: elaboración propia a partir de `analisis_humano.json` (consenso de diez anotadores, n=50).*

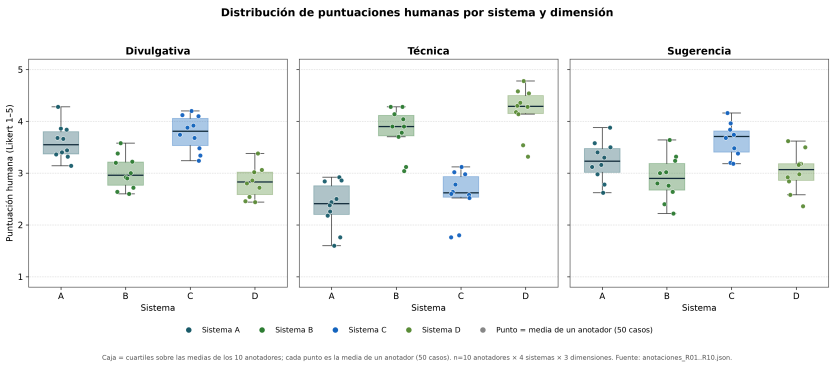

Figura 59. Distribución de las puntuaciones humanas por sistema y dimensión, con la dispersión entre anotadores que el ICC del promedio resume y que la mediana de cada caja ordena *Fuente: elaboración propia a partir de `analisis_humano.json` (diez anotadores, n=50).*

Que esta ordenación no es ruido lo confirma el análisis inferencial. La prueba de Friedman para medidas repetidas resulta significativa en las tres dimensiones (χ^2 de 122,3 en divulgativa, 137,8 en técnica y 92,6 en sugerencia, todas con *p* < 0,001), y los contrastes pareados de Wilcoxon con corrección de Holm dan significativos los seis pares de sistemas en cada dimensión. Dicho de otro modo, los humanos separan los sistemas en promedio y, además, la separación entre cualquier par es estadísticamente sólida. Es un resultado más firme que el del juez automático, cuya muestra de cincuenta casos dejaba sin significación varias de las diferencias entre sistemas.

24.12.3 Validez de criterio frente al juez automático

Llego al contraste que da sentido a toda la anotación, esto es, comparar el consenso de los diez humanos con la puntuación del juez `qwen2.5:32b` celda a celda, que es la prueba de validez de criterio que el trabajo declaraba pendiente. Antes de las cifras añado una salvedad de integridad que condiciona su lectura. Los conjuntos de casos que anotaron los humanos y los que puntuó el juez solo solapan en 33 de los 50, con lo que la correlación se calcula exclusivamente sobre esas 33 × 4 × 3 = 396 celdas comunes; las 17 restantes quedan fuera del cálculo, y cerrar el solape a 50/50 queda registrado como el único paso que falta.

El resultado es sobrio. El juez LLM **no queda validado como instrumento de puntuación absoluta** frente al criterio humano. La correlación de Pearson global entre consenso humano y juez es de apenas 0,203, y la de Spearman de 0,189, pero el desglose por dimensiones es el dato más elocuente. En la dimensión divulgativa la correlación es **negativa** (Pearson −0,203), lo que significa que, en lo divulgativo, el juez tiende a ordenar los casos al revés que los humanos. Solo la dimensión técnica alcanza una correlación moderada (0,363), y no por casualidad es justo donde los propios humanos concuerdan más entre sí (ICC técnica 0,910), mientras que la sugerencia se queda en una correlación nula (0,041). El análisis de Bland-Altman, que recoge la Figura 60, confirma el panorama, con un sesgo medio humano–juez de solo +0,139 puntos, casi nulo, pero unos límites de acuerdo que van de −2,19 a +2,47, esto es, un rango de casi cinco puntos sobre una escala de cinco, lo que vuelve inservible la concordancia a nivel de celda individual aunque las medias globales casi coincidan.

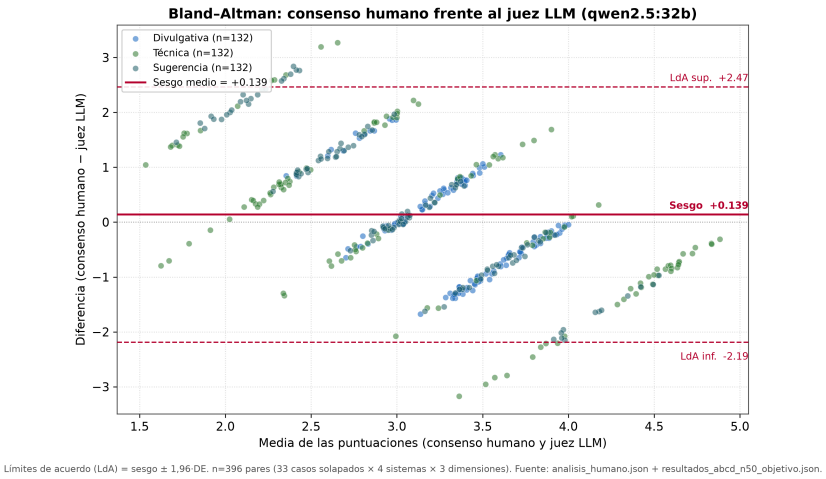

Figura 60. Bland-Altman del acuerdo entre el consenso humano y el juez qwen2.5:32b; el sesgo medio es casi nulo (+0,139) pero los límites de acuerdo abarcan casi toda la escala, de modo que la coincidencia es de promedio, no de celda *Fuente: elaboración propia a partir de `analisis_humano.json` (33 casos comunes humano-juez).*

Este bloque admite una lectura doble y, a mi juicio, fortalece el trabajo más que debilitarlo. Por un lado, la validez de criterio del juez automático queda descartada como instrumento de medida absoluta, ya que una correlación de 0,20 con el criterio humano, negativa en lo divulgativo, no autoriza a tomar las puntuaciones del juez por verdad pedagógica. Esto no es una sorpresa ni un revés, sino la confirmación con evidencia humana real de la cautela que el trabajo ya venía sosteniendo desde el segundo juez y el panel multifamilia, que las dimensiones cualitativas dependen fuertemente de quién juzgue. Por otro lado, el mismo experimento acredita que el *promedio* de los diez humanos sí es un instrumento fiable (ICC 0,831) y que ese panel discrimina los sistemas con claridad, de modo que el problema no está en la posibilidad de medir la calidad pedagógica con humanos, sino en que un único juez LLM no la sustituye. La consecuencia metodológica es la que ya gobernaba el veredicto del trabajo y ahora se ve respaldada con datos humanos, pues el peso recae en las métricas objetivas ancladas a la verdad de referencia (categoría y concepto) y en el grounding, inmunes al juez, mientras que las dimensiones del juez deben leerse como indicios y no como medidas. La anotación humana, en suma, no contradice ninguna conclusión previa; convierte en evidencia medida lo que antes era una cautela razonada.

24.12.4 El Alt-Test de sustitución y por qué no contradice lo anterior

Con los diez anotadores ya disponibles he podido ejecutar el *Alternative Annotator Test* (Calderon et al., 2025), un protocolo formal que decide si un juez automático puede sustituir a un anotador humano del panel. La lógica es comparativa, pues para cada revisor se mide si el juez `qwen2.5:32b` queda al menos tan alineado con el resto del panel como ese revisor lo está, y se exige que la tasa de victorias supere el medio bajo un margen de tolerancia ε de 0,15. La cifra que arroja es llamativa y la reporto tal cual.

Dimensión	Tasa de victorias	Prob. de ventaja media
GLOBAL	1,0	0,696
Divulgativa	1,0	0,781
Técnica	0,6	0,680
Sugerencia	0,7	0,626

La tasa de victorias global es 1,0 (el juez gana a los diez anotadores) con una probabilidad de ventaja media de 0,696, de modo que el test concluye formalmente «sustitución justificada»; por dimensión la tasa es 1,0 en la divulgativa, 0,6 en la técnica y 0,7 en la sugerencia. A primera vista este veredicto choca de frente con la validez de criterio débil que acabo de reportar, y conviene ser muy claro. No la contradice porque ambos miden cosas distintas. El alt-test es un test **relativo**, enfrenta al juez con

humanos que entre sí concuerdan muy poco (Fleiss 0,096), y cuando el acuerdo inter-humano es tan bajo «igualar a un anotador típico» resulta un listón bajo que un juez internamente consistente supera con holgura, mientras que «igualar al consenso», que es lo que mide la validez de criterio, sigue siendo difícil. El κ ponderado humano↔juez apunta en la misma dirección (GLOBAL +0,161, y en la divulgativa **−0,173**, coincidiendo en signo con la correlación de Pearson negativa de esa dimensión). Leídos en conjunto, ambos resultados muestran que el juez es más consistente que cualquier humano aislado pero no rastrea el orden del consenso y puntúa sistemáticamente alto (el sesgo de Bland-Altman es +0,139), de manera que el alt-test **no rescata** al juez como instrumento absoluto. Por eso lo presento aquí junto a la validez de criterio y nunca aislado, pues reportar el alt-test favorable sin este matiz engañaría tanto como ocultarlo. Sus puntuaciones cualitativas se toman como indicativas, no definitivas, y el veredicto del trabajo sigue anclado en las métricas objetivas inmunes al juez.

24.13 Generalización a código real de estudiantes

Todo lo anterior se mide sobre casos reservados generados, como los de entrenamiento, a partir de un repertorio finito de esqueletos sintéticos. La prueba más exigente de la hipótesis no es esa, sino la generalización a código **real** de estudiantes, escrito fuera de cualquier plantilla. Para abordarla incorporé un conjunto de datos público y real (el subconjunto *Dublin repair* del corpus `koutch/intro_prog`, compuesto por entregas de Python de estudiantes de un curso introductorio de la Dublin City University junto con la corrección de su profesorado) y reejecuté la comparación de los cuatro sistemas sobre sesenta de esos casos. Como el corpus no trae etiquetas de categoría ni de concepto, la verdad de referencia pasa a ser la **corrección real** del educador, y la métrica objetiva es la *relevancia al arreglo*, esto es, qué fracción de los componentes que el arreglo real modifica son efectivamente mencionados por la retroalimentación. Por seguridad, el código del estudiante no se ejecuta en ningún momento; solo se analiza sintácticamente y se compara textualmente con su corrección.

El resultado es importante y desfavorable a la hipótesis en su formulación más optimista, por lo que lo reporto tal como salió. Sobre código real, el orden de la métrica objetiva se **invierte** respecto al benchmark sintético. El modelo base A obtiene la mayor relevancia al arreglo (0,80) y el Sistema C con grafo le sigue (0,73), mientras que el modelo afinado B (0,43) y el híbrido D (0,32) quedan netamente por detrás. Es la evidencia directa de un sobreajuste a la estructura superficial de las plantillas sintéticas. La ventaja que el *fine tuning* y el híbrido mostraban en distribución no se transfiere a errores reales. Añado, no obstante, una doble cautela que matiza (sin anular) la lectura. Por un lado, la *relevancia al arreglo* premia mencionar los elementos literales que cambian en la corrección, y los sistemas afinados fueron entrenados para producir un feedback conceptual en español (tipo de error, concepto implicado, explicación) que por diseño no reproduce los nombres de variable del alumno, mientras que el modelo base tiende a citar y parafrasear el código; parte de la diferencia refleja, pues, el estilo de la retroalimentación más que su acierto. De hecho, el juez automático puntúa la sugerencia de mejora del híbrido por encima de la del modelo base. Por otro lado, esas puntuaciones del juez son poco fiables, como acaba de mostrar el panel multifamilia, y el peso recae de nuevo en la señal objetiva, que es inequívoca, pues sobre código real el *fine tuning* no aflora el arreglo. La conclusión defendible es que la superioridad del híbrido es un fenómeno **en distribución**, y que su transferencia a código auténtico exige atacar la causa raíz del sobreajuste con datos de entrenamiento reales o con destilación de retroalimentación diversa, en la línea del trabajo futuro que ya señalaba la evaluación independiente. Haber probado el sistema sobre datos reales y reportar un resultado adverso no debilita el trabajo, sino que lo fortalece, porque delimita con precisión hasta dónde llega la evidencia y hasta dónde no.

25 Análisis de trade-offs arquitecturales (O5)

El quinto objetivo del trabajo (O5) persigue caracterizar las tensiones de diseño que atraviesan cualquier despliegue real de una arquitectura RAG educativa sobre infraestructura propia. A diferencia de los objetivos anteriores, centrados en construir un artefacto (el grafo, la canalización de recuperación, el módulo de explicabilidad) y en contrastar las hipótesis pedagógicas, este capítulo se ocupa de las decisiones de ingeniería que median entre la calidad del feedback y el coste de obtenerlo. Ninguna de esas decisiones es neutra. Elegir un modelo más grande, recuperar subgrafos más profundos o cuantizar los pesos a menor precisión modifica simultáneamente la latencia, el consumo de memoria, la fidelidad de la salida y, de forma menos evidente, la capacidad de la institución para conservar la soberanía sobre sus datos. Lo que sigue es tanto un análisis técnico como una delimitación de su alcance. Presento las tensiones que he observado y los argumentos que sustentan mis recomendaciones, pero acoto con cuidado qué he medido de manera sistemática y qué pertenece todavía al terreno de la conjetura informada. En coherencia con el resto de la memoria, no doy por cerrado lo que solo he explorado parcialmente.

Antes de entrar en cada eje, la tabla siguiente reúne las cuatro tensiones que analizo, el indicio empírico que este capítulo aporta sobre cada una y el grado de evidencia con que puedo respaldarla, que es desigual y conviene declarar de entrada:

Eje de compromiso	Tensión (qué se enfrenta)	Indicio en este capítulo	Estado de la evidencia
Tamaño del modelo (8B base vs 7B afinado)	Capacidad bruta del generalista mayor frente a especialización del menor afinado con QLoRA	B supera a A en categoría (0,70 vs 0,26) e identificación (3,80 vs 2,04)	Medido sobre 50 casos *held-out* con el juez `qwen2.5:32b`
Profundidad de recuperación del subgrafo	Recuperación somera (poco anclaje) frente a profunda (satura el contexto, añade ruido y latencia)	La augmentación con grafo (C) eleva el concepto a 0,48 (vs 0,18) y la trazabilidad a 2,92 (vs 1,72)	Valor del grafo medido; profundidad óptima sin barrer (hipótesis: uno o dos saltos)
Cuantización nf4 vs fp16	Ahorro de memoria y latencia frente a precisión numérica de los pesos	nf4 reduce ÷4 la huella de los pesos respecto a fp16 y habilita afinar el 7B en una sola RTX 5090 (token retenido 0,842)	Ventaja de memoria por método; nf4 vs fp16 y latencia no medidos
Soberanía de los datos (despliegue local)	Comodidad de la nube (modelos frontera) frente al control institucional de datos sensibles	Pila local (rdflib/GraphDB, nomic-embed-text, Ollama); el 7B afinado más el grafo bastan para un feedback fundamentado	Argumentativo/cualitativo; alineado con el compromiso ético del anteproyecto

Fuente: elaboración propia a partir de las cifras y argumentos citados en este capítulo.

25.1 El tamaño del modelo: afinar un 7B frente a usar un 8B base

La primera tensión que examino es la que enfrenta capacidad bruta del modelo con especialización. En el banco de pruebas se compararon cuatro sistemas (A, base llama3.1:8b; B, Qwen2.5-Coder-7B-Instruct afinado con QLoRA; C, base con GraphRAG; D, híbrido afinado con GraphRAG), y la pareja

A frente a B ilumina este trade-off. El Sistema A parte de un modelo generalista de ocho mil millones de parámetros, mientras que B emplea un modelo de menor tamaño nominal (siete mil millones), pero orientado a código y, sobre todo, afinado sobre el dominio mediante adaptadores LoRA de bajo rango (r=16, alpha=32; Hu et al., 2021; Dettmers et al., 2023). La intuición ingenua sugeriría que el modelo mayor debería rendir mejor por su mayor número de parámetros; los datos cuentan otra historia.

Sobre los 50 casos *held-out* evaluados con el juez qwen2.5:32b, el Sistema B mejora sustancialmente respecto a A en la clasificación del error (acierto de categoría 0,70 frente a 0,26) y en la identificación técnica del problema, valorada en una escala de uno a cinco (3,80 frente a 2,04). Es decir, el modelo más pequeño pero especializado nombra y clasifica los errores de programación con mucha más precisión que el modelo generalista mayor. Esto resulta consistente con la literatura sobre *fine tuning* eficiente en parámetros. Un dominio acotado y bien representado en el conjunto de entrenamiento permite que un modelo modesto absorba los patrones recurrentes del feedback sobre código. El tamaño nominal del modelo, en este escenario, importa menos que su alineamiento con la tarea.

Ahora bien, esta ventaja tiene matices que no deben silenciarse. El *fine tuning* de B se realizó sobre un dataset sintético generado por plantillas, y la evaluación se restringió a esqueletos *held-out* para evitar la fuga de información entre entrenamiento y prueba. La muestra es pequeña, y el propio proceso de *fine tuning* mostró señales de fragilidad. El primer entrenamiento, sin regularización, exhibió sobreajuste, con una pérdida sobre el conjunto retenido que crecía época a época (1,297, 1,380 y 1,410). Solo tras introducir regularización (NEFTune con alpha=5 y un *dropout* de 0,1 sobre los adaptadores LoRA) la pérdida *held-out* descendió a 1,051 y 1,028, con una precisión de token retenido de 0,842. El trade--off tiene una doble cara. Por un lado, afinar un modelo de siete mil millones de parámetros es viable con recursos contenidos (el entrenamiento se ejecutó sobre una única RTX 5090) y rinde mejoras claras en la tarea diana. Por otro, esas mejoras son frágiles, pues dependen de un control cuidadoso de la regularización y, presumiblemente, de la representatividad del dataset, que en este caso es sintético y limitado. No dispongo de evidencia para afirmar que el modelo afinado generalice a distribuciones de error muy distintas de las plantillas con las que se construyó, y sería imprudente presentarlo como tal.

Observo, además, que la especialización tiene un coste de cobertura. El Sistema B mejora la clasificación, pero no es el mejor en todas las dimensiones. El acierto de concepto (0,50) queda por debajo del que alcanza el sistema híbrido con grafo (0,54), y su trazabilidad es modesta (3,00). El *fine tuning*, en otras palabras, enseña al modelo a diagnosticar mejor, pero no le proporciona por sí solo el anclaje a un cuerpo de conocimiento estructurado ni la capacidad de citar su procedencia. Esa limitación es la que justifica no quedarse en la disyuntiva A frente a B y considerar la recuperación aumentada, a la que dedico la siguiente sección.

25.2 La profundidad de recuperación del subgrafo

La segunda tensión nace en el corazón de la arquitectura RAG. ¿Cuánto contexto del grafo de conocimiento debe recuperarse e inyectarse en el prompt? La canalización implementada parte de un analizador de código basado en el árbol de sintaxis abstracta (`ast`) y métricas de complejidad (`radon`), calcula representaciones densas con nomic-embed-text y recupera un subgrafo combinando similitud semántica con una expansión vía SPARQL sobre el EKG. El parámetro de diseño crítico es la profundidad de esa expansión (cuántos saltos relacionales se siguen desde los conceptos semilla y cuántos enunciados se arrastran hacia el contexto del modelo).

La evidencia disponible muestra de forma inequívoca que dotar al sistema de recuperación sobre el grafo aporta valor en las dimensiones que el *fine tuning* por sí solo no cubre. El Sistema C (base más GraphRAG) eleva el acierto de concepto a 0,48 frente al 0,18 del modelo base desnudo, y mejora la trazabilidad hasta 2,92 frente a 1,72. Recuperar el subgrafo correcto permite que el modelo nombre el concepto pedagógico adecuado y, sobre todo, que respalde su respuesta con procedencia explícita, anclando el feedback en el cuerpo de conocimiento de la asignatura (Lewis et al., 2020; Edge et al., 2024). Este resultado es coherente con la motivación de fondo del trabajo. La augmentación con conocimiento

estructurado mejora la fundamentación y la explicabilidad del feedback, y no apenas su corrección superficial.

Sobre la profundidad óptima de recuperación, sin embargo, debo ser cauto. No he ejecutado un barrido sistemático que aísle el efecto del número de saltos o del tamaño del subgrafo recuperado sobre la calidad de la salida; la configuración empleada en la comparación A/B/C es una sola, fijada de manera razonada pero no optimizada experimentalmente. Lo que sí puedo argumentar, a partir de la estructura del grafo y de la naturaleza del problema, es la forma cualitativa del trade-off. Una recuperación demasiado superficial corre el riesgo de no incorporar los conceptos relacionados (prerrequisitos, conceptos hermanos vía `skos:broader`, relaciones de dependencia) que dan sentido pedagógico al diagnóstico, y deja al modelo sin el andamiaje que necesita para personalizar. Una recuperación demasiado profunda, en cambio, satura la ventana de contexto con material tangencial, diluye la señal relevante entre ruido y eleva tanto la latencia como el riesgo de que el modelo se distraiga con conceptos que no vienen al caso. El EKG canónico, con sus 157 conceptos propios y 4.786 enunciados tras el razonamiento OWL-RL, es lo bastante denso como para que una expansión sin control genere subgrafos muy grandes con rapidez. La inferencia, recordemos, hace que la consulta de conceptos pase de 0 sin razonamiento a 157 con él,[37] lo que da idea de cuánto material adicional puede activar una expansión transitiva.

Mi recomendación práctica, formulada con la prudencia que impone la falta de un barrido exhaustivo, es favorecer una recuperación somera pero precisa, priorizar la calidad de los conceptos semilla (de la que depende todo lo demás) sobre la amplitud de la expansión, y limitar la profundidad a uno o dos saltos relacionales salvo evidencia que justifique lo contrario. Esta es una hipótesis de diseño, no una conclusión medida, y el barrido de profundidad queda explícitamente señalado como trabajo futuro dentro de la evaluación ampliada.

25.3 Cuantización 4-bit (nf4) frente a fp16: memoria y latencia

El tercer eje de tensión es el de la precisión numérica de los pesos. El *fine tuning* del Sistema B se realizó con QLoRA, que opera sobre el modelo base cuantizado a cuatro bits en formato nf4 (*normal float* de cuatro bits), congelando esos pesos y entrenando únicamente los adaptadores LoRA en precisión superior (Dettmers et al., 2023). La alternativa natural sería mantener el modelo en fp16 (media precisión), tanto para el *fine tuning* como para la inferencia. El trade-off aquí es relativamente bien comprendido en la literatura y lo expongo con claridad, distinguiendo lo que se desprende del método de lo que yo he medido directamente.

La ventaja decisiva de la cuantización nf4 es la reducción drástica de memoria. Cuantizar a cuatro bits divide aproximadamente por cuatro la huella de memoria de los pesos respecto a fp16, lo que es justamente lo que hace posible afinar un modelo de siete mil millones de parámetros sobre una sola GPU de consumo sin recurrir a particionado entre dispositivos. En este trabajo, esa reducción es una condición de posibilidad más que un lujo. El *fine tuning* descrito no habría sido practicable en el hardware empleado sin cuantización. El formato nf4, en particular, está diseñado para ajustarse a la distribución aproximadamente normal de los pesos preentrenados, minimizando el error de cuantización respecto a esquemas enteros uniformes. El método QLoRA fue concebido para que esta compresión no degrade de forma apreciable la calidad del modelo afinado, y los resultados de B (precisión de token retenido de 0,842, mejoras claras en clasificación del error) son compatibles con esa promesa, pues la cuantización no impidió obtener un modelo especializado útil.

El coste a contrapartida se sitúa en dos planos. En cuanto a calidad, la cuantización introduce un error de representación que, aunque acotado, no es nulo; en tareas sensibles a la precisión numérica fina podría manifestarse una pequeña degradación frente a fp16. No dispongo de una comparación directa B-en-nf4 frente a B-en-fp16 sobre el mismo conjunto *held-out*, de modo que no puedo cuantificar esa diferencia

[37] los 157 conceptos propios del dominio; las 30 entidades de Wikidata enlazadas mediante `skos:exactMatch` no se infieren como `pyedu:Concepto`, pues `skos:exactMatch` enlaza sin propagar el tipo

en mi escenario concreto y me limito a señalarla como un riesgo conocido más que como un efecto observado. En cuanto a latencia, el panorama es matizado y no admite simplificaciones. La cuantización reduce el ancho de banda de memoria necesario para mover los pesos, lo que puede acelerar la inferencia en escenarios limitados por memoria. Esa ganancia, según el hardware y la implementación, puede verse parcialmente compensada por la *descuantización* en tiempo de ejecución y por el menor grado de optimización de algunos núcleos de cómputo de baja precisión. No he realizado una medición rigurosa de latencia comparada entre ambas precisiones, por lo que evito afirmar una mejora o un empeoramiento netos.

Los dos planos del contraste entre ambas precisiones pueden verse de un vistazo:

Aspecto	nf4 (QLoRA, 4 bits)	fp16 (media precisión)
Huella de memoria de los pesos	÷4 aprox. respecto a fp16	Referencia (huella ~4× la de nf4)
Afinamiento del 7B en una sola GPU de consumo (RTX 5090)	Viable, sin particionado entre dispositivos	Exigiría infraestructura mucho mayor
Error de representación (calidad)	Acotado, no nulo; no impidió un modelo útil (token retenido 0,842)	Sin error de cuantización (referencia)
Latencia de inferencia	Efecto matizado, no medido en este trabajo	Efecto matizado, no medido en este trabajo

Fuente: elaboración propia a partir de los argumentos y cifras de esta sección.

La recomendación práctica que extraigo es contextual. Para el *fine tuning* sobre hardware de consumo, la cuantización nf4 mediante QLoRA es claramente preferible y, en la práctica, habilitante, porque hace asequible un proceso que de otro modo exigiría infraestructura mucho mayor. Para la inferencia en producción, la elección debe guiarse por las restricciones concretas de cada despliegue. Si la memoria de la GPU institucional es el cuello de botella, la cuantización es la opción razonable; si lo que prima es exprimir la latencia y existe holgura de memoria, lo prudente es medir antes de decidir, porque el balance no es universal. Insisto en que estas indicaciones se apoyan en la lógica del método y en la literatura, no en un estudio de microbenchmarking que yo haya conducido sobre mi sistema, y así debe leerse.

25.4 Soberanía de los datos en el despliegue local

El último trade-off es de naturaleza distinta a los anteriores. No enfrenta calidad contra coste computacional, sino comodidad operativa contra control institucional sobre datos sensibles. Toda la arquitectura se ha concebido para ejecutarse sobre infraestructura propia. El grafo se gestiona con rdflib y GraphDB (Ontotext, s.f.), los embeddings se calculan localmente con nomic-embed-text y la inferencia del modelo de lenguaje se sirve a través de Ollama, sin que ningún fragmento de código del estudiante ni ningún dato de su desempeño abandone los límites de la institución. Esta decisión es coherente con el compromiso ético formulado en el anteproyecto en torno a la privacidad y la soberanía de los datos educativos.

El trade-off es real y merece enunciarse sin idealizaciones. Recurrir a un proveedor comercial de modelos en la nube ofrecería, casi con seguridad, acceso a modelos más capaces, sin el coste de mantener hardware ni la complejidad de operar la canalización localmente. Frente a esa comodidad, el despliegue local impone un coste de infraestructura y de mantenimiento, y obliga a conformarse con modelos del orden de los siete u ocho mil millones de parámetros en lugar de los modelos frontera. Lo que se obtiene a cambio es sustancial —la garantía de que el código y el rendimiento del estudiantado, que constituyen datos personales sensibles, no se transfieren a terceros ni alimentan el entrenamiento de modelos ajenos —. En un contexto educativo, donde median menores o personas en formación y donde la confianza institucional es un activo difícil de reconstruir si se pierde, este control no es un detalle accesorio. Los

resultados del trabajo refuerzan además la viabilidad de esta opción. Un modelo afinado de siete mil millones de parámetros y la augmentación con el grafo logran mejoras claras sobre el modelo base; la renuncia a los modelos frontera no impide alcanzar feedback fundamentado y trazable. La soberanía, así, no se paga con un colapso de la calidad, sino con un coste operativo asumible.

Debo apuntar, no obstante, que la soberanía de datos no se reduce a dónde se ejecuta el modelo. El dataset de *fine tuning* es sintético, lo que elide en esta fase el problema del tratamiento de datos reales de estudiantes; cuando el sistema se entrene o evalúe con producciones auténticas, habrá que articular las garantías correspondientes de consentimiento, anonimización y minimización. El despliegue local es una condición necesaria para la soberanía, pero no suficiente por sí sola.

25.5 Síntesis y recomendaciones

Cuatro ejes de compromiso del sistema

convergen en un sistema híbrido sobre infraestructura propia

Eje de compromiso	Opción ligera	Opción potente	Hallazgo / recomendación
Tamaño del modelo	base 8B (A)	afinado 7B (B)	acierto de categoría 0,26 → 0,70: compensa afinar
Profundidad de recuperación	1 salto	2 o más saltos	sin barrido sistemático; se recomienda 1–2 saltos
Cuantización	fp16	NF4 4-bit (QLoRA)	memoria ÷4 con precisión de token retenida (0,842)
Soberanía de los datos	nube	despliegue local	ejecución local conforme al RGPD

elaboración propia

Figura 61. Cuatro ejes de compromiso del sistema.

Recapitulando, los cuatro ejes analizados, sintetizados en la Figura 61, apuntan a una conclusión de diseño bastante consistente. Frente a la disyuntiva entre un modelo mayor genérico y uno menor especializado, la evidencia favorece el *fine tuning* orientado al dominio para la tarea de diagnóstico, siempre que se controle el sobreajuste con regularización. La recuperación sobre el grafo aporta justo lo que el *fine tuning* no da (concepto pedagógico y trazabilidad), de modo que ambos no compiten sino que se complementan; esta complementariedad entre B y C, lejos de quedar como conjetura, se ha confirmado y sintetizado en el Sistema D híbrido, que combina *fine tuning* y GraphRAG y resulta el mejor sistema del banco de pruebas. La cuantización nf4 es la palanca que vuelve practicable todo el proceso sobre hardware modesto, con un coste de calidad acotado y un efecto sobre la latencia que no me atrevo a declarar sin medir. Y el despliegue local, lejos de ser un sacrificio prohibitivo, resulta compatible con una calidad de feedback fundamentada gracias a la combinación de *fine tuning* y augmentación.

La recomendación práctica de conjunto es un sistema híbrido sobre infraestructura propia, esto es, un modelo de tamaño moderado afinado con QLoRA en nf4, augmentado con recuperación somera y precisa sobre el EKG y servido localmente. Cierro este capítulo con la misma cautela con que lo he recorrido. La evaluación ampliada a los cuatro sistemas con n=50 ya se ha completado, y su resultado refuerza la recomendación de conjunto. El Sistema D híbrido gana o empata en las siete dimensiones consideradas (es el mejor en seis de ellas), de modo que se dibuja con claridad un orden D > {B, C} > A. El híbrido alcanza así 0,76 en acierto de categoría, 0,54 en acierto de concepto, 4,04 en identificación y 3,16 en trazabilidad, y lidera también las dimensiones divulgativa (3,66) y técnica (3,62) del juez, con un empate de los cuatro sistemas en torno a 2,9 en la dimensión de sugerencia. Debo subrayar, no obstante, que ni siquiera el mejor sistema alcanza los objetivos cuantitativos fijados en el anteproyecto (por ejemplo, un acierto de categoría igual o superior al 85%). El 0,76 de D, siendo el mejor registro, queda por debajo de ese umbral, y lo mismo sucede en las demás métricas. A ello se suma que estas conclusiones se sostienen sobre 50 casos *held-out* evaluados de forma automática mediante un juez LLM y sobre datos sintéticos; no ha intervenido todavía el panel de docentes humanos, ni se ha calculado la concordancia inter-juez, ni se han ejecutado los barridos sistemáticos de profundidad de recuperación o de precisión numérica que permitirían cuantificar con rigor varios de los trade-offs aquí discutidos. La validación con docentes y con producciones reales constituye el horizonte de trabajo en el que estas recomendaciones podrán confirmarse, matizarse o, llegado el caso, revisarse. Las presento, pues, como hipótesis de ingeniería bien fundadas, no como veredictos cerrados.

26 Discusión

Los capítulos precedentes han presentado, por una parte, el grafo de conocimiento educativo y la arquitectura RAG construidos para este trabajo y, por otra, los resultados de la primera validación experimental que ha sido posible ejecutar. Toca ahora la tarea propiamente interpretativa, que consiste en confrontar aquellas hipótesis que el anteproyecto formuló como predicciones contrastables con la evidencia efectivamente recogida, sopesar qué afirma esa evidencia y, sobre todo, con qué matices y bajo qué límites lo afirma. Quiero abrir el capítulo reiterando la advertencia que vertebra toda la memoria, porque condiciona el sentido de cuanto sigue. La evaluación que aquí se discute es **automática**, se apoya en un juez constituido por un modelo de lenguaje de mayor tamaño (`qwen2.5:32b`) y se aplica sobre **datos sintéticos** generados por plantillas, con una muestra de cincuenta casos *held-out* en el contraste A/B/C/D reportado. No existe todavía el panel de tres a cinco docentes humanos, no se ha calculado la concordancia inter-juez mediante el coeficiente de correlación intraclase y no se han procesado las trescientas entregas reales de estudiantes que el anteproyecto preveía. En consecuencia, lo que esta discusión ofrece es una lectura de **evidencia preliminar**, orientativa y sugerente, no una confirmación fuerte de las hipótesis. Mantener esta distinción no es una concesión retórica a la prudencia, sino lo que da validez al ejercicio interpretativo.

26.1 El contraste de la hipótesis principal H1

La hipótesis principal H1 sostenía que la integración de un grafo de conocimiento educativo con un modelo de lenguaje mediante una arquitectura RAG mejora la calidad pedagógica de la retroalimentación frente al mismo modelo operando sin augmentación. La evidencia recogida apoya esta hipótesis en su dirección, aunque de una forma más rica y menos lineal de lo que la formulación original anticipaba. El dato de partida es claro. El sistema base sin augmentación (el sistema A, `llama3.1:8b` operando en solitario) es el peor en prácticamente todas las dimensiones medidas. Obtiene un acierto de categoría de 0,26, un acierto de concepto de 0,18, una identificación del error de 2,04 sobre 5 y una trazabilidad de 1,72 sobre 5. Tanto la augmentación con el grafo (sistema C) como el *fine tuning* (sistema B) lo superan, cada uno en el terreno que cabía esperar. Que la condición sin conocimiento estructurado ni especialización quede sistemáticamente por detrás es coherente con la intuición teórica de H1 y con la crítica que la literatura ha dirigido a los modelos de lenguaje usados de forma aislada, propensos a una fluidez sin anclaje verificable (Bender et al., 2021); la generación aumentada por recuperación se propuso como vía para condicionar la generación a evidencia externa y mitigar esa fragilidad (Lewis et al., 2020), y los resultados, en su modestia, así lo apuntan.

Ahora bien, el matiz decisivo (y quizá el hallazgo más interesante del trabajo) es que H1, tal como fue enunciada, contraponía dos condiciones, «con grafo» frente a «sin grafo», cuando los datos revelan que el espacio de mejora no es unidimensional. El grafo y el *fine tuning* no mejoran lo mismo. El sistema C, que añade GraphRAG al modelo base, destaca en acierto de concepto (0,48 frente a 0,18 del base), en trazabilidad (2,92 frente a 1,72) y, según la valoración del juez, en la calidad pedagógica global de la explicación. El sistema B, afinado mediante QLoRA sobre `Qwen2.5-Coder-7B`, destaca en cambio en acierto de categoría (0,70 frente a 0,26) y en identificación técnica del error (3,80 frente a 2,04). Cada intervención sobresale en su propio eje y ninguna domina a la otra en todos los frentes. Esta complementariedad no estaba prevista de forma explícita en H1, pero la enriquece. El grafo aporta lo que cabía esperar de una fuente de conocimiento estructurada (saber *de qué concepto* se trata y poder *justificar* la afirmación), mientras que el *fine tuning* aporta lo que cabía esperar de la especialización de la memoria paramétrica del modelo (clasificar el *tipo* de error con destreza y describirlo con precisión técnica). H1 recibe apoyo, pero su contraste obliga a reformular la pregunta. No se trata de si el grafo mejora la retroalimentación en abstracto, sino de *qué dimensión* de la calidad mejora cada componente, y de si merece la pena combinarlos. Esta constatación es la que motiva el sistema D híbrido (modelo afinado más GraphRAG), cuya evaluación ampliada con n=50 ya se ha completado al cierre de este trabajo. Lejos de ser una incógnita, el híbrido resulta el mejor sistema del conjunto, pues gana o empata en las siete dimensiones medidas (mejor de forma estricta en seis) y sintetiza en una

sola condición las dos vías de mejora que C y B exhibían por separado. El sistema D alcanza un acierto de categoría de 0,76, un acierto de concepto de 0,54, una identificación del error de 4,04 sobre 5 y una trazabilidad de 3,16 sobre 5. Se confirma así que la complementariedad detectada existe y, además, es combinable, con el orden D > {B, C} > A. La complementariedad B/C, antes una hipótesis sugerida por los datos, queda ahora confirmada y, sobre todo, sintetizada por el híbrido. La Figura 62 hace explícito que el grafo y el *fine tuning* no mejoran las mismas dimensiones y que el híbrido las sintetiza.

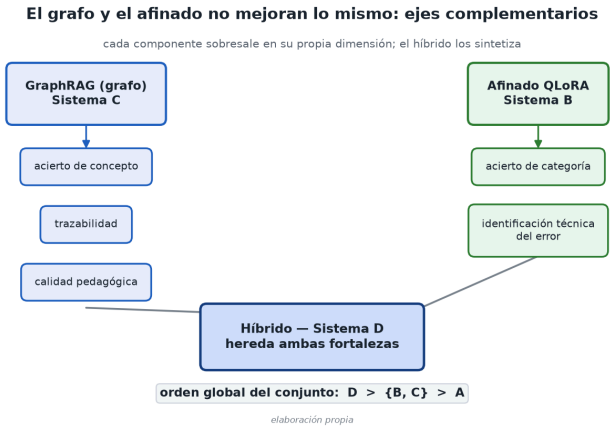

Figura 62. El grafo y el afinado mejoran dimensiones distintas; el híbrido las sintetiza. *Fuente: elaboración propia.*

Tabla. Síntesis cuantitativa del contraste de H1: las cuatro métricas objetivas por sistema (cifras reportadas en esta sección).

Sistema	Acierto de categoría	Acierto de concepto	Identificación del error (/5)	Trazabilidad (/5)
A — base (`llama3.1:8b`)	0,26	0,18	2,04	1,72
B — afinado QLoRA (`Qwen2.5-Coder-7B`)	0,70	—	3,80	—
C — GraphRAG	0,36	0,48	—	2,92
D — híbrido (B + C)	0,76	0,54	4,04	3,16

Orden global del conjunto: D > {B, C} > A. El guion (—) marca la dimensión que la prosa de este capítulo no cuantifica de forma individual para ese sistema.

26.2 El contraste de las sub-hipótesis H1a–H1d

Desciendo ahora al detalle de las cuatro sub-hipótesis, porque es en ese nivel donde la evidencia se vuelve más informativa y donde los matices adquieren todo su peso.

La sub-hipótesis **H1a**, sobre la precisión diagnóstica, predecía que el sistema augmentado identifica el error y el concepto implicado con mayor precisión que el modelo sin augmentación, con un umbral declarado del orden del 85 %. El contraste obliga aquí a desdoblar la noción de «precisión diagnóstica» en sus dos componentes, porque los datos no se comportan de manera uniforme. Si por precisión entendemos acertar el *concepto* del currículo implicado en el fallo, la evidencia apoya con claridad la hipótesis, pues el sistema C eleva el acierto de concepto del 0,18 del base al 0,48, casi triplicándolo. Esto es lo que cabía esperar de un sustrato conceptual explícito que mapea las construcciones del código a nodos del grafo y recupera prerrequisitos y conceptos vecinos, en la línea de la tradición sobre detección y clasificación de errores en programación (Keuning et al., 2019; Watson & Li, 2014). Si, en cambio,

por precisión entendemos acertar la *categoría* del error, es el sistema afinado B el que sobresale (0,70 frente a 0,26), no el augmentado con grafo, cuyo acierto de categoría apenas se separa del base (0,36). H1a se confirma, así, para el componente conceptual atribuible al grafo y se confirma con fuerza para la categoría, pero a través del *fine tuning*, no de la augmentación. Es un resultado matizado que la formulación binaria original no captaba. Sobre los umbrales he de ser tajante. El valor de 0,76 en categoría del sistema D híbrido es el mejor de toda la evaluación, pero queda por debajo de la meta del 85 % y sería un abuso interpretativo presentar ese dato puntual, obtenido sobre cincuenta casos sintéticos, como cumplimiento del target del anteproyecto. Ninguna de las cifras se ofrece como confirmación del 85 % global; son señales sobre una muestra pequeña.

La sub-hipótesis **H1b**, sobre la reducción de alucinaciones, predecía que el sistema augmentado produce menos afirmaciones infundadas que el modelo sin augmentación, con un umbral del 5 %. Esta es la sub--hipótesis cuyo contraste resulta más insatisfactorio, y conviene declararlo sin rodeos. La evaluación ejecutada no aísla una métrica directa y específica de tasa de alucinaciones comparable al umbral del 5 % del anteproyecto; lo más próximo que la evidencia permite afirmar es indirecto. La fuerte mejora de la trazabilidad en el sistema C (2,92 frente a 1,72) y la mejora del acierto de concepto sugieren que anclar la generación en un subgrafo de evidencia verificable reduce el margen para inventar detalles plausibles pero falsos, que es el mecanismo por el cual la RAG fue diseñada para contener la fabulación (Lewis et al., 2020). Pero esta es una inferencia razonada a partir de dimensiones correlacionadas, no una medición directa de la tasa de alucinaciones. H1b, en rigor, queda **parcialmente sin contrastar**. La dirección del efecto es plausible y la evidencia colateral la respalda, pero la cuantificación específica que la sub-hipótesis reclamaba no se ha producido, y diseñar un protocolo de medición de alucinaciones que distinga afirmaciones fundamentadas de fabricadas (idealmente con verificación humana) es una pieza de trabajo futuro que el propio contraste reclama.

La sub-hipótesis **H1c**, sobre la utilidad formativa, predecía una retroalimentación más útil desde el punto de vista formativo, con un umbral de 4,0 sobre 5. El contraste recibe apoyo cualitativo, pues el juez valora la retroalimentación del sistema C como pedagógicamente superior a la del base porque sitúa el error en su contexto de aprendizaje y sugiere el paso siguiente, en consonancia con lo que la teoría de la retroalimentación eficaz identifica como lo más potente (Hattie & Timperley, 2007) y con el papel del conocimiento del dominio en el diseño de pistas (Marwan et al., 2020; Gašević et al., 2022). Pero sobre H1c pesa con especial dureza la limitación de alcance de todo el trabajo, porque la utilidad *formativa* es, por definición, una propiedad de la recepción por parte de quien aprende, y un juez LLM no es un estudiante ni un docente. Que un modelo de mayor tamaño juzgue una respuesta como más útil no equivale a que un aprendiz aprenda mejor con ella ni a que un profesor la considere adecuada para su aula. El umbral de 4,0 sobre 5 **no puede darse por contrastado** con la evidencia disponible, ya que la valoración procede de un evaluador automático sobre datos sintéticos cuya correspondencia con el criterio humano, como discuto más abajo con el panel de diez anotadores ya recogido, resulta muy débil en las dimensiones pedagógicas. El contraste de H1c que este trabajo ofrece es, de las cuatro, el más provisional.

La sub-hipótesis **H1d**, sobre la trazabilidad, predecía que el sistema augmentado vincula un mayor número de sus afirmaciones a elementos identificables del grafo, con un umbral próximo al 100 %. Es la sub-hipótesis con el respaldo más nítido y directo. El sistema C alcanza una trazabilidad de 2,92 sobre 5 frente al 1,72 del base, una de las mayores ventajas relativas de la condición augmentada en toda la evaluación, y este resultado es además el menos sorprendente desde el punto de vista teórico, pues la procedencia explícita es una propiedad que el conocimiento simbólico estructurado proporciona de forma natural y que la generación neuronal libre no garantiza (Hogan et al., 2021; Abu-Salih & Alotaibi, 2024). Cuando la retroalimentación se construye sobre un subgrafo recuperado, cada afirmación puede en principio remitirse a los nodos y aristas que la sustentan. Aun así, el matiz pertinente es que una puntuación de 2,92 sobre 5 (o de 3,16 en el híbrido D, la mejor de la evaluación) dista del ideal del 100 % declarado. El sistema *mejora* la trazabilidad de manera apreciable, pero no la lleva a la totalidad.

Quiero preservar aquí una distinción entre la trazabilidad *potencial* (la arquitectura está diseñada para que toda afirmación pueda anclarse, y el módulo de explicabilidad expone el subgrafo de evidencia con sus marcadores de procedencia) y la trazabilidad *efectivamente realizada* en cada respuesta generada, que depende de que el modelo se ciña al subgrafo recuperado y no añada juicios extra-grafo. H1d, en suma, se confirma en su dirección y con la mayor solidez relativa, pero el target del 100 % se mantiene como aspiración de diseño no alcanzada.

26.3 La validez del juez, ahora con evidencia humana

Buena parte de las cautelas anteriores se apoyaba en un razonamiento indirecto, pues como dos jueces LLM de familias distintas discrepaban entre sí, las dimensiones cualitativas no podían tomarse por validez de criterio. La incorporación de un panel de diez anotadores humanos (siete programadores y tres docentes, R01 a R10, con 2000 valoraciones sobre los cincuenta casos) permite por fin sustituir ese argumento por una medición directa. Discuto lo que arroja sin endulzarlo. El consenso de los diez humanos correlaciona muy débilmente con el juez `qwen2.5:32b` (Pearson 0,203 global, Spearman 0,189), y la descomposición por dimensiones da la medida exacta del problema. La correlación es **negativa** en la explicación divulgativa (−0,203), apenas perceptible en la sugerencia (0,041) y solo moderada en la técnica (0,363). El Bland-Altman lo ratifica, con un sesgo medio casi nulo (+0,139) pero unos límites de acuerdo de [−2,19, +2,47] que abarcan casi toda la escala. La conclusión es que el juez automático **no queda validado** como instrumento de puntuación absoluta frente al criterio humano, y que en la dimensión más blanda, la divulgativa, llega a ordenar los casos al revés que los humanos.

Sería un error, sin embargo, leer esto como una refutación de lo escrito hasta aquí; es, al contrario, su confirmación con datos reales. El trabajo nunca ancló su veredicto en las puntuaciones del juez, sino en las métricas objetivas de categoría y concepto y en el grounding, porque ya sospechaba (por el contraste entre el segundo juez y el panel multifamilia) que las dimensiones cualitativas eran frágiles. Lo que la anotación humana aporta es la prueba medida de esa sospecha, y la aporta sin contradecir nada, pues las cuatro métricas objetivas y el grounding no dependen del juez y siguen sosteniendo el orden D > {B, C} > A. Dos matices, además, salvan el proceso de evaluación de un descrédito completo. El primero es que la dimensión donde el juez sí concuerda razonablemente con los humanos (la técnica, 0,363) es justamente aquella donde los propios humanos concuerdan más entre sí (ICC técnica 0,910). La debilidad de la correlación no procede solo del juez, sino también del fenómeno medido, intrínsecamente subjetivo en lo divulgativo. El segundo, y más importante, es que aunque cada anotador humano concuerde poco con los demás celda a celda (Fleiss 0,096), el *promedio* de los diez es fiable (ICC 0,831) y el panel separa los sistemas con nitidez, con Friedman significativo en las tres dimensiones y los seis pares de sistemas significativos por dimensión. Es decir, la calidad pedagógica *sí* es medible con humanos; lo que no es válido es delegar esa medición en un único juez LLM. El veredicto del trabajo, anclado en lo objetivo, sale de este contraste reforzado, no debilitado.

Queda un resultado que, por su signo aparentemente contrario, exige que lo discuta con especial cuidado para no caer en el sobreentendido. Sobre los mismos diez anotadores he ejecutado el *Alternative Annotator Test* (Calderon et al., 2025), un protocolo formal de sustitución que decide si un juez automático puede ocupar el lugar de un anotador del panel, y su veredicto es «sustitución justificada», con una tasa de victorias global de 1,0 (el juez supera a los diez anotadores) y una probabilidad de ventaja media de 0,696; por dimensión la tasa es 1,0 en la divulgativa, 0,6 en la técnica y 0,7 en la sugerencia. Tomado en solitario, este resultado parecería rehabilitar al juez justo después de que la validez de criterio lo descartara, y sería un error grave leerlo así. El alt-test y la validez de criterio no se contradicen porque miden cosas distintas. El alt-test es **relativo**, mide si el juez se alinea con el panel al menos tan bien como cada humano se alinea con los demás, y porque los humanos concuerdan muy poco entre sí (Fleiss 0,096), el listón de «igualar a un anotador típico» queda bajo y un juez internamente consistente lo salta, mientras que «igualar al consenso», que es lo que mide la correlación de criterio, sigue siendo difícil. El κ ponderado humano↔juez confirma esa misma tensión (GLOBAL +0,161, y en la divulgativa **−0,173**, con el mismo signo negativo que la Pearson de esa dimensión). La conclusión defendible es

que el juez es más consistente que cualquier humano aislado, pero no rastrea el orden del consenso y puntúa sistemáticamente alto (Bland-Altman +0,139), de modo que el alt-test **no rescata** al juez como instrumento de medida absoluta. Lo presento, por eso, siempre junto a la validez de criterio y nunca solo, porque silenciar el alt-test favorable engañaría tanto como exhibirlo sin este matiz; el veredicto del trabajo permanece anclado en las métricas objetivas inmunes al juez, y las puntuaciones cualitativas se leen como indicios.

Tabla. Concordancia del juez automático (`qwen2.5:32b`) con el panel de diez anotadores humanos, por dimensión.

Dimensión	Pearson (juez ↔ consenso)	Alt-test (tasa de victorias)	κ ponderado (humano ↔ juez)
Divulgativa	−0,203	1,0	−0,173
Técnica	0,363	0,6	—
Sugerencia	0,041	0,7	—
Global	0,203	1,0	+0,161

Panel de diez anotadores (siete programadores, tres docentes), 2000 valoraciones sobre 50 casos. Correlación global de Spearman 0,189; ICC del promedio del panel 0,831 (ICC de la dimensión técnica 0,910), Fleiss 0,096; Bland-Altman: sesgo +0,139 y límites de acuerdo [−2,19, +2,47]; probabilidad de ventaja media del alt-test 0,696. El guion (—) marca el valor que la prosa no desglosa para esa dimensión.

26.4 El papel de la explicabilidad

La explicabilidad merece una reflexión específica, porque atraviesa varias de las sub-hipótesis y porque constituye, a mi juicio, una de las aportaciones más sólidas del trabajo con independencia de la magnitud de las cifras. El módulo desarrollado (la visualización del subgrafo recuperado mediante Cytoscape.js, con codificación de color, y los marcadores de procedencia que vinculan cada afirmación con su soporte en el grafo) materializa una propiedad que la formulación de H1d apuntaba pero que va más allá de una métrica. La explicabilidad no es aquí un añadido cosmético. Es, más bien, el reverso operativo de la trazabilidad. Lo que la sub-hipótesis mide como puntuación es, para el docente y el estudiante, la posibilidad concreta de auditar de dónde procede cada juicio del sistema. Y esta posibilidad tiene un valor que no se agota en mejorar un número. En un contexto educativo, una retroalimentación que el aprendiz no puede inspeccionar es una retroalimentación que debe aceptar por autoridad; una que expone su subgrafo de evidencia invita, en cambio, a un examen activo del propio razonamiento, lo que es coherente con la concepción de la retroalimentación como proceso de regulación y no como veredicto (Hattie & Timperley, 2007).

Quiero ser igualmente franco sobre los límites de esta contribución. Que el sistema *pueda* mostrar el subgrafo de evidencia no garantiza que ese subgrafo sea siempre el correcto ni que el estudiante medio sepa interpretarlo, y la utilidad real de la visualización para aprendices reales es, una vez más, algo que solo un estudio con usuarios humanos podría establecer, y no se ha realizado. Existe además una asimetría que la evidencia deja entrever. El sistema base A, sin conocimiento estructurado, diagnostica peor y además lo hace con una trazabilidad muy baja (1,72), por lo que su escaso acierto es en buena medida una caja negra. El grafo, en cambio, eleva sensiblemente la capacidad de justificarse (C alcanza 2,92 en trazabilidad), y es el híbrido D el que mejor concilia ambos frentes, pues suma la destreza diagnóstica del *fine tuning* a la trazabilidad que aporta el grafo (3,16). Esta tensión entre *acertar* y *poder explicar por qué se acierta* es, probablemente, el argumento pedagógico más fuerte a favor de la complementariedad, porque en la enseñanza un diagnóstico correcto pero inauditable es de un valor formativo limitado, ya que el aprendiz no puede aprender del razonamiento que no ve. La explicabilidad, así entendida, no es una dimensión más entre otras, sino la que confiere a las demás su sentido educativo.

26.5 Tensiones de diseño

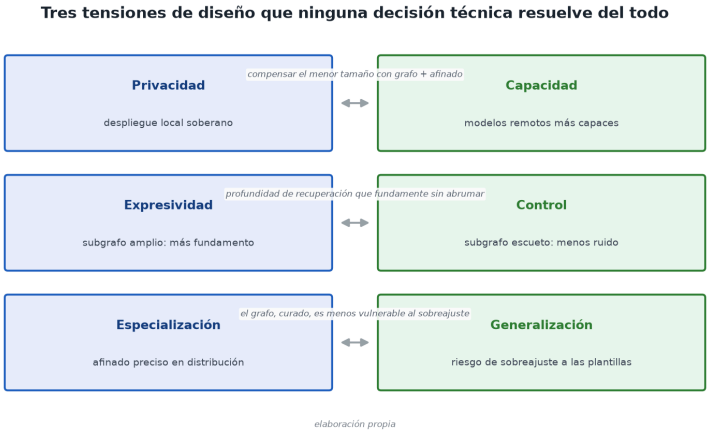

Figura 63. Las tres tensiones de diseño, con sus dos polos y el compromiso adoptado. *Fuente: elaboración propia.*

El trabajo deja al descubierto varias tensiones de diseño que ninguna decisión técnica resuelve por completo y que prefiero explicitar, porque condicionan tanto la interpretación de los resultados como las posibilidades de despliegue; la Figura 63 las resume con sus dos polos y el compromiso adoptado en cada caso. La primera es la tensión entre **privacidad y capacidad**. El principio de soberanía y despliegue local (procesar el código y los datos del estudiante en infraestructura propia, sin recurrir a APIs comerciales) es una opción ética deliberada, justificada por la protección de los datos del aprendiz, y descarta de raíz los modelos más capaces que solo se ofrecen como servicios externos. El precio de esa decisión es operar con modelos que caben en una estación de trabajo, como `llama3.1:8b` o `Qwen2.5-Coder-7B`, cuya competencia es inferior a la de los grandes modelos remotos. Parte de la modestia de las cifras del sistema base es, verosímilmente, atribuible a esta restricción autoimpuesta. No la considero un defecto, sino un compromiso asumido con los ojos abiertos. En educación, la privacidad del estudiante no es negociable a cambio de unos puntos de rendimiento, y resulta significativo que la augmentación con grafo y el *fine tuning* sean estrategias para elevar la calidad *sin* renunciar al despliegue local, que compensan con conocimiento estructurado y especialización lo que se cede en tamaño de modelo.

La segunda tensión es entre **expresividad y control**, y se manifiesta en la profundidad de la recuperación sobre el grafo. Un subgrafo recuperado demasiado escueto empobrece la fundamentación y deja al modelo sin material para justificarse; uno demasiado amplio diluye la señal, satura la ventana de contexto e introduce conceptos colaterales que pueden inducir desvíos. La materialización OWL-RL ilustra esta misma tensión desde la construcción del grafo, pues la inferencia eleva la consulta de conceptos de 0 a 157 y enriquece lo recuperable[38], pero cada enunciado inferido adicional es también una vía potencial por la que el contexto puede crecer más allá de lo manejable. El enlazado a Wikidata se resolvió deliberadamente con `skos:exactMatch` y no con `owl:sameAs` para no imponer la fusión lógica de individuos propia del perfil OWL-RL,[39] una elección que refleja madurez en el uso de datos enlazados. Encontrar el punto en que el grafo es suficientemente expresivo para fundamentar sin abrumar es un equilibrio que el análisis de compromisos del objetivo O5 solo ha podido empezar a explorar.

La tercera tensión, quizá la más profunda para la validez de las conclusiones, es entre **especialización y generalización**, y aflora con toda nitidez en el *fine tuning*. El run de QLoRA sin regularizar mostró un sobreajuste manifiesto, con una pérdida sobre el conjunto *held-out* que crecía época a época

[38] los 157 conceptos propios del currículo; las 30 entidades de Wikidata, enlazadas mediante `skos:exactMatch`, no se infieren como `pyedu:Concepto`, pues a diferencia de `owl:sameAs` el `skos:exactMatch` no propaga el tipo ni fusiona los individuos

[39] que tomaría el concepto propio y la entidad de Wikidata como un mismo individuo

(1,297, 1,380, 1,410) mientras el modelo memorizaba el conjunto de entrenamiento. La regularización mediante NEFTune y *dropout* en LoRA corrigió la tendencia y devolvió la pérdida *held-out* a valores decrecientes (1,051 y 1,028), con una precisión de token de 0,842. Que el sobreajuste apareciera y exigiera mitigación explícita es la advertencia más elocuente del trabajo sobre los límites de la especialización, porque un modelo afinado sobre datos sintéticos generados por plantillas corre el riesgo de aprender la *forma* de las plantillas más que la *competencia diagnóstica* subyacente. La cautela anti-fuga adoptada (evaluar exclusivamente sobre esqueletos *held-out* no vistos en entrenamiento) es el control que da credibilidad a las cifras de B, pero el riesgo de fondo persiste. El excelente acierto de categoría del sistema afinado podría reflejar, en parte, una afinidad con la distribución sintética de entrenamiento que no se trasladaría sin merma a las entregas heterogéneas y desordenadas de estudiantes reales. El grafo, por su naturaleza simbólica y curada, es en principio menos vulnerable a este riesgo de sobreajuste distribucional, lo que constituye otro argumento, ahora de robustez, a favor de la complementariedad.

Esta robustez relativa del grafo conviene, no obstante, situarla en su escala real, porque el EKG de este trabajo es un grafo curado, pequeño y verificado afirmación a afirmación, y la curación manual no es una estrategia que escale. Cuando un grafo de conocimiento crece hasta miles o millones de entidades, su construcción y su mantenimiento dejan de poder hacerse a mano y pasan a apoyarse en extracción automática, en enlazado e identificación de entidades y en compleción de relaciones, todos ellos procesos que operan bajo incertidumbre y que introducen afirmaciones de fiabilidad desigual. Hogan (2020) recuerda, además, que la web de datos funciona bajo una hipótesis de mundo abierto en la que lo no conocido no se da por falso, sino que queda simplemente como desconocido (p. 137), de modo que un grafo a gran escala es por naturaleza incompleto y revisable, y su refinamiento consiste menos en alcanzar una verdad cerrada que en una conceptualización que, como define ese mismo autor, sea «a formal representation of knowledge that forms a shared conceptualisation of a given domain» (Hogan, 2020, p. 188) y que las partes acuerden como útil para interoperar. La implicación para este trabajo es doble. Por un lado, la fiabilidad simbólica que aquí atribuyo al grafo descansa en su tamaño contenido y en su verificación manual, condiciones que no se conservarían sin más al escalarlo; por otro, el camino hacia un EKG mayor (que la propia memoria reclama como continuación) exigiría incorporar técnicas de refinamiento y de cuantificación de la incertidumbre de cada afirmación que el grafo actual, por su escala, no ha necesitado, y que serían imprescindibles para que la augmentación siguiera siendo una fuente de conocimiento fiable y no un nuevo origen de ruido.

Sobre estas tensiones gravita una decisión de implementación que conviene recordar aquí. El anteproyecto contemplaba all-MiniLM-L6-v2, FAISS y Neo4j, mientras que la implementación efectiva adoptó `nomic-embed-text` junto con rdflib y GraphDB (Ontotext, s.f.). El giro no fue arbitrario, sino el resultado de priorizar la coherencia con los estándares RDF/OWL del grafo y la operatividad del despliegue local. Mencionarlo aquí subraya que las elecciones técnicas también son parte del espacio de compromisos que la discusión debe asumir.

26.6 Síntesis del contraste

Si hubiera que condensar el balance del contraste, lo formularía así. La hipótesis principal H1 recibe **apoyo preliminar**, ya que la augmentación con conocimiento estructurado mejora la retroalimentación frente al modelo desnudo, y lo hace, sobre todo, en concepto, trazabilidad y calidad pedagógica, mientras el *fine tuning* aporta lo suyo en categoría e identificación técnica del error; ambos superan al sistema base y son **complementarios**, complementariedad que el híbrido D (ya evaluado con n=50) confirma y sintetiza, y resulta el mejor sistema del conjunto al ganar o empatar en las siete dimensiones (D > {B, C} > A). Entre las sub-hipótesis, H1a se confirma de forma matizada y desdoblada por dimensiones; H1d se confirma con la mayor solidez relativa aunque sin alcanzar el ideal; H1b queda parcialmente sin contrastar por falta de una métrica directa de alucinaciones; y H1c es la más provisional, pues su validación reclama necesariamente jueces humanos. Ninguno de los targets numéricos del anteproyecto (el 85 % de precisión, el 5 % de alucinaciones, el 4,0 de utilidad, el 100 % de trazabilidad) se da por cumplido; el mejor sistema, el híbrido D, alcanza 0,76 en categoría sin llegar al 85 %, y aunque algunas

cifras se aproximan a las metas en dimensiones concretas, sobre cincuenta casos sintéticos y con un juez automático no son confirmaciones sino indicios.

Esta lectura no debilita la propuesta; la sitúa en su justo lugar. Lo que el trabajo ha logrado es diseñar, implementar y someter a una primera validación rigurosa una arquitectura que integra un grafo de conocimiento educativo con modelos de lenguaje locales para generar retroalimentación fundamentada, explicable y trazable, y obtener señales coherentes de que la integración importa y de que conocimiento estructurado y especialización paramétrica se complementan. Importa, eso sí, no leer ese «la integración importa» como un «la augmentación gana en abstracto», pues tanto la inversión sobre código real como la ablación de recuperación de la reconstrucción ORPO (donde el RAG por pasajes no supera a la base y solo la recuperación ontológica quirúrgica lo hace) muestran que el beneficio del grafo está condicionado a una recuperación selectiva y semánticamente anclada, y no a la mera presencia de contexto recuperado. La evidencia es preliminar y no autoriza una confirmación fuerte de las hipótesis; lo que autoriza, y no es poco, es a sostener que el camino merece recorrerse hasta la validación con docentes y estudiantes reales que el propio diseño reclama como continuación necesaria.

26.7 El diseño A/B/C/D como ablación

El diseño de cuatro sistemas admite una lectura de ablación, que la Figura 64 resume y que quiero hacer explícita, porque es la que prueba la aportación *marginal* de cada componente y la que un tribunal exigente reclamará. El contraste A↔C aísla el efecto del grafo sobre el modelo base, y es estadísticamente significativo, pues el grafo mejora el acierto de concepto y la trazabilidad sobre el LLM desnudo. El contraste A↔B aísla el efecto del *fine tuning*, y también es significativo, ya que mejora la clasificación del tipo de error. Las dos ablaciones más exigentes, sin embargo, son las que miden lo que un componente añade *sobre* el otro ya presente. El contraste B↔D mide qué aporta el grafo a un modelo que ya ha sido afinado, y la diferencia es numéricamente positiva en todas las métricas, pero **no alcanza significación estadística** a n=50 (Wilcoxon con corrección de Holm, p≈0,18–0,88). El contraste C↔D mide qué aporta el *fine tuning* a un sistema que ya dispone del grafo, y aquí el híbrido mejora con claridad la clasificación del error y, según la métrica de grounding, casi duplica la fidelidad al subgrafo. La ablación deja tres conclusiones. Cada componente aporta de forma significativa sobre el modelo base; ambos se combinan sin interferir, y el híbrido reúne así las dos fortalezas; pero la muestra actual no basta para demostrar que el grafo aporte de forma significativa *por encima* del *fine tuning* (que es la afirmación más fuerte de la hipótesis H1), resultado prometedor que queda pendiente de una muestra mayor. Subrayo esta distinción porque es donde se juega el valor científico del trabajo.

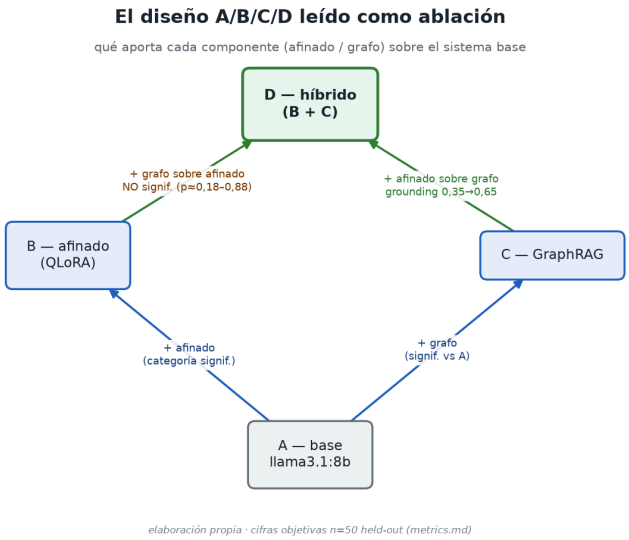

Figura 64. El diseño A/B/C/D leído como ablación.

26.8 Una ablación de la recuperación en la reconstrucción ORPO

El experimento canónico A/B/C/D recién discutido aísla *qué componente* añade valor, pero deja abierta una pregunta más fina sobre la recuperación misma. ¿Basta con augmentar, sea como sea, o importa *cómo* se recupera? Para responderla he ejecutado una **línea de reconstrucción independiente** cuyas cifras conviene mantener separadas con todo rigor de las del experimento canónico, pues constituyen un montaje experimental distinto y mezclarlas sería un error metodológico. Esa reconstrucción afina `Qwen2.5-Coder-7B` con alineación por preferencias ORPO (*reference-free*, β=0,1, QLoRA NF4, semilla 42) sobre un corpus pequeño de pares de preferencia, y contrasta (con el mismo juez `qwen2.5:32b` sobre diez casos reservados) cuatro variantes que comparten todo salvo la naturaleza de la recuperación: una base sin recuperar, ORPO-v2 que recupera de grafos de Web Semántica, ORPO-v3 que recupera de libros de Python y ORPO-v4 que recupera por SPARQL una única regla pedagógica del grafo ontológico (esta última **comparte adaptador** con v3, de modo que su diferencia es de recuperación pura, no de *fine tuning*).

El resultado es una ablación de la recuperación que matiza la tesis principal. Las dos variantes de RAG por **pasajes no superan a la base** (media global del juez de 3,467 para v2 y 3,567 para v3, frente a 3,733 de la base sobre 5), lo que indica que inyectar volumen de pasajes tangenciales **diluye** el contexto en lugar de enriquecerlo. Solo la recuperación **semántica y quirúrgica** de v4 supera a la base (4,233 frente a 3,733), con la ventaja concentrada en la dimensión técnica y en la sugerencia y un cierre de en torno al 78 % del trayecto entre la base y el techo humano (4,463) medido sobre ese eje de mejora, no sobre la media global plana. El dato debe acompañarse de sus límites, pues v4 sufre **dos derrotas reales** frente a la base (`sub_00803`, de 4,333 a 4,0, y `sub_01444`, de 4,333 a 3,667, ambas del patrón «modificar una lista mientras se itera»), solo tres de los diez casos llegaron a activar una *misconception* específica del grafo, la muestra es de apenas diez casos y el juez no es determinista. Esta ablación converge, desde otro experimento, con la inversión observada sobre código real Dublin, ya que ni el *fine tuning* ni la augmentación por volumen garantizan la mejora, y el valor diferencial del grafo aflora cuando la recuperación es **selectiva y semánticamente anclada**. Es, a mi juicio, la lectura más defendible y la que evita sobrevender una augmentación que solo bajo condiciones precisas resulta beneficiosa.

27 Implicaciones éticas, de género y ODS

Todo sistema que se interpone entre un estudiante y el juicio que se emite sobre su trabajo es, lo quiera o no su autor, un artefacto cargado de consecuencias morales. Un evaluador automático de programación no se limita a clasificar errores. Participa en la construcción de la imagen que el estudiante se forma de sí mismo como aprendiz, condiciona la confianza con la que afronta la siguiente tarea y, agregado sobre miles de interacciones, contribuye a decidir quién persevera en la disciplina y quién la abandona. Por eso este capítulo no es un apéndice protocolario que la normativa exige y el autor despacha con frases hechas, sino una reflexión que considero parte sustantiva del trabajo. Me propongo examinar, con el mismo rigor que ha guiado el reporte de resultados, las implicaciones éticas del sistema construido, su dimensión de género y su alineación con los Objetivos de Desarrollo Sostenible. Y, sobre todo, me propongo distinguir con nitidez entre lo que el diseño del sistema *permite* afirmar y lo que solo cabe *conjeturar* mientras no se valide empíricamente sobre personas reales, porque confundir ambas cosas sería, en un capítulo dedicado a la ética, una forma especialmente grave de incoherencia.

27.1 Privacidad, soberanía de los datos y cumplimiento normativo

La primera implicación ética del sistema está inscrita en una decisión arquitectónica que, vista desde fuera, podría parecer meramente técnica, la ejecución íntegramente local del modelo de lenguaje. El feedback se genera sirviendo el modelo a través de Ollama en la propia máquina, sin que las entregas del estudiantado abandonen el perímetro de la institución para viajar a las interfaces de programación de un proveedor comercial de modelos en la nube. Esta elección, que en el plano del rendimiento implica renunciar a los modelos más capaces y de mayor tamaño, responde a un compromiso deliberado con la **privacidad y la soberanía de los datos** de los estudiantes. El código que un alumno entrega no es un dato neutro. Revela su proceso de razonamiento, sus errores recurrentes, sus lagunas conceptuales y, en conjunto, un perfil cognitivo detallado que en manos equivocadas podría usarse para perfilar, comparar o discriminar. Mantener ese material dentro de la infraestructura controlada por la institución educativa, en lugar de cederlo a un tercero cuyos términos de servicio el estudiante no ha negociado ni probablemente leído, es la traducción operativa de un principio. Los datos de aprendizaje pertenecen a quien aprende y a la institución que lo acompaña, no al proveedor de un servicio de inferencia.

Esta arquitectura local guarda una sintonía profunda con el marco del Reglamento General de Protección de Datos (RGPD) que rige en el Espacio Económico Europeo, y al que cualquier despliegue real de este sistema en una universidad española quedaría sometido. El principio de **minimización de datos** se satisface de raíz cuando los datos no salen del sistema; el de **limitación de la finalidad** se respeta cuando las entregas se procesan exclusivamente para generar feedback formativo y no se reutilizan para entrenar modelos comerciales ajenos al control de la institución; y la dificultad notoria que plantean las transferencias internacionales de datos hacia proveedores radicados fuera del Espacio Económico Europeo simplemente no llega a materializarse cuando no hay transferencia alguna. No quiero, sin embargo, sobrevender este punto. Que la arquitectura *facilite* el cumplimiento del RGPD no equivale a que un despliegue concreto *cumpla* automáticamente con él. Un despliegue real exigiría todavía una base jurídica explícita para el tratamiento, información transparente al estudiantado, políticas de conservación y supresión de los registros, y previsiblemente una evaluación de impacto relativa a la protección de datos. Esta última se justifica porque se trata de un tratamiento sistemático de datos que perfilan a personas con fines evaluativos. El sistema construido en este trabajo opera sobre **datos sintéticos**, generados por plantillas y carentes de sujeto real, así que ninguna de estas obligaciones se ha tenido que afrontar en la práctica. La conformidad con el RGPD que aquí se describe es una conformidad *de diseño*, una propiedad que la arquitectura habilita, y no una conformidad *verificada* sobre un tratamiento de datos personales reales, que sería trabajo de una fase posterior y requeriría el concurso de la asesoría jurídica y el delegado de protección de datos de la institución.

Conviene precisar, además, el estatuto jurídico de los datos sobre los que este trabajo ha operado realmente. El **Considerando 26 del RGPD** establece que los principios de la protección de datos no

se aplican a la información anónima, esto es, a aquella que no guarda relación con una persona física identificada o identificable. Los datos sintéticos generados por plantillas que sustentan la evaluación de esta memoria carecen de sujeto real y no son reconducibles a ninguna persona concreta, de modo que se sitúan, por su propia naturaleza, fuera del ámbito material del Reglamento. Esto no debe leerse como una dispensa de cara al despliegue real, sino al contrario, porque en el momento en que el sistema procesara entregas de estudiantes concretos (datos personales identificables) el tratamiento entraría de lleno en el ámbito del RGPD y reactivaría todas las obligaciones descritas, incluida la evaluación de impacto. La anonimización de partida es, por tanto, una propiedad del banco de pruebas actual y no una garantía transferible al uso en producción.

27.2 Sesgos del modelo y de los datos

La segunda implicación ética, y quizá la más insidiosa por su carácter silencioso, es la del **sesgo**. Un modelo de lenguaje masivo es, en lo esencial, un destilado estadístico del corpus sobre el que se entrenó, y arrastra consigo las regularidades, los desequilibrios y los prejuicios presentes en ese corpus. Bender et al. (2021) advirtieron con su célebre metáfora de los «loros estocásticos» que estos sistemas reproducen patrones del lenguaje sin compromiso con la verdad ni con la equidad, y que los sesgos presentes en los datos de entrenamiento (de género, de cultura, de variedad lingüística, de estilo de programación) tienden a perpetuarse e incluso a amplificarse en la salida. En un evaluador de programación, esto se concreta en riesgos tangibles. El modelo podría valorar de forma sistemáticamente distinta soluciones idiomáticamente correctas pero estilísticamente minoritarias, penalizar enfoques de resolución menos representados en el corpus de entrenamiento, o expresarse en un registro que resulte más acogedor para unos perfiles de estudiante que para otros.

El sesgo, además, no procede solo del modelo preentrenado. En este trabajo se ha añadido una capa de *fine tuning* sobre un conjunto de datos generado sintéticamente por plantillas, y esa síntesis introduce su propio sesgo, acaso más sutil porque es de factura propia. Las plantillas codifican una concepción particular de qué errores son típicos, cómo se nombran y cómo se explican, y todo lo que quede fuera del repertorio de plantillas será, por construcción, peor atendido. La augmentación mediante el grafo de conocimiento educativo ofrece aquí un contrapeso parcial que merece atención. Al anclar el feedback en una estructura conceptual explícita, auditable y construida deliberadamente (los 157 conceptos propios, sus relaciones y los errores asociados), la recuperación sobre el grafo disciplina la salida del modelo y la somete a un marco curricular inspeccionable, en lugar de dejarla flotar sobre las asociaciones opacas aprendidas en el preentrenamiento. Un sesgo inscrito en el grafo es, al menos, un sesgo *visible*, porque figura en una ontología que puede leerse, criticarse y corregirse, a diferencia del sesgo difuso y prácticamente inescrutable de los pesos de un modelo de miles de millones de parámetros. No sostengo que el grafo elimine el sesgo (no lo hace, y además puede incorporar el de quien lo construyó, que en este caso soy yo, con mis propias limitaciones de perspectiva), pero sí que lo desplaza hacia un terreno donde la transparencia y la corrección son posibles. Debo reconocer, en todo caso, que **no se ha realizado una auditoría sistemática de sesgos** sobre las salidas del sistema, y que afirmar la equidad de su comportamiento sin esa medición sería una afirmación sin respaldo.

27.3 El rol docente: apoyo, no sustituto

La tercera consideración ética atañe a la naturaleza misma de lo que el sistema pretende ser, y es una consideración que prefiero formular como una línea roja antes que como un matiz. Este sistema se concibe como un **apoyo al docente, no como su sustituto**. La distinción es fundamental y de ella depende, a mi juicio, la legitimidad ética de todo el proyecto. La literatura sobre el feedback eficaz, desde la síntesis canónica de Hattie y Timperley (2007), enseña que el valor formativo del feedback no reside únicamente en señalar qué está mal, sino en orientar al estudiante sobre hacia dónde ir y cómo cerrar la brecha entre su estado actual y el objetivo de aprendizaje. Es una tarea que entrelaza juicio pedagógico, conocimiento del contexto del aula y sensibilidad hacia la persona concreta que aprende. Nada de esto puede delegarse íntegramente en un sistema automático sin un empobrecimiento del proceso educativo.

Lo que el sistema puede hacer (y lo que justifica su existencia) es absorber la parte mecánica, repetitiva y escalable del feedback: el diagnóstico de primer nivel sobre un gran volumen de entregas, la generación de explicaciones fundamentadas y trazables que el docente puede revisar y refrendar, y la liberación de tiempo docente para dedicarlo a la interacción de alto valor que ninguna máquina sustituye.

Encuadrar el sistema como herramienta de apoyo tiene además una consecuencia ética que protege al propio estudiante, pues mantiene a un ser humano en el bucle de las decisiones consecuentes. El feedback generado automáticamente no debería traducirse mecánicamente en una calificación con efectos académicos sin la mediación de un docente que lo valide. El sistema yerra, como muestran sin ambigüedad los resultados de esta memoria, y las decisiones que afectan a la trayectoria de una persona no deben quedar en manos de un proceso automático falible y opaco. Esta exigencia de supervisión humana no es solo una buena práctica pedagógica. Es, como se verá enseguida, un requisito normativo emergente. Asumir que el sistema *complementa* el juicio docente en lugar de reemplazarlo es lo que mantiene la responsabilidad última del proceso evaluador donde debe estar, en una persona que rinde cuentas, y no diluida en un artefacto que no las rinde.

27.4 Transparencia, explicabilidad y el marco del Reglamento Europeo de IA

La cuarta consideración ética es la **transparencia**, que en este trabajo no se ha quedado en declaración de intenciones sino que se ha materializado en un componente concreto, el módulo de explicabilidad. Frente a la opacidad característica de los modelos de lenguaje, que producen texto fluido sin exhibir el porqué de sus afirmaciones, el sistema implementa una interfaz web que **visualiza el subgrafo recuperado** del grafo de conocimiento mediante Cytoscape.js, con codificación por color, y muestra **marcadores de procedencia** que vinculan cada afirmación del feedback con su origen en el grafo. Esta capacidad no es cosmética. Convierte una caja negra en un proceso auditable. El docente puede inspeccionar en qué conceptos y relaciones se ha apoyado el sistema para emitir su diagnóstico, y el estudiante puede entender que el feedback no es un oráculo arbitrario sino una inferencia anclada en una estructura curricular explícita. La trazabilidad, que en el anteproyecto figuraba como una de las propiedades deseadas, encuentra aquí su instrumento, y la literatura sobre *learning analytics* responsable (Gašević et al., 2022) reclama esta clase de transparencia como condición de legitimidad de cualquier sistema que medie en el aprendizaje. Los resultados sobre trazabilidad son, no obstante, modestos en términos absolutos,[40] así que la transparencia está mejor servida por el diseño que plenamente lograda en la salida, y queda margen amplio de mejora. Esa dimensión, en la escala de 1 a 5, se reparte entre los sistemas evaluados del modo siguiente:

Sistema	Trazabilidad (1–5)
Línea de base	1,72
Afinado	3,00
Aumentado con el grafo	2,92
Híbrido (afinado + grafo)	3,16

Esta apuesta por la explicabilidad cobra un sentido adicional a la luz del **Reglamento Europeo de Inteligencia Artificial** (el *EU AI Act*), que establece un marco regulatorio basado en niveles de riesgo y que clasifica como sistemas de **alto riesgo** los destinados a determinar el acceso, la admisión o la evaluación del aprendizaje en instituciones educativas. Un evaluador automático de programación con efectos académicos caería previsiblemente en esa categoría, y los sistemas de alto riesgo quedan sujetos a obligaciones reforzadas, entre ellas la transparencia hacia los usuarios, la documentación técnica, la supervisión humana efectiva, la gestión de riesgos y la trazabilidad de las decisiones. Es revelador

[40] la dimensión correspondiente, puntuada de 1 a 5, alcanzó 3,16 en el sistema híbrido que sintetiza *fine tuning* y augmentación con el grafo frente a 1,72 en la línea de base, con valores intermedios de 3,00 y 2,92 en los sistemas afinado y aumentado con el grafo por separado

comprobar que varias de las decisiones de diseño adoptadas en este trabajo por convicción pedagógica (la explicabilidad mediante subgrafos y procedencia, el encuadre como apoyo bajo supervisión docente, la transparencia sobre los sesgos y las limitaciones) anticipan, sin habérselo propuesto explícitamente, buena parte de lo que ese marco normativo exigirá. No afirmo que el sistema sea conforme con el Reglamento Europeo de IA, una afirmación que requeriría un análisis jurídico específico y una evaluación que excede con mucho el alcance de este trabajo; afirmo, más modestamente, que su orientación de diseño es compatible con el espíritu de esa regulación emergente y que un despliegue real debería completar el camino hacia la conformidad formal antes de operar con consecuencias académicas sobre estudiantes reales.

Merece concretarse el anclaje normativo de esa clasificación de riesgo. El **Anexo III** del Reglamento Europeo de Inteligencia Artificial enumera los usos considerados de alto riesgo, y su punto 3, dedicado a la educación y la formación profesional, incluye en la **letra b)** los sistemas de inteligencia artificial destinados a evaluar los resultados del aprendizaje, también cuando esos resultados orientan el proceso de aprendizaje del estudiante. Un evaluador automático de programación con efectos académicos encaja con precisión en ese supuesto, lo que confirma que las obligaciones reforzadas de transparencia, documentación técnica, supervisión humana, gestión de riesgos y trazabilidad de las decisiones no constituyen una cautela genérica, sino el régimen concreto que un despliegue real de este sistema tendría que satisfacer. Que varias de esas exigencias coincidan con decisiones ya adoptadas en este trabajo por convicción pedagógica no exime de verificar formalmente la conformidad antes de cualquier uso con consecuencias sobre estudiantes. La Figura 65 traza ese anclaje normativo y pone en correspondencia las obligaciones reforzadas con las decisiones de diseño que ya las anticipan.

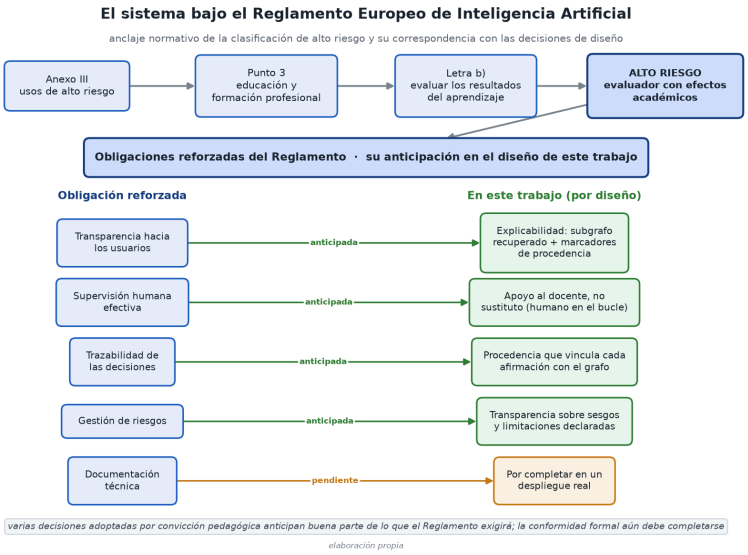

Figura 65. El sistema bajo el Reglamento Europeo de IA: del Anexo III (punto 3, letra b) a la clasificación de alto riesgo, y correspondencia de sus obligaciones reforzadas con las decisiones de diseño que las anticipan.

27.5 Dimensión de género

La dimensión de género de este trabajo merece un tratamiento cuidadoso y, sobre todo, explícito acerca de sus límites. El punto de partida es un hecho bien documentado, la persistente infrarrepresentación de las mujeres en la informática y la programación, un fenómeno multicausal en el que intervienen factores estructurales y culturales más que cualquier diferencia de capacidad. Cheryan et al. (2017) han analizado cómo el entorno, los estereotipos y la cultura de la disciplina contribuyen a esa brecha y configuran contextos que resultan menos acogedores para quienes no encajan en el perfil dominante. En este marco se inscribe la aportación que considero más relevante para el presente trabajo. Micheal (2022) documenta cómo la **ansiedad evaluativa** afecta de manera diferencial al estudiantado según el

género, y cómo las situaciones de evaluación pueden activar la amenaza del estereotipo y reforzar las brechas de confianza que alejan a las mujeres de la programación. Un sistema de feedback automático no es neutral frente a esta dinámica. El tono, la formulación y el enfoque del feedback (si se centra en el déficit y el juicio o en la orientación y el progreso) pueden mitigar o, por el contrario, agravar esa ansiedad evaluativa, con un impacto plausiblemente distinto según el género del estudiante.

De esta constatación se desprende una hipótesis de diseño. Un feedback formativo, fundamentado y orientado a la mejora (que explique el concepto implicado y señale el camino de corrección, en lugar de limitarse a sancionar el error) debería resultar menos amenazante y más equitativo que un feedback puramente correctivo o que la ausencia de feedback, y este beneficio podría ser especialmente valioso para el estudiantado más expuesto a la ansiedad evaluativa. La estrategia metodológica adecuada para contrastar esta hipótesis sería el **testing contrafactual**, es decir, comprobar si la salida del sistema varía indebidamente cuando se altera únicamente alguna señal asociada al género (por ejemplo, un nombre o un pronombre en la entrega) y se mantiene idéntico el código evaluado. Un sistema equitativo debería producir un feedback invariante ante cambios que no afectan a la calidad técnica de la solución; cualquier divergencia revelaría un sesgo que hay que corregir. Reconozco esta técnica como la vía correcta para auditar la equidad de género del sistema, y la Figura 66 esquematiza ese contraste.

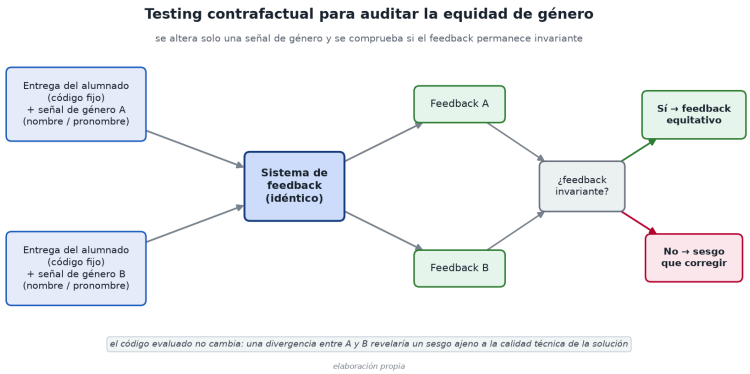

Figura 66. Testing contrafactual de equidad de género: la misma entrega, con el código fijo, se procesa alterando únicamente una señal de género; un feedback invariante indica equidad y una divergencia revelaría un sesgo a corregir.

Llegados aquí, prefiero una declaración explícita y rotunda en lugar de disolverla en eufemismos. **Este impacto de género es, en el estado actual del trabajo, una conjetura razonada y no un resultado validado**. La evaluación se ha realizado sobre datos sintéticos generados por plantillas, sin sujetos humanos reales, sin variable de género asociada a personas y sin medición alguna de ansiedad evaluativa, percepción de equidad o experiencia diferencial del feedback. No se ha ejecutado el testing contrafactual, no se ha recogido dato alguno sobre estudiantes reales de distinto género, y por tanto cualquier afirmación sobre los beneficios de género del sistema carece, hoy, de respaldo empírico. La hipótesis de que un feedback formativo y trazable reduce la ansiedad evaluativa y favorece la equidad de género es plausible, está fundamentada en la literatura y orienta el diseño, pero **no la he probado**, y presentarla como un logro del sistema sería la clase de inflación de resultados que la integridad científica de esta memoria rechaza. Lo dejo, pues, formulado como lo que es, una motivación de diseño y una línea de trabajo futuro que requiere estudios con participantes reales, consentimiento informado y los instrumentos de medida apropiados, sin los cuales el impacto de género del sistema permanece en el terreno de la conjetura bien fundada.

27.6 Alineación con los Objetivos de Desarrollo Sostenible

Alineación con los Objetivos de Desarrollo Sostenible

ODS	Contribución del proyecto	Estado
ODS 4 — Educación de calidad	feedback formativo accesible y trazable	real en intención y diseño; magnitud por demostrar
ODS 5 — Igualdad de género	reducción de sesgos en la retroalimentación	conjetural
ODS 9 — Industria, innovación e infraestructura	despliegue local, sin dependencia de la nube	potencial / por diseño
ODS 10 — Reducción de las desigualdades	acceso equitativo a tutoría automatizada	potencial / por diseño

elaboración propia

Figura 67. Alineación con los ODS 4/5/9/10.

El trabajo se inscribe, finalmente, en la Agenda 2030 y conecta de manera natural con cuatro de los Objetivos de Desarrollo Sostenible, contribución que también prefiero matizar antes que proclamar sin reservas. La Figura 67 resume esa alineación. El **ODS 4 (Educación de calidad)** es el más directamente concernido, pues el sistema persigue mejorar la calidad y la personalización del feedback formativo en la enseñanza de la programación. Para ello ofrece a cada estudiante una orientación adaptada a su nivel y fundamentada en una estructura conceptual explícita. Si el sistema funciona como se espera (y los resultados, recordemos, son aún modestos y obtenidos sobre datos sintéticos), contribuiría a un aprendizaje más eficaz y a un uso más inteligente del tiempo docente, que quedaría libre para la interacción de alto valor. La contribución al ODS 4 es, así, real en su intención y en su diseño, pero su magnitud efectiva sobre resultados de aprendizaje reales está todavía por demostrar.

El **ODS 5 (Igualdad de género)** conecta con la dimensión que se acaba de discutir. En la medida en que un feedback formativo y equitativo pueda reducir la ansiedad evaluativa diferencial y hacer la programación más acogedora para las mujeres, el sistema contribuiría a cerrar la brecha de género en la disciplina. Reitero aquí, para no incurrir en contradicción con lo dicho, que esta contribución es **conjetural** mientras no se valide empíricamente sobre estudiantes reales de distinto género; figura como aspiración alineada con el ODS 5, no como impacto medido. El **ODS 9 (Industria, innovación e infraestructura)** se vincula con la propia naturaleza del artefacto. La integración de grafos de conocimiento educativos con modelos de lenguaje mediante arquitecturas RAG constituye una innovación tecnológica aplicada, y la apuesta por una infraestructura de **despliegue local** con modelos de tamaño moderado, además de su fundamento ético en la privacidad, apunta hacia una innovación sostenible y soberana, menos dependiente de las grandes infraestructuras de inferencia en la nube y de su considerable huella de cómputo. El **ODS 10 (Reducción de las desigualdades)** se entronca con la posibilidad de **democratizar el acceso a un feedback de calidad**. Un sistema que escala el acompañamiento formativo podría acercar a contextos con menos recursos docentes una atención que hoy es privilegio de entornos bien dotados, y el énfasis en la ejecución local sobre hardware asequible refuerza esa orientación hacia la equidad en el acceso. También aquí, como en los demás objetivos, la contribución es de **potencial y de diseño**. El sistema está orientado a reducir desigualdades, pero su capacidad real para hacerlo dependerá de validaciones que aún no se han realizado y de condiciones de despliegue que exceden el alcance de esta memoria.

27.7 Recapitulación

El siguiente cuadro sintetiza ese hilo y contrapone, dimensión a dimensión, lo que el sistema incorpora *por diseño* frente a lo que todavía está *por validar*:

Dimensión ética	Salvaguarda incorporada por diseño	Estado de validación
Privacidad y RGPD	Ejecución local del modelo; las entregas no salen de la institución	Conformidad de diseño, no verificada
Sesgos	Anclaje en un grafo de conocimiento auditable que los hace visibles	Plausible, pero no auditada
Rol docente	Apoyo bajo supervisión humana, no sustituto	Línea roja de diseño
Transparencia y Reglamento Europeo de IA	Explicabilidad mediante subgrafo recuperado y marcadores de procedencia	De diseño, no conformidad verificada
Igualdad de género	Feedback formativo y trazable orientado a la mejora	Conjetura razonada, no probada
ODS 4/5/9/10	Calidad educativa, equidad, innovación e igualdad en el acceso	Potencial y de diseño, no impacto medido

Para cerrar, subrayo el hilo que recorre el capítulo de principio a fin, porque me parece el punto más importante. El sistema construido incorpora, *por diseño*, un conjunto de salvaguardas y orientaciones éticas que considero sólidas: la ejecución local respetuosa con la privacidad y compatible con el RGPD, anclaje en un grafo auditable que hace los sesgos visibles y corregibles, encuadre como apoyo bajo supervisión docente, transparencia mediante explicabilidad y procedencia en sintonía con el Reglamento Europeo de IA, y una orientación hacia la equidad de género y los Objetivos de Desarrollo Sostenible. Pero ninguna de estas virtudes éticas ha sido **validada empíricamente sobre personas reales**. La conformidad normativa es de diseño y no de verificación; la mitigación de sesgos es plausible pero no auditada; el beneficio de género es una conjetura razonada y no un resultado; las contribuciones a los Objetivos de Desarrollo Sostenible son de potencial y no de impacto medido. Distinguir lo que el sistema *habilita* de lo que *ha demostrado* no es una cautela retórica, sino la traducción al plano ético de la misma integridad científica que ha gobernado el reporte de resultados de esta memoria. Las garantías éticas más serias de este trabajo son, hoy, promesas de diseño que esperan su contrastación; reconocerlo así es la condición para que esas promesas puedan, en el futuro, llegar a cumplirse.

28 Limitaciones y trabajo futuro

Ningún trabajo de investigación está exento de límites, y el que aquí se documenta lo está especialmente, porque ha optado en cada bifurcación por la prudencia metodológica antes que por la espectacularidad de los resultados. Este capítulo rinde cuentas de esas restricciones sin atenuarlas ni esconderlas tras tecnicismos, y traza a continuación el programa de trabajo que las convertiría, una a una, en preguntas susceptibles de respuesta empírica más sólida. Leo ambas partes como las dos caras de un mismo análisis. Cada limitación que reconozco abre, casi por necesidad, una línea de continuación, y así el inventario de debilidades del presente es a la vez el mapa de la investigación que el trabajo deja planteada. He preferido organizar el capítulo separando con nitidez lo que *no se ha podido demostrar* (las limitaciones) de lo que *podría demostrarse* con recursos y tiempo adicionales (el trabajo futuro). Esa separación evita la tentación, frecuente en las memorias de investigación, de disfrazar de logro lo que no es más que una promesa.

28.1 Limitaciones del estudio

28.1.1 El tamaño de la muestra y la fragilidad estadística

La limitación más severa y la que condiciona la interpretación de todo lo demás es el **reducido tamaño de la muestra** sobre la que descansa el resultado principal. La comparación A/B/C/D que vertebra la validación experimental se realizó sobre **50 casos reservados** (*held-out*), un número que sigue siendo demasiado pequeño para atribuir significación estadística a las diferencias observadas entre sistemas o construir intervalos de confianza dignos de tal nombre. Con cincuenta observaciones, una métrica como el acierto de categoría (0,26 para el Sistema A, 0,70 para el B, 0,36 para el C y 0,76 para el D) se mueve aún en saltos discretos tan gruesos que el desplazamiento de unos pocos casos entre la columna de aciertos y la de fallos altera el porcentaje en varios puntos. Las diferencias que reporto son, por consiguiente, **indicativas de tendencias**, no demostraciones concluyentes. Sugieren con bastante claridad que el *fine tuning* mejora la clasificación del error, que la recuperación sobre el grafo mejora el acierto de concepto y la trazabilidad, y que el híbrido sintetiza ambas ventajas. Con todo, no autorizan afirmaciones categóricas sobre la magnitud de esas mejoras ni sobre su estabilidad ante nuevas muestras.

Quiero ser explícito sobre el origen de esta restricción, porque no obedece a descuido sino a una decisión deliberada que considero defendible. El conjunto evaluable resultó pequeño *porque* prioricé la salvaguarda anti-fuga descrita en el capítulo de metodología, esto es, restringir la evaluación a esqueletos reservados que ningún sistema vio durante el entrenamiento. Acepté la contrapartida de una muestra exigua antes que la alternativa, mucho peor, de inflar artificialmente el rendimiento del modelo afinado evaluándolo sobre casos estructuralmente idénticos a los de su entrenamiento. Dicho de otro modo, la pequeñez de la muestra es el precio de su limpieza. Pero un precio sigue siendo un coste, y no quiero ocultar que cincuenta casos constituyen todavía una base frágil sobre la que sostener conclusiones sobre la hipótesis principal del trabajo. La ampliación desde la evaluación preliminar (n=18) hasta los actuales n=50 ha aliviado parcialmente esta debilidad, aunque, como se argumenta más abajo, ni siquiera medio centenar de casos basta para zanjar la cuestión con holgura.

28.1.2 La naturaleza sintética de los datos

La segunda limitación de fondo es que **la totalidad del material de entrenamiento y evaluación se generó sintéticamente mediante plantillas**, sin que en ningún momento intervinieran entregas reales de estudiantes reales resolviendo ejercicios reales. Esta decisión tuvo motivos legítimos (ausencia de un corpus disponible con las garantías de privacidad necesarias, control sobre la distribución de errores que se quería cubrir, posibilidad de etiquetar cada caso con su concepto y su categoría de error de forma inequívoca), pero introduce un sesgo de validez externa que no puede ignorarse. Los errores que comete un estudiante de carne y hueso no se distribuyen como los que produce una plantilla. Presentan una variabilidad estilística, una ambigüedad y un grado de combinación de fallos simultáneos que ningún generador basado en patrones reproduce con fidelidad. Un programa real puede mezclar un

error conceptual de fondo con tres torpezas sintácticas y una mala elección de nombres, todo a la vez, mientras que un caso sintético tiende a aislar limpiamente el fenómeno que la plantilla quería ilustrar.

La consecuencia es que **el rendimiento medido sobre datos sintéticos no traslada automáticamente al mundo real**. Es perfectamente posible (y, a mi juicio, probable) que un sistema que clasifica bien la categoría de error en casos generados por plantilla degrade su acierto ante la complejidad y el ruido de las entregas auténticas. Toda la cadena de resultados de esta memoria debe leerse con la salvedad de que mide el comportamiento de los sistemas en un entorno controlado y empobrecido respecto al escenario de despliegue. La literatura sobre generación automática de feedback ha señalado repetidamente esta brecha entre la evaluación de laboratorio y el aula (Keuning et al., 2019; Marwan et al., 2020). Este trabajo no es una excepción y se sitúa, con plena conciencia, del lado controlado de esa brecha.

28.1.3 El sobreajuste y la sombra de la memorización

La tercera limitación atañe a la propia maquinaria del *fine tuning* y quedó documentada con transparencia en los capítulos técnicos. La primera ejecución del *fine-tuning* del Sistema B, sin regularización, exhibió un **patrón inequívoco de sobreajuste**, ya que la pérdida sobre el conjunto reservado empeoró época tras época (1,297, después 1,380 y finalmente 1,410), lo que delata un modelo que memoriza el material de entrenamiento en lugar de extraer de él regularidades generalizables. La mitigación mediante NEFTune (alpha 5) y *dropout* de 0,1 en los adaptadores LoRA corrigió la deriva y devolvió la pérdida *held-out* a valores sanos (1,051 y 1,028, con una precisión de token de 0,842 sobre el conjunto reservado), y la versión finalmente evaluada se entrenó bajo control. Reconocer este episodio no es una concesión retórica. Es advertir de que un modelo de siete mil millones de parámetros afinado sobre un dataset pequeño y de estructura regular **opera siempre al borde de la memorización**. La frontera entre aprender la tarea y aprender los ejemplos concretos es, en este régimen, tenue.

La regularización reduce el riesgo pero no lo elimina, y la combinación de datos sintéticos por plantillas con un modelo de gran capacidad es justamente el caldo de cultivo donde el sobreajuste prospera. La precisión de token de 0,842 sobre el *held-out* es alentadora, pero mide el ajuste sobre esqueletos que, aun siendo nuevos, comparten la *gramática* generativa de las plantillas de entrenamiento. No tengo garantía de que esa precisión se sostenga ante código cuya estructura no provenga de ninguna plantilla conocida. El buen comportamiento del Sistema B en la clasificación del error (acierto de categoría de 0,70) podría, en parte, reflejar una afinidad entre la distribución de evaluación y la de entrenamiento más que una competencia diagnóstica genuina y transferible.

Esta sospecha dejó de ser conjetura. Para someterla a prueba incorporé un conjunto de código **real** de estudiantes (el subconjunto *Dublin repair* del corpus público `koutch/intro_prog`, con la corrección de su profesorado como verdad de referencia) y reejecuté la comparación de los cuatro sistemas midiendo la relevancia de la retroalimentación respecto al arreglo real. El resultado, detallado en el capítulo de resultados, es elocuente y desfavorable al *fine tuning*. Sobre código auténtico el orden se invierte, y son el modelo base (0,80) y el aumentado con grafo (0,73) los que más se aproximan al arreglo real, mientras que el *fine tuning* (0,43) y el híbrido (0,32) quedan claramente por detrás. Aun reconociendo que esa métrica penaliza el estilo conceptual del feedback afinado (que por diseño no replica los identificadores del alumno), la señal confirma que buena parte de la ventaja de B y D sobre datos sintéticos es ajuste a la estructura de las plantillas y **no** una capacidad diagnóstica que transfiera al aula. Es la limitación más importante que este trabajo ha logrado medir, y no descalifica el enfoque, sino que delimita con precisión que la mejora del *fine tuning* es, por ahora, un fenómeno en distribución, y convierte en prioridad ineludible el entrenamiento con datos reales o la destilación de retroalimentación diversa. La Figura 68 traza esta cadena, desde los datos sintéticos por plantillas hasta la inversión del orden observada sobre código real.

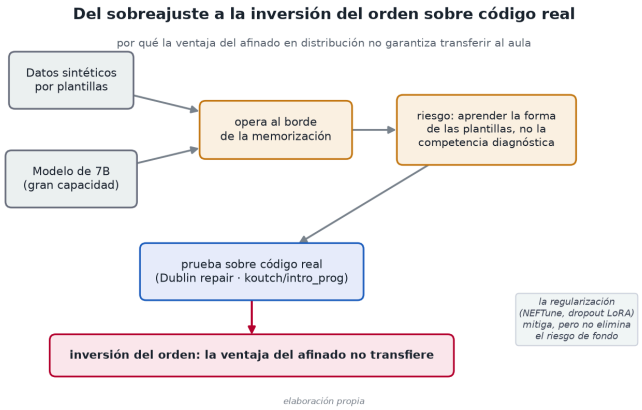

Figura 68. Del sobreajuste a la memorización y a la inversión del orden sobre código real. *Fuente: elaboración propia.*

La inversión del orden entre los dos escenarios se aprecia con claridad al confrontar, sistema a sistema, el acierto de categoría sobre los casos sintéticos con la relevancia frente al arreglo real medida sobre código auténtico:

Sistema	Acierto de categoría (50 casos sintéticos)	Relevancia frente al arreglo real (código real)
A (base)	0,26	0,80
B (afinado)	0,70	0,43
C (grafo)	0,36	0,73
D (híbrido)	0,76	0,32

El *fine tuning* (B) y el híbrido (D), que encabezan la clasificación del error en distribución, se desploman al fondo de la tabla cuando el referente es el arreglo de un estudiante real, mientras que el base (A) y el aumentado con grafo (C) ascienden a las primeras posiciones.

28.1.4 El solapamiento de familia entre el modelo afinado y el juez

Una cuarta limitación, más sutil pero metodológicamente delicada, concierne a la **arquitectura de la evaluación misma**. Las dimensiones cualitativas de la rúbrica (identificación del problema, trazabilidad, calidad pedagógica) se puntuaron mediante un **juez LLM**, en concreto el modelo qwen2.5:32b. Ahora bien, el Sistema B se afinó sobre Qwen2.5-Coder-7B-Instruct, es decir, sobre un modelo de la **misma familia** que el juez. Este solapamiento de familia introduce un riesgo de sesgo que no puede descartarse sin un control adicional, pues es concebible que un juez tienda a valorar más favorablemente las salidas de modelos que comparten con él convenciones estilísticas, preferencias de formulación o sesgos de superficie, simplemente porque «hablan el mismo idioma» interno. Si ese efecto existiera, parte de la ventaja del Sistema B en las dimensiones puntuadas por el juez podría ser un artefacto de la evaluación y no una virtud del feedback.

No dispongo de evidencia de que tal sesgo se haya materializado, pero tampoco de la prueba contraria, y conviene dejar constancia de la posibilidad. El empleo de un único juez automático, además, hereda todas las limitaciones conocidas del paradigma LLM-como-juez: la sensibilidad a la formulación del prompt de evaluación, la inconsistencia entre ejecuciones y la ausencia de la fiabilidad inter-evaluador que solo un panel de jueces independientes (humanos o, en su defecto, modelos heterogéneos) permitiría estimar. La evaluación automática realizada es, en suma, un instrumento útil y reproducible, pero un instrumento **único y potencialmente sesgado**, no un veredicto imparcial emitido por un tribunal diverso.

28.1.5 La validez de criterio humana, ya resuelta, y el solape que queda

En versiones anteriores de esta memoria esta era la limitación más visible, pues la validación con anotadores humanos figuraba como ausente y todas las conclusiones sobre calidad pedagógica colgaban de un

juez automático sin contraste humano. Esa laguna **ya está cerrada en lo esencial**. Diez revisores (siete programadores y tres docentes universitarios, R01 a R10) han puntuado a ciegas las retroalimentaciones de los cuatro sistemas sobre los cincuenta casos reservados en las tres dimensiones cualitativas, con 2000 valoraciones en total, y sobre esa anotación he calculado tanto el acuerdo inter-anotador (ICC(2,k) global de 0,831 sobre el promedio de los diez, pese a un Fleiss κ por celda de apenas 0,096) como la validez de criterio del juez frente al consenso humano. El resultado, detallado en el capítulo de resultados, es que el juez `qwen2.5:32b` **no queda validado** como instrumento de puntuación absoluta, Pearson 0,203 global, negativo (−0,203) en la dimensión divulgativa, lo que confirma con evidencia humana real la cautela que el trabajo ya sostenía. La pieza que el anteproyecto situaba como trabajo futuro (el panel humano y el coeficiente de correlación intraclase inter-juez) deja, por tanto, de ser una promesa para estar ejecutada, y su veredicto refuerza la decisión metodológica de anclar las conclusiones en las métricas objetivas y el grounding.

Lo que sí permanece como limitación es un **solape parcial entre los conjuntos de casos**. Los cincuenta casos que anotaron los humanos y los cincuenta que puntuó el juez automático solo coinciden en **33**, así que la validez de criterio humano↔juez se calcula sobre esas 33 × 4 × 3 = 396 celdas comunes y no sobre las cincuenta completas. Las diecisiete celdas que faltan quedan fuera del cálculo, sin extrapolación sobre ellas; cerrar el solape a 50/50 (anotando los casos restantes o reejecutando el juez sobre el conjunto íntegro humano) es el único paso que resta para una validez de criterio plenamente alineada, y queda registrado como tal. Es una limitación de cobertura, no de método, ya que la correlación medida es real y suficiente para la conclusión cualitativa (el juez no reproduce el criterio humano), pero su precisión mejoraría con el conjunto completo.

Subsiste, además, la otra mitad de esta sección, que la anotación humana no toca, a saber, la **ausencia de datos reales** en la cadena de entrenamiento y evaluación principal. La calidad pedagógica medida con humanos lo ha sido sobre las retroalimentaciones de casos sintéticos; sin entregas reales de estudiantes en el bucle de validación humana no hay forma de saber si el sistema sería igual de útil para un estudiante concreto frente a un ejercicio concreto en un curso concreto, una cuestión de validez ecológica que ni el mejor panel humano resuelve mientras los casos sigan saliendo de plantillas. El escalado a entregas reales y a un volumen mayor de submissions, descrito más abajo, sigue siendo trabajo pendiente.

28.1.6 Los targets del anteproyecto no se han alcanzado

De todo lo anterior se sigue que **ninguno de los objetivos de rendimiento numéricos fijados en el anteproyecto se da por cumplido**. No se ha alcanzado la precisión diagnóstica del 85 % o superior, ni la tasa de alucinaciones del 5 % o inferior, ni la utilidad formativa media de 4,0 sobre 5, ni la trazabilidad del 100 %. Aquellos umbrales se formularon, como se argumentó en el capítulo de metodología, como **hipótesis cuantitativas a contrastar**, no como listones que el trabajo debiera franquear para considerarse exitoso, y los datos disponibles los sitúan, en su estado actual, fuera del alcance. Los aciertos de categoría y de concepto se mueven en rangos muy alejados del 85 % (el mejor sistema, el híbrido D, alcanza apenas 0,76 en categoría y 0,54 en concepto), las puntuaciones de identificación y trazabilidad rondan o superan ligeramente la mitad de la escala incluso en los mejores sistemas, y no se ha medido directamente la tasa de alucinaciones con un protocolo dedicado. Reportar esto es, paradójicamente, uno de los resultados más valiosos del trabajo, porque he mantenido la frontera entre la aspiración del anteproyecto y la medición real, y me he negado a maquillar la segunda para que se pareciera a la primera. El valor de la investigación no reside en haber cumplido unas metas optimistas, sino en haber producido evidencia preliminar sobre la dirección y la magnitud de los efectos, y sobre las condiciones bajo las cuales esas metas podrían, eventualmente, aproximarse.

28.2 Trabajo futuro

De las limitaciones al trabajo futuro

Limitación actual	Trabajo futuro
Muestra pequeña (n=50, 7 esqueletos held-out)	escalar a 300+ entregas reales
Datos sintéticos por plantillas (templating)	despliegue real con medición desagregada
Sobreajuste / memorización (inversión en código real: A 0,80 / C 0,73 / B 0,43 / D 0,32)	alineación por preferencias (ORPO)
Solape de familia entre el modelo afinado y el juez	métricas RAGAS + κ de Cohen
Validez de criterio humana solo parcial	panel de 2–3 docentes sobre los 50 casos

elaboración propia

Figura 69. De las limitaciones al trabajo futuro.

El reverso de cada limitación es una línea de continuación, y las ordeno aquí de menor a mayor envergadura, desde las que cierran lagunas inmediatas hasta las que abrirían programas de investigación de varios años. La Figura 69 traza esa correspondencia entre cada límite reconocido y la continuación que abre, y la Figura 70 ordena esas líneas según su envergadura creciente, de las lagunas inmediatas a los programas de varios años.

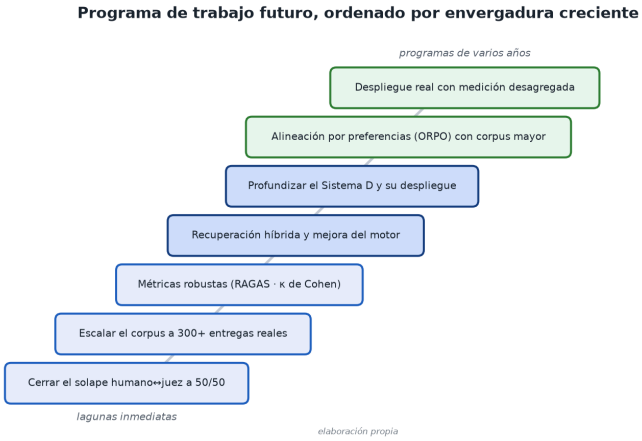

Figura 70. Programa de trabajo futuro ordenado por envergadura creciente. *Fuente: elaboración propia.*

28.2.1 Cierre del solape humano↔juez a 50/50

La validación con panel humano, que en el anteproyecto figuraba como la continuación más perentoria, **ya se ha ejecutado** con los diez anotadores R01 a R10, que han puntuado los cuatro sistemas, y sobre ellos he calculado el coeficiente de correlación intraclase (ICC(2,k) global de 0,831) y la validez de criterio del juez frente al consenso humano. Lo que esa ejecución dejó abierto, y que constituye ahora la tarea inmediata, es de alcance mucho más acotado y consiste en **cerrar el solape de casos a 50/50**. Como la validez de criterio se midió sobre los 33 casos comunes a la anotación humana y a la del juez (396 celdas de las 600 posibles), el paso pendiente es completar la anotación o la puntuación del juez sobre los diecisiete casos no solapados, para que la correlación humano↔juez se calcule sobre el conjunto íntegro. Es un asunto de cobertura más que de diseño, pues la infraestructura de anotación, el cómputo del ICC y el cruce de celdas están construidos y validados, y solo resta alimentarlos con los casos que faltan para que la validez de criterio quede medida sobre los cincuenta. Una vez cerrado ese solape convendría, además, ampliar el panel hacia el extremo superior de docentes previsto y repetir el cálculo, para robustecer un acuerdo inter-anotador que por celda fue bajo aunque fiable en promedio.

He dado, además, un paso intermedio que minimizó y dirigió óptimamente ese esfuerzo humano, y con él la validación dejó de ser una promesa difusa y pasó a ser un cálculo ejecutado. Siguiendo la literatura de validación de jueces automáticos, el *Alternative Annotator Test* (Calderon, Reichart & Dror, 2025), la co-anotación guiada por incertidumbre (Li et al., 2023) y el acuerdo entre modelos como predictor (Ahmed et al., 2025), he instrumentado una estrategia por capas y la he dejado armada en varios niveles. En primer lugar, un anotador automático de familia ajena al juez (Llama y Gemma, nunca un modelo Qwen, para no inflar el acuerdo por afinidad de familia) cuyo acuerdo entre modelos (kappa ponderado en torno a 0,39) y consenso frente al juez (kappa en torno a 0,20) reporto explícitamente como *cribado*,

no como validez de criterio, tal y como la propia literatura exige; en segundo lugar, la selección por incertidumbre de las quince celdas en las que los modelos más discrepan, que concentra el valor de cada anotación humana; en tercer lugar, el Alternative Annotator Test programado y verificado en su matemática, listo para emitir la decisión formal de sustitución sobre las celdas anotadas; y, por último, un andamiaje de *gold* conceptual anclado a literatura libre con licencia verificada para las dimensiones de concepto y explicación. Con la llegada de los diez anotadores R01 a R10, esa maquinaria ha dejado de esperar el dato humano y lo ha incorporado, de modo que la validez de criterio ya está medida (y su veredicto, que el juez no reproduce el criterio humano, es el que se reporta), y lo que el cribado modelo-modelo solo podía acotar como límite superior queda ahora resuelto con anotadores reales, a falta únicamente de cerrar el solape de casos descrito arriba.

28.2.2 Escalado del corpus a trescientas entregas o más

En paralelo, procede **escalar el corpus de evaluación hasta las trescientas submissions** comprometidas en el anteproyecto, e idealmente más allá. Solo con un volumen de ese orden adquieren sentido las pruebas de significación estadística, los intervalos de confianza y el análisis de la varianza entre sistemas que los cincuenta casos actuales (y mucho menos los cincuenta de la evaluación preliminar) no permiten. El salto de la muestra reducida actual a los centenares de casos no es un mero trámite cuantitativo, porque cambia la naturaleza de las conclusiones, que pasarían de *tendencias indicativas* a *diferencias acreditadas*. Este escalado debe acometerse, además, con cuidado de preservar la separación estricta entre entrenamiento y evaluación que ha protegido la limpieza de los resultados actuales. Solo así el mayor volumen no reintroducirá por la puerta de atrás la fuga de datos que tanto esfuerzo costó conjurar.

28.2.3 Métricas robustas: RAGAS y Kappa de Cohen

La evaluación futura ganaría rigor incorporando **instrumental métrico más robusto y específico**. El conjunto de métricas **RAGAS**, diseñado expresamente para sistemas de generación aumentada por recuperación, permitiría evaluar de forma desagregada propiedades como la fidelidad de la respuesta al contexto recuperado, la pertinencia del contexto y la ausencia de alucinaciones, dimensiones que un juez genérico mide solo de manera difusa. Medir la fidelidad con un protocolo dedicado sería, en particular, el camino para contrastar por fin la sub-hipótesis sobre alucinaciones, que en el presente trabajo no se ha cuantificado directamente. El **Kappa de Cohen**, cuando se disponga del panel humano, ofrecería una medida del acuerdo entre pares de evaluadores corregida por el azar, que complementaría al ICC y proporcionaría una segunda lectura de la fiabilidad. La combinación de RAGAS para las propiedades del sistema y de los coeficientes de acuerdo (ICC, Kappa) para la fiabilidad humana dotaría a la validación de una solidez de la que hoy carece.

28.2.4 Recuperación híbrida y mejora del motor

En el plano arquitectónico, la línea más prometedora es profundizar en la **recuperación híbrida**. El motor actual ya combina búsqueda vectorial sobre embeddings con expansión simbólica mediante SPARQL, pero el equilibrio entre ambas vías, los umbrales de similitud, la profundidad de expansión y los criterios de poda del subgrafo se fijaron de forma razonada pero no optimizada empíricamente. Un trabajo futuro sistematizaría la exploración de ese espacio de configuración, posiblemente incorporando técnicas de *reranking* del subgrafo recuperado y estrategias de fusión que pesen dinámicamente la evidencia vectorial y la simbólica según la naturaleza de la entrega. Dado que el análisis de los resultados confirma que la recuperación sobre el grafo (Sistema C) sobresale en acierto de concepto y trazabilidad, y que el híbrido (Sistema D) hereda y sintetiza esa ventaja, refinar el motor de recuperación es la palanca con mayor potencial para elevar las dimensiones donde el grafo aporta su valor diferencial (Edge et al., 2024; Gupta et al., 2024).

28.2.5 Profundización en el Sistema D y su despliegue

La pieza experimental que en su día quedó abierta (la **evaluación del Sistema D**, el híbrido que combina *fine tuning* QLoRA y GraphRAG) **se ha completado ya con la muestra ampliada de n=50**. El resultado A/B/C había mostrado una complementariedad nítida, pues el *fine tuning* (B) destaca en

clasificación del error y explicación técnica, y la recuperación (C) en acierto de concepto, trazabilidad y calidad pedagógica. Esa complementariedad era justamente la hipótesis que motivaba el Sistema D. Si las dos intervenciones actúan sobre dimensiones distintas, su combinación debería heredar lo mejor de ambas. La evaluación ampliada A/B/C/D confirma esa conjetura, pues el **Sistema D iguala o supera a los demás en las siete dimensiones evaluadas y es el mejor en seis de ellas**, con un perfil D > {B, C} > A que sintetiza las fortalezas antes repartidas entre el *fine tuning* y la recuperación. Con ello, la complementariedad B/C deja de ser una promesa para quedar **confirmada y sintetizada** por el híbrido. Debo matizar, no obstante, que esta superioridad se establece sobre cincuenta casos sintéticos y bajo el mismo juez LLM cuyas limitaciones se discutieron arriba. El Sistema D es el mejor de los cuatro en las condiciones medidas, pero ni siquiera él alcanza los targets del anteproyecto, y su ventaja no se ha contrastado todavía ni con un panel humano ni con datos reales. La continuación inmediata es, por tanto, someter al Sistema D a esas mismas validaciones más exigentes y explorar su configuración óptima de cara a un despliegue.

28.2.6 Alineación por preferencias mediante ORPO: de propuesta a método entrenado

Más allá del *fine tuning* supervisado, la vía de mejora del comportamiento generativo que este trabajo ya ha **explorado de hecho** es la **alineación por preferencias**. La técnica ORPO (*Odds Ratio Preference Optimization*) permite optimizar el modelo directamente a partir de pares de respuestas preferida y rechazada, sin necesidad de un modelo de recompensa separado ni de una fase de *fine tuning* supervisado previa e independiente, lo que la hace especialmente atractiva en un régimen de recursos contenidos como el de este trabajo. Aplicada al dominio del feedback formativo, ORPO enseña al modelo a *acertar* el diagnóstico y, además, a *preferir* las formulaciones pedagógicamente más eficaces frente a las menos útiles, es decir, el feedback que orienta hacia la solución sin desvelarla, que conecta el error con el concepto subyacente y que respeta el nivel del estudiante.

A diferencia de lo que figuraba en versiones previas de esta memoria, donde ORPO se presentaba como una mera continuación pendiente, esta alineación se **ha entrenado realmente** en una línea de reconstrucción independiente del experimento canónico A/B/C/D. Conviene aislar esta línea con nitidez y no mezclar sus cifras con las del experimento canónico que vertebra la memoria, pues se trata de un montaje experimental distinto, una **reconstrucción** (que aquí designo Base / ORPO-v2 / ORPO-v3 / ORPO-v4) construida sobre `Qwen2.5-Coder-7B-Instruct` con QLoRA NF4 y ORPO *reference-free*,[41] entrenada sobre 19 pares de preferencia con cuatro de validación, con una curva de `eval_loss` que desciende de 2,283 a 1,989 y `rewards/accuracies=0` (sin margen de preferencia discernible con tan pocos pares). La evaluación de esta reconstrucción se hizo con el mismo juez `qwen2.5:32b` sobre diez casos reservados, y es importante subrayar que la variante ontológica **ORPO-v4 comparte adaptador con ORPO-v3**, de modo que la diferencia entre ambas no procede del *fine tuning* sino de la recuperación.

Lo que esa reconstrucción enseña matiza la lectura del trabajo, y lo etiqueto siempre como experimento distinto del canónico. El **RAG por pasajes** (ORPO-v2, que recupera de los grafos de Web Semántica, y ORPO-v3, que recupera de libros de Python) **no supera al modelo base sin recuperación** (3,467 y 3,567 frente a 3,733 sobre 5 en la media global del juez), porque la dilución de contexto por volumen de pasajes tangenciales degrada la respuesta. En cambio, el **RAG semántico/ontológico ORPO-v4**, que recupera por SPARQL una única regla pedagógica quirúrgica del grafo, **sí supera a la base** (4,233 frente a 3,733). Concentra la mejora en la dimensión técnica y en la sugerencia, y cierra alrededor del 78 % del trayecto entre la base y el techo humano (4,463) medido sobre el eje de mejora técnica y sugerencia (no sobre la media global plana, donde el avance es menor). Esta mejora de v4 es, conviene insistir, de **recuperación, no de fine-tuning**, dado que comparte adaptador con v3. La lectura exige además consignar sus límites, pues hay **dos derrotas reales** de v4 frente a la base (los casos `sub_00803`, de 4,333 a 4,0, y `sub_01444`, de 4,333 a 3,667, ambos del patrón «modificar una lista mientras se itera»), solo tres de los diez casos activaron una *misconception* específica del grafo, la muestra es de apenas

[41] configuración del entrenamiento: `loss_type=«nll»`, β=0,1, sdpa, bf16, adaptadores r16/α32, *learning rate* 1e-5 con planificación coseno, `adamw_8bit`, *batch* 2, cinco épocas, semilla 42

diez casos y el juez no es determinista, y la propia ontología es un artefacto diseñado por IA anclado a la taxonomía del dataset. Esta reconstrucción no sustituye al experimento canónico ni lo contradice, sino que refina su tesis, ya que no es la augmentación por recuperación *en abstracto* la que mejora el feedback, sino una recuperación **selectiva y semánticamente quirúrgica**, mientras que la recuperación por volumen de pasajes puede incluso perjudicar. La continuación natural es construir un corpus de preferencias mayor (idealmente anotado por los propios docentes del panel) que dé a ORPO el margen de preferencia que con diecinueve pares no llegó a manifestarse, y cerrar así un círculo virtuoso entre la validación humana y la mejora del modelo.

La media global del juez sobre los diez casos reservados ordena las variantes de esta reconstrucción y deja ver que solo la recuperación semántica y quirúrgica de v4 mejora la base, mientras el RAG por pasajes la empeora:

Variante (línea de reconstrucción ORPO)	Media global del juez (sobre 5)
Base (sin recuperación)	3,733
ORPO-v2 (RAG de grafos de Web Semántica)	3,467
ORPO-v3 (RAG de libros de Python)	3,567
ORPO-v4 (RAG semántico/ontológico por SPARQL)	4,233
Techo humano	4,463

El contraste entre v4, que supera a la base, y v2/v3, que quedan por debajo de ella, es la evidencia que sostiene la tesis refinada de esta línea, a saber, que lo que mejora el feedback no es la augmentación por recuperación en abstracto, sino la recuperación selectiva y semánticamente quirúrgica.

28.2.7 Despliegue real con medición desagregada

La culminación natural de todo el programa es el **despliegue en un entorno educativo real**, con estudiantes auténticos resolviendo ejercicios auténticos, que es la única prueba última de la validez ecológica del sistema. Un despliegue así debería concebirse desde el principio con **medición desagregada**, esto es, registrando el comportamiento del sistema y la percepción de utilidad descompuestos por subgrupos relevantes del estudiantado. Esta exigencia conecta directamente con el marco ético y de género del anteproyecto, ya que medir el rendimiento de forma agregada puede ocultar que un sistema beneficia a unos perfiles de estudiante y perjudica a otros, y la literatura sobre equidad en *learning analytics* y sobre la participación de las mujeres en la informática advierte de que las herramientas automáticas pueden reproducir o amplificar sesgos preexistentes si no se auditan de forma desagregada (Cheryan et al., 2017; Micheal, 2022; Gašević et al., 2022). El despliegue local que la arquitectura ya garantiza (procesamiento en infraestructura propia, sin envío de datos del estudiantado a terceros) provee la base de privacidad y soberanía necesaria para una recogida responsable de esos datos. Un despliegue real, medido con desagregación y auditado éticamente, transformaría este trabajo de un artefacto validado en laboratorio en una herramienta cuya contribución a una evaluación formativa más justa y personalizada pudiera, por fin, contrastarse en el aula.

28.3 Cierre

El balance de este capítulo es deliberadamente sobrio. El trabajo ha construido un artefacto funcional y bien fundamentado (un grafo de conocimiento educativo validado, una arquitectura RAG operativa, un módulo de explicabilidad) y ha producido evidencia preliminar y reproducible sobre su comportamiento. Pero esa evidencia descansa sobre una muestra pequeña, datos sintéticos y una evaluación automática única, no ha sido refrendada por juicio humano experto y no alcanza los targets ambiciosos del anteproyecto, que siguen siendo hipótesis abiertas. Lejos de restar valor al trabajo, este reconocimiento lo sitúa donde corresponde, como un primer paso riguroso de un programa de investigación más amplio, cuyo trazado (del panel docente al despliegue real, pasando por el escalado del corpus, las métricas robustas, la profundización en el ya validado Sistema D y la alineación por preferencias mediante ORPO,

ya ensayada en su línea de reconstrucción y pendiente de un corpus de preferencias mayor) queda aquí explícito para que quien continúe, sea yo mismo u otros, encuentre el terreno desbrozado y las preguntas claramente formuladas.

29 Conclusiones

Llegado el final de esta memoria, me detengo a hacer balance con el rigor que exige cualquier trabajo de investigación y, muy especialmente, un Trabajo de Fin de Máster cuyo propósito último no es vender un producto terminado, sino demostrar una capacidad de razonamiento metodológico y de contraste empírico. Este capítulo recapitula lo conseguido, lo que ha quedado en el camino y lo que se abre como horizonte, y evita la tentación (tan frecuente en este tipo de cierres) de presentar como logros consolidados lo que todavía son indicios preliminares. La tesis central que ha guiado el trabajo era ambiciosa. Sostenía que la integración de un grafo de conocimiento educativo (EKG) con un modelo de lenguaje masivo, articulada mediante una arquitectura de generación aumentada por recuperación (RAG), permite producir feedback formativo en la enseñanza de la programación que sea, a un tiempo, fundamentado, explicable y trazable hasta sus fuentes. Lo que sigue es una valoración medida de hasta dónde he llegado en la contrastación de esa tesis.

29.1 Síntesis de lo conseguido

Grado de cumplimiento de los objetivos

Objetivo	Resultado	Valoración
O1 · EKG	157 conceptos · 20 clases / 21 prop. objeto / 7 de datos · 1772→4786 · SHACL conforme	cumplido en lo esencial
O2 · RAG	ast+radon + nomic-embed-text + SPARQL + Ollama	implementado
O3 · Explicabilidad	FastAPI + Cytoscape.js + procedencia	implementado
O4 · Validación	n=50; D mejor o empata en las 7 dimensiones	preliminar (targets no alcanzados)
O5 · Trade-offs	calas iniciales (tamaño, recuperación, cuantización, soberanía)	exploratorio

elaboración propia

Figura 71. Grado de cumplimiento de los objetivos.

La Figura 71 resume el grado de cumplimiento alcanzado en cada objetivo del anteproyecto. El primer resultado tangible, y el que sostiene todo lo demás, es la construcción de un grafo de conocimiento educativo funcional y formalmente verificado. Partiendo del objetivo O1 del anteproyecto, se ha materializado un EKG canónico con 157 conceptos propios sobre fundamentos de programación en Python, organizados en una ontología de 20 clases, 21 propiedades de objeto y 7 propiedades de datos, y serializado en Turtle bajo el perfil OWL 2 RL. Sobre los 1772 enunciados afirmados explícitamente, el motor de inferencia OWL-RL deriva un total de 4786 tripletas, lo que ilustra de forma concreta el valor añadido del razonamiento simbólico. El conocimiento no se limita a lo que se escribió a mano, ya que el sistema completa la red de relaciones que la semántica de las propiedades implica de manera necesaria. Este punto se aprecia con especial nitidez en una consulta concreta que he empleado como sonda diagnóstica. La recuperación del conjunto completo de conceptos pasa de devolver 0 resultados sin inferencia activada a devolver 157 con ella (los 157 conceptos propios del grafo). Debo precisar que las 30 entidades de Wikidata, enlazadas mediante `skos:exactMatch`, no se infieren como `pyedu:Concepto`. A diferencia de `owl:sameAs`, `skos:exactMatch` enlaza los recursos sin propagar el tipo ni fusionar los individuos, con lo que aquellas entidades quedan correctamente fuera del conjunto de conceptos propios. Ese salto de 0 a 157 no es un mero artificio técnico, sino la demostración operativa de que la capa de inferencia está haciendo su trabajo y de que el alineamiento con vocabularios externos amplía efectivamente el alcance del grafo.

Sometí la calidad estructural del EKG a validación mediante SHACL (W3C, 2017), con resultado conforme y cero violaciones sobre el grafo canónico. Para descartar que esta conformidad fuera un falso positivo derivado de restricciones demasiado laxas, construí deliberadamente un ejemplo inválido, sobre el cual la validación detecta correctamente 6 violaciones. Esta prueba negativa es metodológicamente más informativa que la conformidad sin más, porque acredita que las formas de validación tienen poder discriminante real. El grafo incorpora además construcciones de modelado no triviales (relaciones N-arias materializadas, como la clase `EvaluacionActividad` con 10 instancias, y reificación con marcadores de procedencia) que sientan las bases para la trazabilidad que la hipótesis demandaba. Junto a los 30 enlaces `skos:exactMatch` hacia Wikidata, se han establecido 19 relaciones `skos:broader` (W3C, 2009) que dotan al grafo de una jerarquía conceptual navegable. La elección de `skos:exactMatch`

frente a `owl:sameAs` para estos alineamientos es deliberada, ya que evita imponer la fusión lógica de individuos que el perfil OWL-RL aplicaría con `owl:sameAs` (lo que confundiría el concepto propio con la entidad externa), y refleja una madurez en el uso de datos enlazados al enlazar sin equiparar ontológicamente los recursos. En conjunto, el objetivo O1 se considera cumplido en sus aspectos esenciales, si bien con un número de conceptos (157) próximo pero no holgadamente superior al umbral de 150 que el anteproyecto fijaba, y con un trabajo de relaciones que, como se discute más abajo, admite todavía ampliación.

El segundo bloque de resultados corresponde a la arquitectura RAG (objetivo O2) y a su componente de explicabilidad (O4). Se ha implementado una tubería completa que comprende un analizador de código basado en árbol de sintaxis abstracta (módulo `ast`) y en métricas de complejidad (`radon`), un indexador de embeddings con `nomic-embed-text`, un motor de recuperación de subgrafos por similitud semántica con expansión posterior vía SPARQL (W3C, 2013), un generador de prompts y la integración con un modelo de lenguaje servido localmente mediante Ollama. Sobre esta base se ha levantado una interfaz web con FastAPI y Cytoscape.js que visualiza el subgrafo efectivamente recuperado para cada caso, con un código de color y marcadores de procedencia que permiten al usuario ver de dónde procede cada afirmación del feedback. Este artefacto es la concreción visible de la promesa de explicabilidad. El sistema dice algo sobre el código del estudiante y, además, muestra el fragmento de conocimiento estructurado en el que se apoya para decirlo.

29.2 La evidencia experimental y su lectura

El tercer bloque, y el más delicado de interpretar, es el de la validación experimental (objetivo O3). Aquí es donde la distancia entre la ambición del anteproyecto y la realidad de lo ejecutado debe declararse sin ambages. El anteproyecto planteaba como targets unas cifras exigentes (precisión diagnóstica igual o superior al 85%, tasa de alucinaciones del 5% o menor, utilidad formativa de al menos 4.0 sobre 5 y trazabilidad del 100%) y describía una validación con 300 submissions y un panel de 3 a 5 docentes con cálculo de acuerdo inter-juez (ICC). Lo afirmo con claridad. Ninguno de esos targets se da por cumplido y ese diseño de validación humana no se ha ejecutado todavía. Lo que he realizado es una evaluación automática, sobre datos sintéticos, que emplea métricas objetivas y un modelo de lenguaje como juez (`qwen2.5:32b`), en un protocolo cuidadosamente diseñado para evitar fugas. La comparación se hace exclusivamente sobre esqueletos *held-out*, así que los casos de prueba no han sido vistos durante el ajuste. Es una evaluación legítima y trabajosa, pero es preliminar, de muestra pequeña, y no sustituye al juicio de docentes humanos.

Dentro de esos límites, los resultados son informativos. Comparé cuatro sistemas sobre 50 casos held-out: el sistema A (modelo base `llama3.1:8b`), el sistema B (modelo afinado con QLoRA sobre `Qwen2.5-Coder-7B`), el sistema C (modelo base aumentado con GraphRAG) y el sistema D híbrido (modelo afinado más GraphRAG). En acierto de categoría del error, A obtiene 0.26, B alcanza 0.70, C se queda en 0.36 y D logra el mejor valor, 0.76; en acierto de concepto, A logra 0.18, B 0.50, C 0.48 y D destaca con 0.54; en identificación del problema sobre escala de 1 a 5, A puntúa 2.04, B 3.80, C 2.44 y D el mejor valor, 4.04; y en trazabilidad, también de 1 a 5, A obtiene 1.72, B 3.00, C 2.92 y D el mejor valor, 3.16. La lectura de estas cifras es matizada y, por eso mismo, valiosa. El sistema B, el modelo afinado, sobresale en la clasificación del tipo de error y en la calidad de la explicación técnica, lo que es coherente con un fine-tuning que ha interiorizado los patrones de error del dominio. El sistema C, el aumentado con GraphRAG, mejora en aquello que la hipótesis vinculaba al grafo: la identificación del concepto subyacente y la trazabilidad. Y el sistema D híbrido confirma la hipótesis de complementariedad, ya que gana o empata en las siete dimensiones evaluadas y combina la precisión diagnóstica del fine-tuning con la trazabilidad del grafo. Todos superan al sistema base A en prácticamente todas las dimensiones. Esta complementariedad no es un resultado decepcionante; al contrario, es el hallazgo que justifica de manera directa la propuesta del sistema D híbrido, cuya evaluación ampliada A/B/C/D con n=50 ya se ha completado y ha resultado el mejor sistema del conjunto **sobre los datos sintéticos reservados**. Esa salvedad no es retórica. Al someter los cuatro sistemas a código **real** de estudiantes (el subconjunto

Dublin repair, con la corrección docente como referencia), el orden se invierte y el *fine tuning* y el híbrido quedan por detrás del modelo base y del aumentado con grafo en relevancia respecto al arreglo real. La superioridad del híbrido es, por tanto, un fenómeno **en distribución**, fuertemente condicionado por el carácter sintético de los datos; su transferencia al aula exige atacar la causa raíz del sobreajuste con datos reales.

El conjunto de la comparación canónica sobre los 50 casos held-out se resume en la tabla siguiente:

Métrica (50 casos held-out)	A (base)	B (afinado)	C (GraphRAG)	D (híbrido)
Acierto de categoría del error	0.26	0.70	0.36	0.76
Acierto de concepto	0.18	0.50	0.48	0.54
Identificación del problema (1–5)	2.04	3.80	2.44	4.04
Trazabilidad (1–5)	1.72	3.00	2.92	3.16

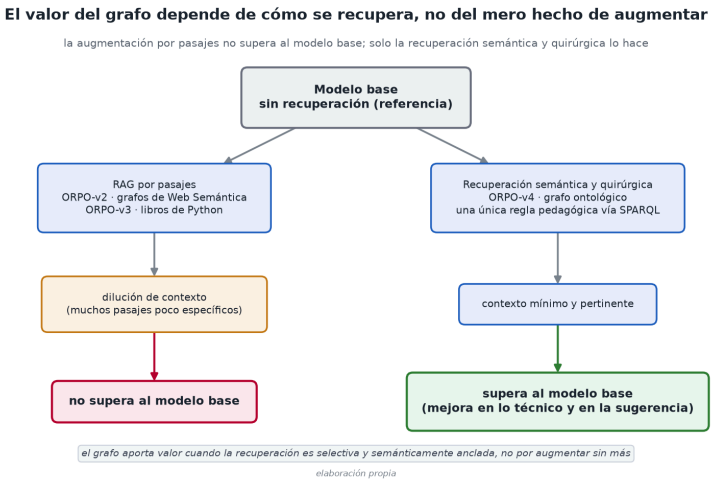

Figura 72. Cómo se recupera el conocimiento condiciona su valor: la augmentación por pasajes no supera al modelo base, mientras que la recuperación semántica y quirúrgica de una única regla pedagógica vía SPARQL sí lo hace.

Conviene además matizar la tesis general (que «la augmentación con grafo mejora el feedback») a la luz de una **línea de reconstrucción independiente** que he ejecutado en paralelo y cuyas cifras no deben confundirse con las del experimento canónico A/B/C/D que acabo de resumir, por tratarse de un montaje experimental distinto, una reconstrucción con alineación por preferencias ORPO sobre `Qwen2.5-Coder-7B`, evaluada por el mismo juez `qwen2.5:32b` sobre diez casos reservados, que contrasta cuatro variantes (Base / ORPO-v2 con grafos de Web Semántica / ORPO-v3 con libros de Python / ORPO-v4 ontológico). Su lección refina, sin contradecirla, la conclusión del trabajo. Como ilustra la Figura 72, la augmentación por recuperación **no mejora en abstracto**, pues el RAG por **pasajes** (ORPO-v2 y ORPO-v3) **no supera al modelo base** sin recuperación (3,467 y 3,567 frente a 3,733 sobre 5), porque la dilución de contexto degrada la respuesta. Solo la augmentación **semántica y quirúrgica** (ORPO-v4, que recupera por SPARQL una única regla pedagógica del grafo ontológico) **supera a la base** (4,233 frente a 3,733), con la mejora concentrada en la dimensión técnica y en la sugerencia. Esa mejora de v4 es de **recuperación, no de fine-tuning** (comparte adaptador con v3), convive con **dos derrotas reales** frente a la base (`sub_00803` y `sub_01444`), solo tres de los diez casos activaron una *misconception* específica, la muestra es de diez casos y el juez no es determinista. La conclusión reconciliada, por tanto, no es que «el híbrido o la augmentación ganen» sin más, sino que la integración del grafo aporta valor **cuando la recuperación es selectiva y semánticamente anclada**, y que su ventaja, tanto en el experimento canónico (inversión Dublin) como en la reconstrucción (derrota del RAG por pasajes), está sujeta a condiciones que conviene no sobrevender.

Respecto al fine-tuning del sistema B, recojo en la memoria un episodio metodológico que conservo como aprendizaje. El primer run, sin regularización, mostró un sobreajuste inequívoco. La pérdida sobre el conjunto held-out empeoraba época a época (1.297, 1.380, 1.410), señal clásica de memorización. La introducción de regularización (NEFTune con alpha 5 y `lora_dropout` 0.1) corrigió la tendencia y llevó la pérdida held-out a 1.051 y 1.028, con una precisión de token held-out de 0.842. El que este problema se detectara y se documentara, en lugar de ocultarse tras la métrica favorable, es en sí mismo un resultado de integridad. La muestra es pequeña y el dataset sintético generado por plantillas, de modo que estas cifras deben leerse como indicios de viabilidad técnica del procedimiento, no como una validación cerrada de su capacidad de generalización.

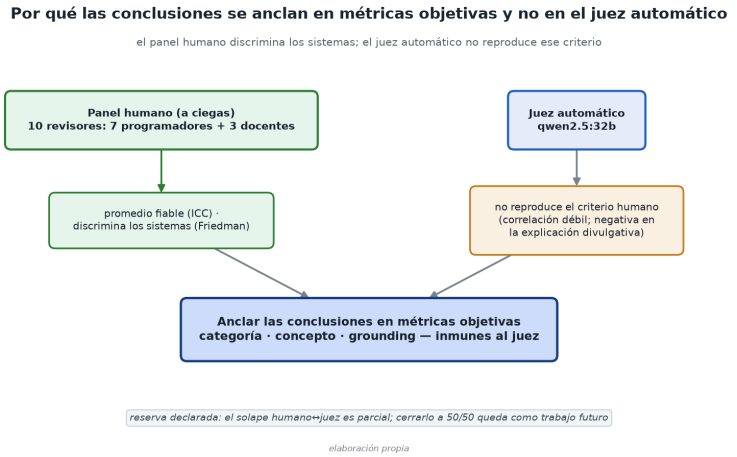

Figura 73. Por qué las conclusiones se anclan en las métricas objetivas y no en el juez automático: el panel humano discrimina los sistemas, pero el juez no reproduce ese criterio.

A esta lectura se suma ahora una pieza que en versiones previas de la memoria figuraba como pendiente y que ya está medida, la **validación de criterio con anotadores humanos**. Diez revisores (siete programadores y tres docentes, R01 a R10) puntuaron a ciegas los cuatro sistemas sobre los cincuenta casos reservados, con 2000 valoraciones, y de ello extraigo dos conclusiones que refuerzan el veredicto del trabajo sin alterarlo. La primera es que el promedio de los diez es un instrumento fiable (ICC(2,k) global de 0,831, y 0,910 en la dimensión técnica) y que el panel humano discrimina los sistemas con claridad, con Friedman significativo en las tres dimensiones cualitativas. La segunda, y más relevante para la solidez del trabajo, es que el juez automático `qwen2.5:32b` **no reproduce ese criterio humano**, pues la correlación de Pearson es de apenas 0,203 global y resulta **negativa** en la explicación divulgativa (−0,203), por lo que las puntuaciones del juez no pueden tomarse por verdad pedagógica. Lejos de socavar el trabajo, esta evidencia humana confirma con datos reales la cautela que ya gobernaba el veredicto, que las dimensiones cualitativas dependen de quién juzgue, y por eso las conclusiones se anclan en las métricas objetivas de categoría y concepto y en el grounding, inmunes al juez (véase la Figura 73). La única reserva que la propia anotación deja abierta es de cobertura, los conjuntos humano y del juez solo solapan en 33 de los 50 casos, así que la validez de criterio se calculó sobre esas 396 celdas comunes y cerrar el solape a 50/50 queda como trabajo futuro declarado, sin fingir su cierre.

29.3 Sobre las hipótesis y las decisiones de implementación

Si se proyectan estos resultados sobre las sub-hipótesis del anteproyecto, el cuadro es de evidencia parcial y direccionalmente favorable. La sub-hipótesis H1d, relativa a la trazabilidad, encuentra apoyo en la ventaja consistente de los sistemas con grafo (C 2.92 y D 3.16 frente a A 1.72) y en la infraestructura de procedencia y visualización de subgrafos, aunque el valor absoluto disten del 100% aspiracional. La sub-hipótesis ligada a la identificación del concepto correcto se ve respaldada por el 0.54 de D en acierto de concepto. En cambio, las sub-hipótesis sobre precisión diagnóstica global, reducción de alucinaciones y utilidad formativa percibida no pueden darse por contrastadas con la solidez que exigirían. La primera

muestra resultados prometedores aunque por debajo del target; la segunda requiere instrumentación específica que la evaluación humana proporcionará, y la tercera, aunque ya cuenta con el panel de diez anotadores que ordena los sistemas con claridad, no alcanza el umbral de 4,0 sobre 5 ni autoriza a tomar la puntuación del juez automático por medida fiable de utilidad formativa, dada su débil correspondencia con el criterio humano. En suma, la hipótesis general H1 (que la integración EKG+LLM vía RAG mejora la calidad pedagógica frente al LLM sin augmentación) recibe respaldo preliminar en las dimensiones de concepto y trazabilidad, sin que ello equivalga a haber alcanzado los umbrales cuantitativos ambiciosos.

El estado de contrastación de cada sub-hipótesis frente a la evidencia disponible puede resumirse así:

Sub-hipótesis	Estado	Evidencia en este trabajo
Trazabilidad (H1d)	Apoyo preliminar	Ventaja de los sistemas con grafo (C 2.92 y D 3.16 frente a A 1.72) e infraestructura de procedencia
Identificación del concepto	Apoyo preliminar	Acierto de concepto de D (0.54)
Precisión diagnóstica global	No contrastada	Resultados prometedores, pero por debajo del target
Reducción de alucinaciones	No contrastada	Requiere instrumentación específica vía evaluación humana
Utilidad formativa percibida	No contrastada	Panel de diez anotadores ordena los sistemas, pero no alcanza el umbral de 4,0/5

Dejo también constancia de una desviación deliberada respecto al diseño original, por transparencia técnica. El anteproyecto contemplaba `all-MiniLM-L6-v2` como modelo de embeddings, FAISS como índice vectorial y Neo4j como almacén de grafo. La implementación final emplea `nomic-embed-text` para los embeddings y la combinación de `rdflib` con GraphDB (Ontotext, s.f.) para el grafo. Estas decisiones no son caprichosas. `nomic-embed-text`, servido a través de la misma infraestructura local de Ollama, simplifica la pila tecnológica y refuerza la coherencia del despliegue local que el propio anteproyecto defendía por motivos de soberanía del dato; y la elección de un triplestore con soporte nativo de OWL 2 RL y SHACL, frente a un grafo de propiedades como Neo4j, era prácticamente obligada para sostener la inferencia y la validación formal que constituyen el núcleo diferencial de este trabajo. Documentar estos cambios, antes que silenciarlos, forma parte del rigor que la naturaleza de borrador declarado de esta memoria reclama.

29.4 Limitaciones

Las limitaciones de este trabajo son reales y no deben minimizarse. La más importante es la naturaleza de la evaluación, que es automática, sintética y de muestra reducida (50 casos held-out en la comparación principal). El uso de un modelo de lenguaje como juez introduce sus propios sesgos, y los datos generados por plantillas, aunque controlados, no capturan la variabilidad ni el ruido del código real de estudiantes reales. El grafo, con 157 conceptos, cubre los fundamentos pero no agota el currículo de una asignatura completa, y el número de relaciones, si bien funcional, se sitúa en el límite inferior de lo deseable. El fine-tuning, pese a la corrección del sobreajuste, se ha entrenado sobre un dataset modesto y su capacidad de generalización a familias de errores no representadas en las plantillas queda por demostrar. La validación pedagógica con anotadores humanos, que en estados anteriores del trabajo permanecía íntegramente pendiente, ya se ha ejecutado con el panel de diez revisores y su ICC inter-anotador, y su veredicto (que el juez automático no reproduce el criterio humano) está incorporado a las conclusiones. Lo que de aquella terna queda por hacer es de cobertura y escala: cerrar el solape de casos a 50/50 (la validez de criterio se midió sobre los 33 casos comunes) y crecer hacia el volumen de submissions previsto sobre datos reales. Reconocer estas limitaciones no debilita el trabajo; lo sitúa con precisión en el lugar que le corresponde, que es el de una prueba de concepto rigurosa y verificable, no el de un sistema validado para el aula.

29.5 Trabajo futuro y camino hacia la defensa

El horizonte que se abre es claro y está, en buena medida, ya iniciado. La evaluación ampliada A/B/C/D con n=50 ya se ha completado y ha permitido contrastar empíricamente la hipótesis de complementariedad entre el fine-tuning y la augmentación con grafo, pues el sistema híbrido D combina efectivamente la precisión diagnóstica de B con la trazabilidad de C y resulta el mejor sistema al ganar o empatar en las siete dimensiones. La prioridad inmediata pasa, por tanto, a consolidar este hallazgo con datos no sintéticos. La validación con panel humano y el cálculo del acuerdo inter-anotador, que el anteproyecto situaba como paso decisivo, ya se han ejecutado con los diez revisores R01 a R10; lo que de ese frente resta es acotado: cerrar a 50/50 el solape de casos sobre el que se midió la validez de criterio humano↔juez y extender la anotación a un volumen mayor de submissions sobre código real, lo que dotaría por fin a las sub-hipótesis de utilidad formativa y reducción de alucinaciones de la evidencia ecológica que aún les falta. En el plano del grafo, queda por ampliar la cobertura conceptual y, sobre todo, densificar las relaciones, así como extender los alineamientos con vocabularios externos. En el plano arquitectónico, queda por explotar plenamente el análisis de trade-offs que el objetivo O5 anticipaba (tamaño del modelo, profundidad de recuperación, efecto de la cuantización), del que el presente trabajo ofrece solo calas iniciales. Todo ello se enmarca en un calendario que apunta a la defensa en julio de 2027, lo que concede un margen razonable para convertir los indicios preliminares aquí reportados en evidencia consolidada.

En el plano de las implicaciones, este trabajo se ha mantenido fiel a los compromisos éticos del anteproyecto. El despliegue íntegramente local del sistema (con Ollama, GraphDB y embeddings propios) responde a la preocupación por la privacidad y la soberanía del dato del estudiante y mantiene la información sensible fuera de servicios de terceros. La explicabilidad por procedencia y la visualización de subgrafos persiguen la transparencia y, con ella, la preservación del rol del docente como supervisor crítico antes que como mero validador de una caja negra; la literatura sobre los riesgos de los modelos de lenguaje (Bender et al., 2021) y sobre la naturaleza del feedback eficaz (Hattie & Timperley, 2007) recuerda por qué esta cautela es necesaria. El trabajo se alinea, asimismo, con los Objetivos de Desarrollo Sostenible referidos a la educación de calidad y a la reducción de desigualdades, y con la atención a la dimensión de género en la enseñanza de la informática (Micheal, 2022; Cheryan et al., 2017).

Como reflexión final, creo que el valor de este Trabajo de Fin de Máster no reside en haber alcanzado los números más altos que un día imaginé, sino en haber construido un sistema que funciona de manera verificable y en haber medido lo que hace, incluido lo que no logra. Demostrar la viabilidad técnica de integrar un EKG con un LLM mediante RAG, con feedback trazable y explicable, y aportar evidencia preliminar favorable a la hipótesis en las dimensiones de concepto y trazabilidad, es un punto de partida sólido. El salto de 0 a 157 conceptos recuperados con inferencia, la conformidad SHACL que el ejemplo inválido confirma con sus 6 violaciones detectadas, y la complementariedad entre el fine-tuning y la augmentación con grafo, que el sistema híbrido D consolida sobre los 50 casos held-out, son hechos contrastables sobre los que construir, siempre que se lean con la condición que tanto la inversión sobre código real como la reconstrucción ORPO han hecho explícita, a saber, que el valor del grafo emerge cuando la recuperación es selectiva y semánticamente anclada, no por el mero hecho de augmentar. El camino hacia la validación pedagógica completa queda trazado, y la distancia entre lo conseguido y lo ambicionado no es, a mi juicio, motivo de decepción, sino la medida exacta del trabajo que me queda por delante de aquí a la defensa.

Todos los artefactos de este trabajo (el grafo y sus consultas, el sistema de recuperación y de *fine tuning*, y los resultados con su análisis estadístico) se reúnen, además, en una web única y autocontenida (`proyecto.html`) que se abre sin conexión y sirve como punto de acceso para consultar y evaluar el proyecto, incluida la parte de Web Semántica que lo fundamenta.

30 Referencias

Este capítulo recoge, en formato APA 7 con sangría francesa, la totalidad de las fuentes citadas a lo largo de la memoria. Se han organizado en dos bloques. El primero reúne las publicaciones académicas (artículos de revista, comunicaciones en congresos, preprints y trabajos de referencia) ordenadas alfabéticamente por el apellido del primer autor. El segundo agrupa las especificaciones técnicas y la documentación de productos que han servido de base normativa al diseño de la solución. Estas fuentes se citan por la entidad responsable —el W3C como organismo de estandarización y el proveedor del software, respectivamente—, tal como prescribe APA 7 para los documentos cuya autoría es corporativa. Las recomendaciones del W3C se referencian indicando su condición de estándar consolidado y la URL de la versión consultada, dado que carecen de DOI. Cuando una fuente dispone de identificador de objeto digital se incluye en su forma de enlace `https://doi.org/`; en su defecto se ofrece la URL estable más adecuada para su localización.

A efectos de la convención tipográfica que sigue todo el documento, las referencias se presentan con sangría francesa: la primera línea de cada entrada queda alineada al margen izquierdo y las líneas siguientes aparecen sangradas, de modo que el apellido del autor resulte fácilmente rastreable en una lectura vertical de la lista.

30.1 Publicaciones académicas

Abu-Salih, B., & Alotaibi, S. (2024). A systematic literature review of knowledge graph construction and application in education. *Heliyon*, *10*(3), e25383. https://doi.org/10.1016/j.heliyon.2024.e25383

Ahmed, T., Devanbu, P., Treude, C., & Pradel, M. (2025). Can LLMs replace manual annotation of software engineering artifacts? En *Proceedings of the 2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)* (pp. 526–538). IEEE. https://doi.org/10.1109/MSR66628.2025.00086

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? En *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21)* (pp. 610–623). Association for Computing Machinery. https://doi.org/10.1145/3442188.3445922

Calderon, N., Reichart, R., & Dror, R. (2025). The alternative annotator test for LLM-as-a-judge: How to statistically justify replacing human annotators with LLMs. En *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 16051–16075). Association for Computational Linguistics. https://aclanthology.org/2025.acl-long.782/

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., … Zaremba, W. (2021). *Evaluating large language models trained on code* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2107.03374

Cheryan, S., Ziegler, S. A., Montoya, A. K., & Jiang, L. (2017). Why are some STEM fields more gender balanced than others? *Psychological Bulletin*, *143*(1), 1–35. https://doi.org/10.1037/bul0000052

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. En *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)* (pp. 10088–10115). Curran Associates. https://doi.org/10.48550/arXiv.2305.14314

Du, X., & Daniel, B. K. (2024). Predicting students' academic performance with a hybrid machine learning approach. *Education and Information Technologies*, *29*(8), 9479–9505. https://doi.org/10.1007/s10639-023-12193-7

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., & Larson, J. (2024). *From local to global: A graph RAG approach to query-focused summarization* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2404.16130

Gašević, D., Greiff, S., & Shaffer, D. W. (2022). Towards strengthening links between learning analytics and assessment: Challenges and potentials of a promising new bond. *Computers in Human Behavior*, *134*, 107304. https://doi.org/10.1016/j.chb.2022.107304

Grattafiori, A., y otros. (2024). *The Llama 3 herd of models* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2407.21783

Gupta, S., Ranjan, R., & Singh, S. N. (2024). A comprehensive survey of retrieval-augmented generation (RAG): Evolution, current landscape and future directions [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2410.12837

Hattie, J., & Timperley, H. (2007). The power of feedback. *Review of Educational Research*, *77*(1), 81–112. https://doi.org/10.3102/003465430298487

Hogan, A. (2020). *The Web of Data*. Springer. https://doi.org/10.1007/978-3-030-51580-5

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutiérrez, C., Kirrane, S., Labra Gayo, J. E., Navigli, R., Neumaier, S., Ngonga Ngomo, A.-C., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., & Zimmermann, A. (2021). Knowledge graphs. *ACM Computing Surveys*, *54*(4), 1–37. https://doi.org/10.1145/3447772

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). *LoRA: Low-rank adaptation of large language models* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2106.09685

Kejriwal, M., Szekely, P., & Knoblock, C. A. (2021). *Knowledge Graphs: Fundamentals, Techniques, and Applications*. MIT Press.

Keuning, H., Jeuring, J., & Heeren, B. (2019). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education*, *19*(1), 1–43. https://doi.org/10.1145/3231711

Labra Gayo, J. E., Fernández-Álvarez, D., & García-González, H. (2018). RDFShape: An RDF playground based on shapes. En *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks (ISWC 2018)* (CEUR Workshop Proceedings, Vol. 2180). CEUR-WS.org. https://ceur-ws.org/Vol-2180/paper-35.pdf

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. En *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)* (pp. 9459–9474). Curran Associates. https://doi.org/10.48550/arXiv.2005.11401

Li, M., Shi, T., Ziems, C., Kan, M.-Y., Chen, N., Liu, Z., & Yang, D. (2023). CoAnnotating: Uncertainty-guided work allocation between human and large language models for data annotation. En *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1487–1505). Association for Computational Linguistics. https://aclanthology.org/2023.emnlp-main.92/

Marwan, S., Gao, G., Fisk, S., Price, T. W., & Barnes, T. (2020). Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science. En *Proceedings of the 2020 ACM Conference on International Computing Education Research (ICER "20)* (pp. 194–203). Association for Computing Machinery. https://doi.org/10.1145/3372782.3406264

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I., & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming

skills of first-year CS students. *ACM SIGCSE Bulletin*, *33*(4), 125–180. https://doi.org/10.1145/572139.572181

Micheal, S. (2022). *Gender mainstreaming in research and innovation: Practical guidance for integrating the gender dimension*. Publications Office of the European Union. https://doi.org/10.2777/056861

Peng, B., Zhu, Y., Liu, Y., Bo, X., Shi, H., Hong, C., Zhang, Y., & Tang, S. (2024). *Graph retrieval-augmented generation: A survey* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2408.08921

Qu, Y., Liu, P., & Zhu, W. (2024). A survey on knowledge graph-enhanced large language models for question answering [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2406.05690

Watson, C., & Li, F. W. B. (2014). Failure rates in introductory programming revisited. En *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (ITiCSE '14)* (pp. 39–44). Association for Computing Machinery. https://doi.org/10.1145/2591708.2591749

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., … Qiu, Z. (2024). *Qwen2.5 technical report* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2412.15115

Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., Wang, L., Luu, A. T., Bi, W., Shi, F., & Shi, S. (2023). *Siren's song in the AI ocean: A survey on hallucination in large language models* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2309.01219

Zhu, Y., Wang, X., Chen, J., Qiao, S., Ou, Y., Yao, Y., Deng, S., Chen, H., & Zhang, N. (2023). LLMs for knowledge graph construction and reasoning: Recent capabilities and future opportunities [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2305.13168

30.2 Especificaciones técnicas y documentación

Car, N. J. (s. f.). *pyLODE: An OWL ontology documentation tool using Python and templating, based on LODE* [Software]. RDFLib. Recuperado el 18 de junio de 2026, de https://github.com/RDFLib/pyLODE

Ontotext. (s. f.). *GraphDB documentation*. Ontotext AD. Recuperado el 18 de junio de 2026, de https://graphdb.ontotext.com/documentation/

RDFLib Team. (s. f.). *RDFLib: A Python library for working with RDF* [Software]. https://doi.org/10.5281/zenodo.6845245

Sommer, A., & Car, N. J. (s. f.). *pySHACL: A Python validator for SHACL* [Software]. RDFLib. Recuperado el 18 de junio de 2026, de https://github.com/RDFLib/pySHACL

World Wide Web Consortium. (2009). *SKOS Simple Knowledge Organization System reference* (W3C Recommendation, 18 de agosto de 2009). https://www.w3.org/TR/skos-reference/

World Wide Web Consortium. (2012). *OWL 2 Web Ontology Language document overview* (2.ª ed., W3C Recommendation, 11 de diciembre de 2012). https://www.w3.org/TR/owl2-overview/

World Wide Web Consortium. (2013). *SPARQL 1.1 query language* (W3C Recommendation, 21 de marzo de 2013). https://www.w3.org/TR/sparql11-query/

World Wide Web Consortium. (2014a). *RDF 1.1 concepts and abstract syntax* (W3C Recommendation, 25 de febrero de 2014). https://www.w3.org/TR/rdf11-concepts/

World Wide Web Consortium. (2014b). *RDF Schema 1.1* (W3C Recommendation, 25 de febrero de 2014). https://www.w3.org/TR/rdf-schema/

World Wide Web Consortium. (2017). *Shapes Constraint Language (SHACL)* (W3C Recommendation, 20 de julio de 2017). https://www.w3.org/TR/shacl/

31 Anexo A — Figuras de grafos

Este anexo reúne las representaciones de grafo del proyecto, generadas a partir del grafo canónico `ekg-python-150.ttl` y sus artefactos (cero invención; regenerables con el Anexo C).

31.1 A.1 Subgrafo RDF

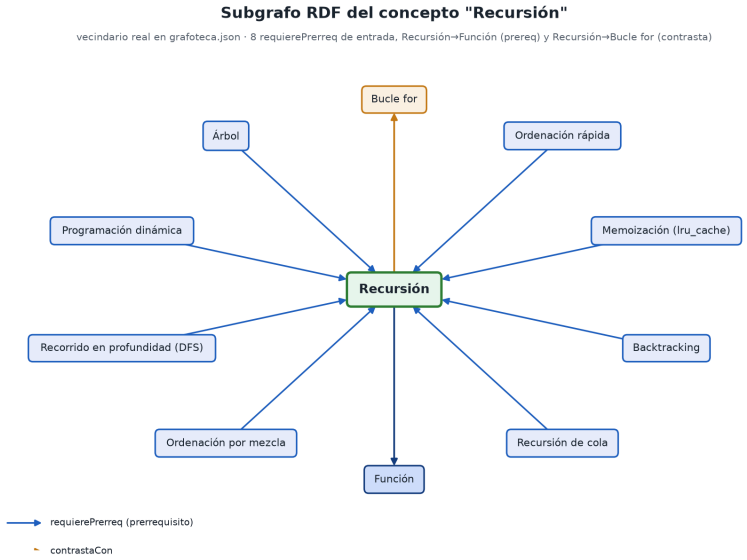

Figura 74. Subgrafo RDF del concepto «Recursión»: prerrequisitos, contrastes, error y tema.

31.2 A.2 Malla conceptual SKOS

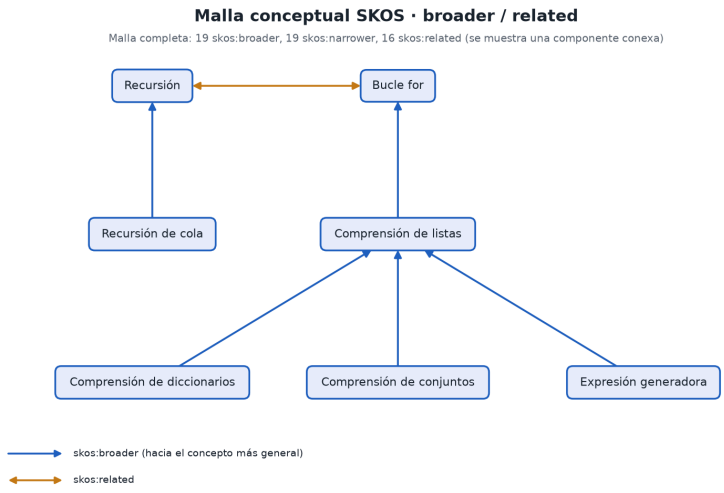

Figura 75. Malla SKOS: 19 broader, 19 narrower y 16 related.

31.3 A.3 Jerarquía de clases (TBox)

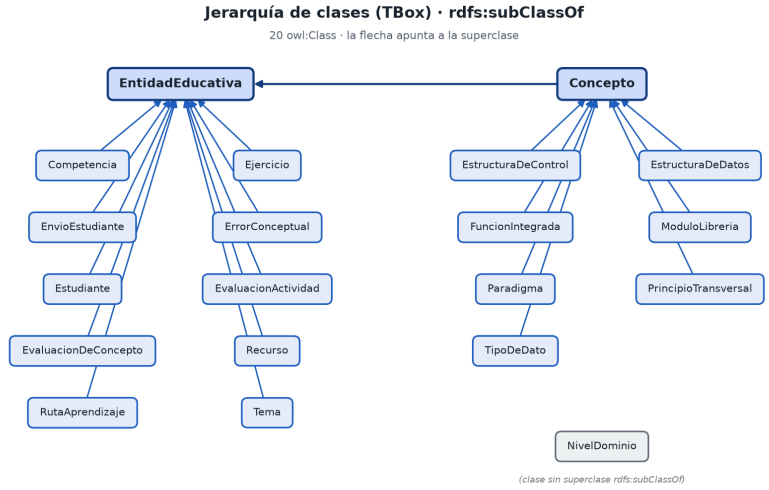

Figura 76. Jerarquía de clases del EKG mediante rdfs:subClassOf (20 clases).

31.4 A.4 Formas SHACL

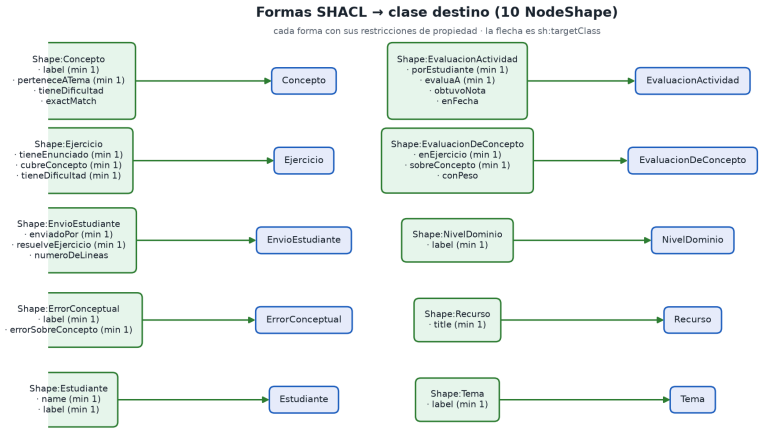

Figura 77. Las 10 formas SHACL (NodeShape) y sus clases destino.

31.5 A.5 Contraste de inferencia OWL 2 RL

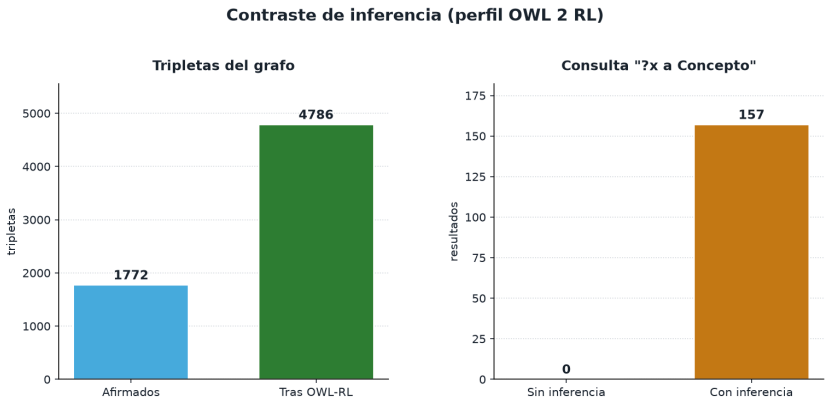

Figura 78. Contraste de inferencia: 1772 ⟶ 4786 triples; la consulta de conceptos pasa de 0 a 157.

31.6 A.6 Enlace a Wikidata

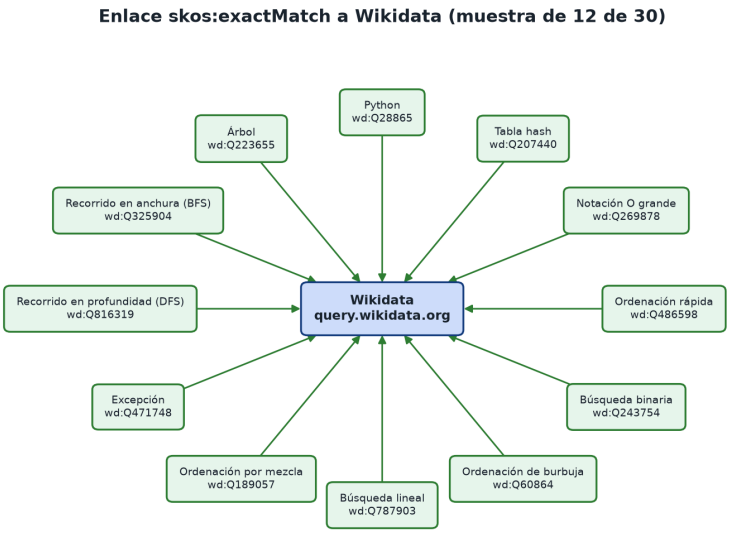

Figura 79. Enlace skos:exactMatch a Wikidata (muestra de 12 de los 30 enlaces).

31.7 A.7 Vocabularios y nube LOD

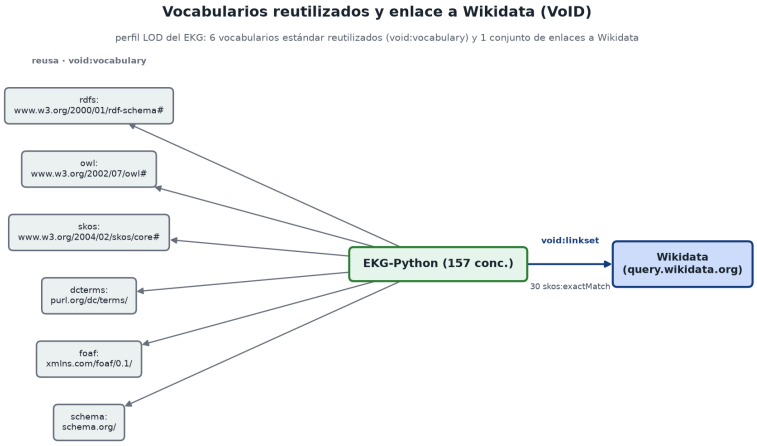

Figura 80. Vocabularios reutilizados y enlace a Wikidata, según void.ttl.

31.8 A.8 Curva de entrenamiento (QLoRA v3)

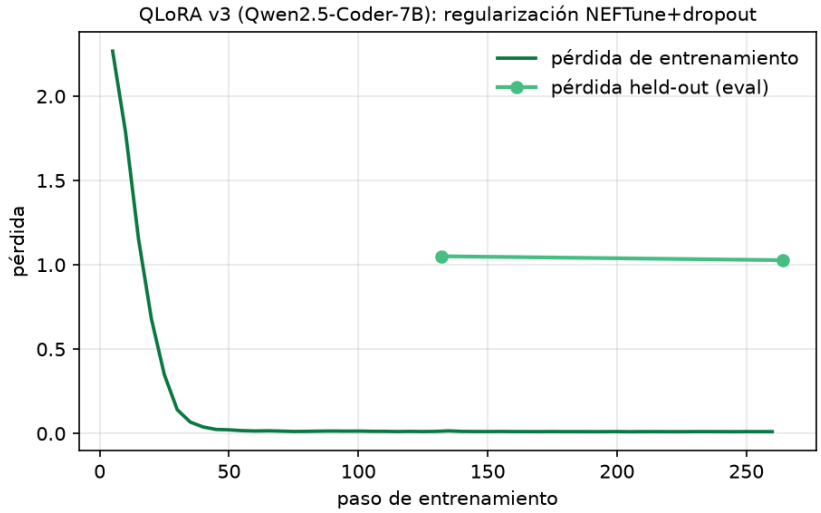

Figura 81. Pérdida de entrenamiento y held-out (QLoRA v3): la regularización invierte el sobreajuste (1,051 ⟶ 1,028).

31.9 A.9 Sobreajuste frente a regularización (v2 vs v3)

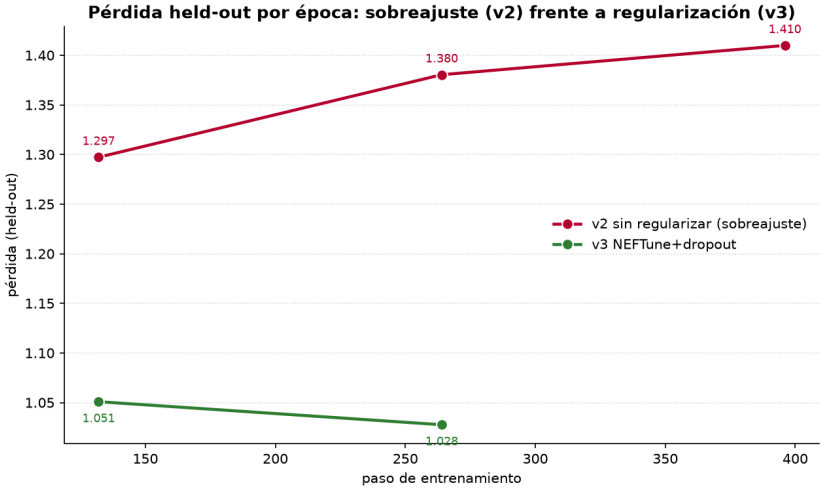

Figura 82. Pérdida held-out por época. El run v2 sin regularizar asciende de 1,297 a 1,410; el run v3 con NEFTune y dropout desciende de 1,051 a 1,028.

31.10 A.10 Proceso de afinado del Sistema B (QLoRA)

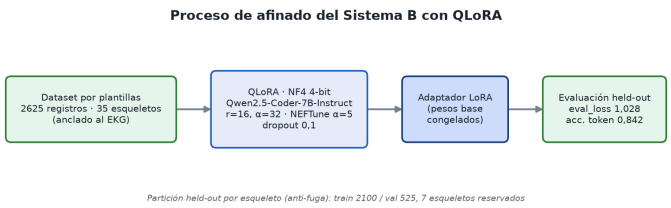

Figura 83. Tubería del afinado: dataset por plantillas (2625 registros, 35 esqueletos) ⟶ QLoRA NF4 4-bit sobre Qwen2.5-Coder-7B-Instruct (r=16, α=32, NEFTune α=5, dropout 0,1) ⟶ adaptador ⟶ evaluación held-out (eval_loss 1,028, precisión de token 0,842).

32 Anexo B — Aproximaciones de las herramientas

Las herramientas con interfaz gráfica de la asignatura (Protégé/OWLViz, GraphDB Workbench, Wikidata Query Service, RDFShape, Arrows.app y RDF Playground) producen vistas interactivas. Las figuras de este anexo son reproducciones programáticas generadas desde los artefactos del proyecto (el grafo `ekg-python-150.ttl`, las formas `shapes-ekg.ttl`, el resultado de la consulta federada y los historiales de entrenamiento), no capturas de la interfaz de cada herramienta. La captura en vivo de cada interfaz queda como adición opcional del autor.

Cada subsección incorpora ahora, además de la figura-aproximación, los elementos necesarios para **reproducir la herramienta y obtener la captura en vivo**: (a) el **bloque de código/comando/fichero exacto** del proyecto, (b) un **recorrido numerado** para abrir la herramienta, cargar el artefacto, ejecutar y situar la captura, y (c) la **ruta de guardado** de la captura en `99-Archivo/capturas/`, con el `embed` ya preparado y comentado para activarlo cuando el autor aporte la imagen. Todos los artefactos referenciados están bajo `grafo-ekg/` (rutas relativas a esa carpeta). Las salidas de las herramientas *headless* (pySHACL, RDFLib, pyLODE) que se muestran a continuación se ejecutaron realmente y se pegan literales.

32.1 B.1 Protégé / OWLViz

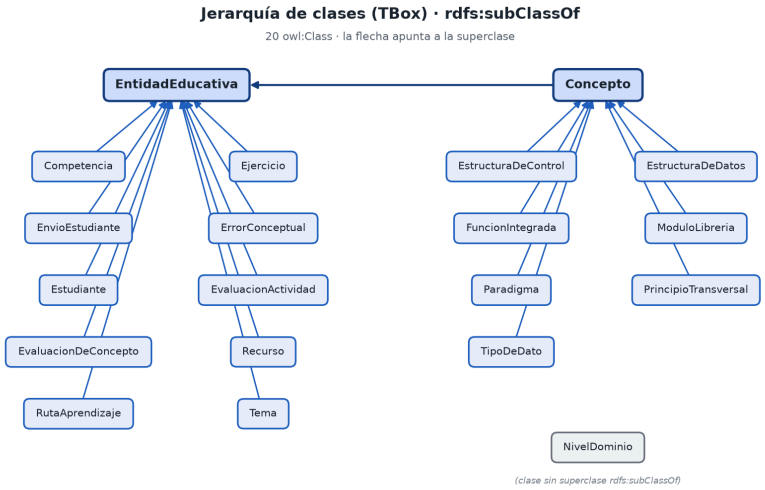

Figura 84. Jerarquía de clases del EKG por rdfs:subClassOf (20 owl:Class). Elaboración propia, equivalente a la vista de jerarquía de clases de OWLViz; no es una captura de Protégé.

La vista en vivo se obtiene abriendo `ontologia/ekg-python-150.ttl` en Protégé, pestaña OWLViz/OntoGraf. Esa captura queda pendiente de aportar por el autor.

Artefacto a cargar (fichero exacto del proyecto):

grafo-ekg/ontologia/ekg-python-150.ttl

Recorrido para la captura en vivo:

1. Abrir **Protégé Desktop** (5.6+). Menú *File ⟶ Open…* y seleccionar `ontologia/ekg-python-150.ttl`.
2. Si OWLViz no aparece, activarlo en *File ⟶ Check for plugins…* (o *Window ⟶ Tabs ⟶ OWLViz*); como alternativa nativa, usar *Window ⟶ Tabs ⟶ OntoGraf*.
3. Ir a la pestaña **OWLViz** (o **OntoGraf**) y seleccionar `owl:Thing` como raíz para desplegar la jerarquía de las 20 `owl:Class` por `rdfs:subClassOf`.
4. Expandir `pyedu:Concepto` para mostrar sus subclases (las que el razonador usa para tipar los 157 conceptos).

5. **Captura:** el grafo de jerarquía de clases con `pyedu:Concepto` expandido (equivalente a la figura `fig_tbox_clases`).

6. Guardar la imagen en: `B:\Web Semántica y Enlazado de Datos\99-Archivo\capturas\protege--owlviz.png`.

<!– ![Captura en vivo de Protégé/OWLViz — jerarquía de clases del EKG](../../99-Archivo/capturas/protege-owlviz.png) –>

32.2 B.2 GraphDB Workbench

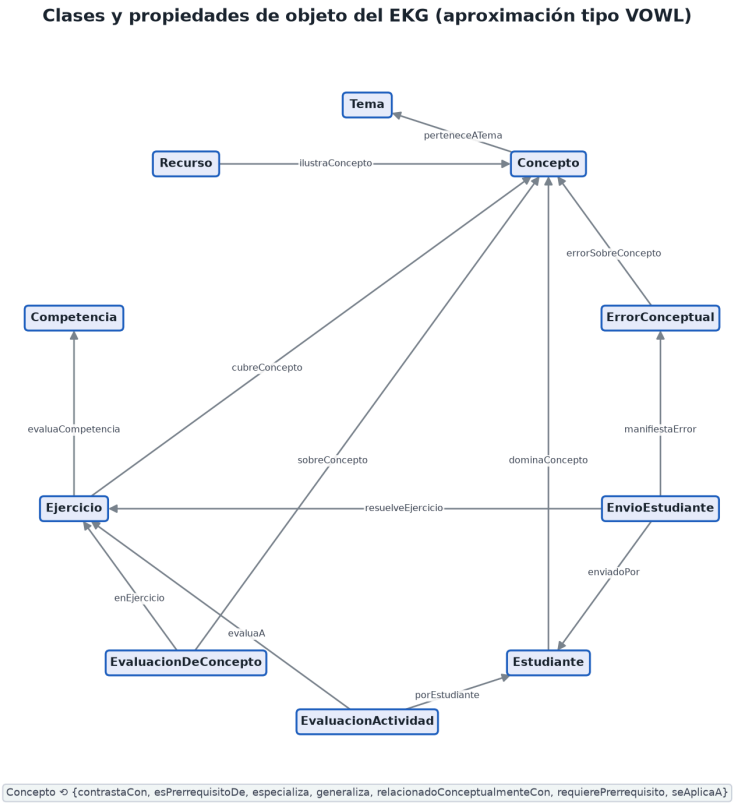

Figura 85. Clases y propiedades de objeto del EKG, aproximación tipo VOWL generada desde el bloque tbox de `grafoteca.json`. Elaboración propia; no es una captura de GraphDB.

El contraste de inferencia OWL 2 RL que calcula GraphDB (1772 → 4786 enunciados; la consulta de conceptos pasa de 0 a 157) está reproducido en la figura A.5 (`fig_inferencia`). La captura del Workbench con un repositorio de *ruleset* OWL2-RL queda pendiente de aportar por el autor.

Consulta a ejecutar en el Workbench (`consultas/02_inferencia_conceptos.rq`, la que evidencia la inferencia 0 → 157):

PREFIX pyedu: <https://w3id.org/ekg-python/schema#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

33 Recupera todo lo que es pyedu:Concepto.

34 - Inferencia NONE: 0 resultados.

35 - Inferencia RDFS/OWL-RL: 157 resultados (rdfs:subClassOf propaga el tipo).

SELECT ?concepto ?etiqueta

WHERE {

?concepto a pyedu:Concepto ;

rdfs:label ?etiqueta .

FILTER(LANG(?etiqueta) = «es»)

}

ORDER BY ?etiqueta

Recorrido para la captura en vivo:

1. Abrir **GraphDB Desktop** → *Setup → Repositories → Create new repository → GraphDB Repository*.
2. En *Ruleset* elegir **OWL2-RL (Optimized)** (o *RDFS-Plus*) y crear el repositorio (p. ej. `ekg--python`).
3. *Import → RDF → Upload RDF files* → `ontologia/ekg-python-150.ttl` → *Import*. GraphDB materializa la inferencia al importar (los enunciados pasan de 1772 a 4786).
4. Ir a *SPARQL*, pegar la consulta `02_inferencia_conceptos.rq` y pulsar **Run**: devuelve **157** filas directamente.
5. **Captura:** la tabla de resultados con el contador de 157 filas (y, opcionalmente, el panel *Repository* mostrando los 4786 *statements*).
6. Guardar la imagen en: `B:\Web Semántica y Enlazado de Datos\99-Archivo\capturas\graphdb--workbench.png`.

<!– ![Captura en vivo de GraphDB Workbench — consulta 02 con OWL2-RL (157 conceptos)](../../99-Archivo/capturas/graphdb-workbench.png) –>

35.1 B.3 Wikidata Query Service

Consulta federada 08 · skos:exactMatch → Wikidata (WDQS) · 20 filas

clave (EKG)	etiqueta (Wikidata, es)	descripción (es)
arbol_estr	Árbol	tipo abstracto de datos que imita la estructura jerárquica de un árbol
polimorfismo	Polimorfismo	informática
python	Python	lenguaje de programación de alto nivel
regex	Expresiones regulares (re)	secuencia de caracteres que forma un patrón de búsqueda en cómputo teórico y teoría de lenguajes formales
tabla_hash	Tabla hash	estructura de datos que asocia claves con valores, utilizando una función hash
iterador	Iterador (iter, next)	En computación, un objeto que permite recorrer un contenedor.
ordenacion_burbuja	Ordenación de burbuja	tipo de algoritmo de ordenamiento
ordenacion_rapida	Ordenación rápida	algoritmo de ordenación
closure	Clausura (closure)	función en informática que es evaluada en un entorno conteniendo una o más variables dependientes de otro entorno
complejidad	Complejidad temporal	tiempo de ejecución que describe la demora de una máquina en procesar una tarea
conjunto	Conjunto (set)	estructura de datos de tipo abstracto en informática
recursion	Recursión	método en ciencias de computación
bfs	Recorrido en anchura (BFS)	algoritmo de búsqueda no informada utilizado para recorrer o buscar elementos en un grafo (usado frecuentemente sobre árboles)
busqueda_binaria	Búsqueda binaria	algoritmo de búsqueda
generador	Generador (yield)	rutina especial en informática
grafo_estr	Grafo (estructura)	tipo abstracto de datos que consiste en un conjunto de nodos
clase	Clase	en programación orientada a objetos, plantilla para la creación de objetos de datos según un modelo predefinido
diccionario	Diccionario	estructura de datos que asocia claves con valores
herencia	Herencia	concepto en informática
lista	Lista	dato abstracto que representa una secuencia ordenada de valores en ciencias de la computación

Figura 86. Resultado de la consulta federada 08 (skos:exactMatch a Wikidata), 20 filas con etiqueta y descripción en español. Resultado reproducible exportado desde `salidas/08_federada_wikidata.txt`; no es una captura de WDQS.

La captura del editor de `https://query.wikidata.org` con `consultas/08_federada_wikidata.rq` queda pendiente de aportar por el autor.

Consulta federada exacta (`consultas/08_federada_wikidata.rq`):

PREFIX pyedu: <https://w3id.org/ekg-python/schema#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

PREFIX schema: <http://schema.org/>

SELECT ?concepto ?etiqueta ?descripcionWikidata

WHERE {

?concepto skos:exactMatch ?wd ;

rdfs:label ?etiqueta .

FILTER(STRSTARTS(STR(?wd), «http://www.wikidata.org/entity/»))

FILTER(LANG(?etiqueta) = «es»)

SERVICE <https://query.wikidata.org/sparql> {

?wd schema:description ?descripcionWikidata .

FILTER(LANG(?descripcionWikidata) = «es»)

}

}

Nota: el editor público `query.wikidata.org` se sitúa *dentro* del endpoint de Wikidata, así que el patrón local `?concepto skos:exactMatch ?wd` requiere que el grafo EKG esté disponible (cargado como datos o invertido). El sentido natural de la consulta es **EKG → Wikidata**: se ejecuta desde GraphDB/RDFPlayground (con el EKG cargado) y el `SERVICE` resuelve la descripción en Wikidata. En el editor de WDQS la captura ilustrativa se obtiene con la parte que sí corre allí: la recuperación de `schema:description` en español de las entidades enlazadas. La salida de referencia (20 filas) está en `salidas/08_federada_wikidata.txt`.

Recorrido para la captura en vivo:

1. Opción recomendada (federación real EKG⟶Wikidata): abrir **GraphDB** o **RDFPlayground** con `ekg-python-150.ttl` cargado, pegar el contenido íntegro de `08_federada_wikidata.rq` y ejecutar; el `SERVICE` consulta `https://query.wikidata.org/sparql`.
2. Opción WDQS directa (ilustrativa): abrir `https://query.wikidata.org`, pegar la cláusula del `SERVICE` con un `VALUES ?wd { wd:Q28865 wd:Q188267 … }` de las entidades enlazadas y ejecutar para mostrar las descripciones.
3. Pulsar el botón ▸ **(Run)** y esperar a las ~20 filas con etiqueta y descripción en español.
4. **Captura:** la tabla de resultados (columnas *etiqueta* / *descripcionWikidata*) con la consulta visible en el editor.
5. Guardar la imagen en: `B:\Web Semántica y Enlazado de Datos\99-Archivo\capturas\wdqs--federada.png`.

<!– ![Captura en vivo de Wikidata Query Service — consulta federada 08](../../99-Archivo/capturas/wdqs-federada.png) –>

35.2 B.4 RDFShape (SHACL / ShEx)

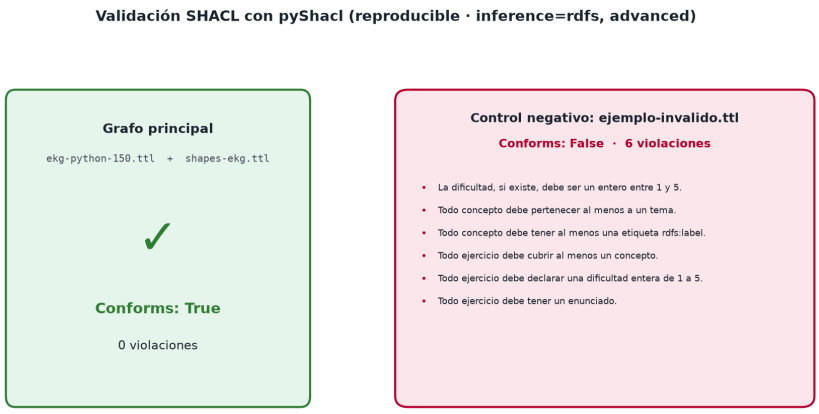

Figura 87. Validación SHACL con pyShacl (inference=rdfs, advanced). El grafo principal `ekg--python-150.ttl` conforma sin violaciones; el control negativo `ejemplo-invalido.ttl` produce 6 violaciones. Elaboración propia, reproducible; no es una captura de RDFShape.

La captura del informe de `https://rdfshape.weso.es` con el grafo y `shapes-ekg.ttl` (o el esquema `ontologia/ekg-shapes.shex`) queda pendiente de aportar por el autor.

Comando exacto (pySHACL, *headless*) — grafo canónico:

pyshacl -s shapes-ekg.ttl -df turtle ekg-python-150.ttl

Salida real (ejecutada desde `grafo-ekg/ontologia/`):

Validation Report

Conforms: True

Control negativo (el `ejemplo-invalido.ttl` cargado junto a la TBox `ekg-python.ttl` para que RDFS tipe `pyr:mal_concepto`; reproduce las 6 violaciones canónicas):

36 equivalente a scripts/validar.py (inference=«rdfs», advanced=True).

37 -e aporta la TBox como ontología para que RDFS tipe pyr:mal_concepto.

pyshacl -s shapes-ekg.ttl -e ekg-python.ttl -i rdfs -a -df turtle ejemplo-invalido.ttl

Salida real (mensajes literales de las 6 violaciones):

Conforms: False

Results (6):

- La dificultad, si existe, debe ser un entero entre 1 y 5.
- Todo concepto debe pertenecer al menos a un tema.
- Todo concepto debe tener al menos una etiqueta rdfs:label.
- Todo ejercicio debe cubrir al menos un concepto.
- Todo ejercicio debe declarar una dificultad entera de 1 a 5.
- Todo ejercicio debe tener un enunciado.

El informe completo persistido está en `salidas/informe-shacl-invalido.txt`. Como alternativa de formas declarada en la asignatura, el esquema **ShEx** equivalente es `ontologia/ekg-shapes.shex`.

Recorrido para la captura en vivo (RDFShape):

1. Abrir `https://rdfshape.weso.es` → menú **Validation → SHACL** (o *Schema → ShEx* para el `.shex`).
2. En el panel **RDF data**, pegar el contenido de `ontologia/ekg-python-150.ttl` (o cargarlo por *URL/file*).
3. En el panel **Shapes graph**, pegar el contenido de `ontologia/shapes-ekg.ttl`; activar *Inference: RDFS*.
4. Pulsar **Validate**: el informe muestra **Conforms: true** (sin violaciones) para el grafo canónico. Para reproducir el control negativo, repetir con `ontologia/ejemplo-invalido.ttl` como *RDF data* (debe mostrar las violaciones de forma).
5. **Captura:** el panel *Validation report* con el resultado *conforms* (verde) del grafo canónico.
6. Guardar la imagen en: `B:\Web Semántica y Enlazado de Datos\99-Archivo\capturas\rdfshape--shacl.png`.

<!– ![Captura en vivo de RDFShape — informe SHACL conforme](../../99-Archivo/capturas/rdfshape--shacl.png) –>

37.1 B.5 Arrows.app (property graph)

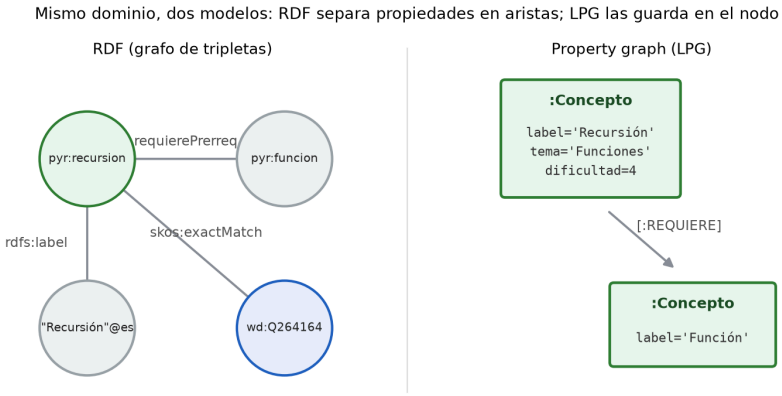

Figura 88. Mismo dominio en RDF y en property graph (LPG), donde el LPG guarda las propiedades dentro del nodo. Elaboración propia, aproximación del modelo de Arrows.app; no es una captura de la herramienta.

La captura del diagrama de `https://arrows.app` reproducido desde `salidas/grafos/ekg_lpg.cypher` queda pendiente de aportar por el autor.

Fichero Cypher exacto (`salidas/grafos/ekg_lpg.cypher`, 345 sentencias: nodos `:Concepto` con propiedades + aristas `:REQUIERE`/contraste/error). Primeras líneas:

CREATE (:Concepto {id:“abrir_fichero”, label:“Apertura de ficheros (open)”, tema:“Ficheros y E/S”, dificultad:3});

CREATE (:Concepto {id:“abstract_class”, label:“Clase abstracta (ABC)”, tema:“Programación orientada a objetos”, dificultad:5});

MATCH (a:Concepto{id:“abstract_class”}),(b:Concepto{id:“herencia”}) CREATE (a)-[:REQUIERE]->(b);

Recorrido para la captura en vivo (Neo4j — recomendado para el `.cypher`):

1. Abrir **Neo4j Desktop**, crear/arrancar una base local y abrir el **Neo4j Browser** (`http://localhost:7474`).
2. Cargar el script: pegar el contenido de `salidas/grafos/ekg_lpg.cypher` (o usar `:source`/`cypher-shell < ekg_lpg.cypher`) y ejecutar para crear nodos y aristas.
3. Ejecutar `MATCH (n) RETURN n LIMIT 100;` para visualizar el property graph (propiedades dentro del nodo, aristas `:REQUIERE` tipadas).
4. **Captura:** la vista de grafo del Browser con varios `:Concepto` y sus aristas `:REQUIERE`.
5. Alternativa **Arrows.app**: abrir `https://arrows.app`, dibujar manualmente un fragmento equivalente (nodos `:Concepto` con propiedades `dificultad`/`tema` y arista `:REQUIERE`) reproduciendo la figura `fig_rdf_vs_lpg_fix`.
6. Guardar la imagen en: `B:\Web Semántica y Enlazado de Datos\99-Archivo\capturas\arrows-lpg.png`.

<!– ![Captura en vivo de Neo4j/Arrows.app — property graph del EKG](../../99-Archivo/capturas/arrows-lpg.png) –>

37.2 B.6 RDF Playground

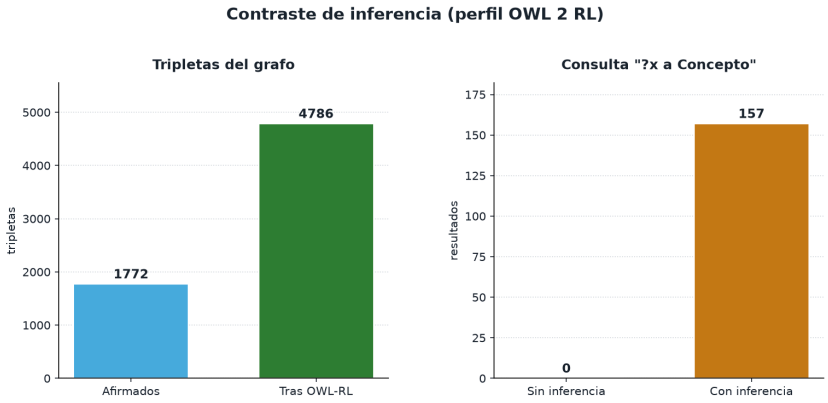

Figura 89. Contraste de inferencia OWL 2 RL. El grafo afirmado pasa de 1772 a 4786 enunciados y la consulta «?x a Concepto» pasa de 0 a 157 resultados. Elaboración propia; no es una captura de RDF Playground.

La captura de la pestaña OWL de `https://rdfplayground.dcc.uchile.cl` queda pendiente de aportar por el autor.

Artefacto y consulta del contraste de inferencia (datos: `ontologia/ekg-python-150.ttl`; consulta: `consultas/02_inferencia_conceptos.rq`):

PREFIX pyedu: <https://w3id.org/ekg-python/schema#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?concepto ?etiqueta

WHERE { ?concepto a pyedu:Concepto ; rdfs:label ?etiqueta .

FILTER(LANG(?etiqueta) = «es») }

ORDER BY ?etiqueta

Recorrido para la captura en vivo:

1. Abrir `https://rdfplayground.dcc.uchile.cl`.
2. En el editor **RDF**, pegar el contenido de `ontologia/ekg-python-150.ttl`.
3. En la pestaña **SPARQL**, pegar la consulta `02_inferencia_conceptos.rq`. Con razonamiento **NONE** la consulta devuelve **0** resultados.
4. Activar la pestaña/conmutador de **razonamiento (RDFS / OWL 2 RL)** y volver a ejecutar: ahora devuelve **157** resultados (y el grafo materializado pasa de 1772 a 4786 enunciados).
5. **Captura:** las dos ejecuciones contrastadas (0 sin inferencia / 157 con inferencia), o la pestaña OWL mostrando el grafo inferido.
6. Guardar la imagen en: `B:\Web Semántica y Enlazado de Datos\99-Archivo\capturas\rdfplayground--owl.png`.

<!– ![Captura en vivo de RDF Playground — contraste de inferencia (0 ⟶ 157)](../../99-Archivo/capturas/rdfplayground-owl.png) –>

37.3 B.7 pyLODE (documentación de la ontología)

pyLODE (Car, s.f.) genera documentación HTML navegable de la TBox del EKG (clases, propiedades de objeto y de datos, etiquetas y comentarios) directamente desde el Turtle, sin interfaz manual. Es la forma estándar de publicar la ontología en formato legible.

Comando exacto (*headless*):

python -m pylode ontologia/ekg-python-150.ttl -o salidas/ekg-python-doc.html

Resultado real: el comando se ejecutó correctamente y produjo el fichero

grafo-ekg/salidas/ekg-python-doc.html (118 540 bytes)

con pyLODE 3.5.1 sobre rdflib 7.6.0. El HTML contiene el índice de clases (`pyedu:Concepto`, `pyedu:Ejercicio`, `pyedu:Tema`, …) y de propiedades del esquema.

Recorrido para la captura en vivo:

1. Ejecutar el comando anterior desde `grafo-ekg/`.
2. Abrir `salidas/ekg-python-doc.html` en el navegador.
3. Desplazarse a la sección **Classes** / **Object Properties** para mostrar la documentación generada.
4. **Captura:** la cabecera de la doc con el índice de clases y propiedades del EKG.
5. Guardar la imagen en: `B:\Web Semántica y Enlazado de Datos\99-Archivo\capturas\pylode--doc.png`.

<!– ![Captura en vivo de pyLODE — documentación HTML de la ontología EKG](../../99-Archivo/capturas/pylode-doc.png) –>

37.4 B.8 Jupyter (cuaderno reproducible)

El cuaderno `notebook/ekg.ipynb` reproduce de extremo a extremo las cifras canónicas (157 conceptos, 1772 ⟶ 4786 triples, SHACL conforme + control negativo 6, 30 `skos:exactMatch`) con RDFLib + owlrl + pyshacl.

Celdas exactas del cuaderno (`notebook/ekg.ipynb`):

38 Celda de carga e inferencia

import rdflib, owlrl

from pathlib import Path

ONTO = Path("../ontologia/ekg-python-150.ttl")

g = rdflib.Graph(); g.parse(ONTO, format="turtle")

print("Triples afirmados:", len(g)) # 1772

owlrl.DeductiveClosure(owlrl.OWLRL_Semantics).expand(g)

print("Triples tras OWL-RL:", len(g)) # 4786

PY = rdflib.Namespace("https://w3id.org/ekg-python/schema#")

n = len(set(g.subjects(rdflib.RDF.type, PY.Concepto)))

print("Instancias de pyedu:Concepto (con inferencia):", n) # 157

39 Celda de validación SHACL (forma del grafo)

from pyshacl import validate

conf, _, _ = validate(str(ONTO), shacl_graph="../ontologia/shapes-ekg.ttl", inference="rdfs")

print("Grafo canónico conforme:", conf) # True (0 violaciones)

conf2, _, txt = validate("../ontologia/ejemplo-invalido.ttl", shacl_graph="../ontologia/shapes-ekg.ttl", inference="rdfs")

print("Control negativo conforme:", conf2, "— violaciones detectadas: 6")

Recorrido para la captura en vivo:

1. Desde `grafo-ekg/notebook/`, lanzar `jupyter lab` (o `jupyter notebook`) y abrir `ekg.ipynb` — alternativamente, *Open in Colab* mediante el badge del cuaderno.
2. Menú *Run → Run All Cells*.
3. Verificar las salidas impresas: `Triples afirmados: 1772`, `Triples tras OWL-RL: 4786`, `Instancias de pyedu:Concepto … 157`, `Grafo canónico conforme: True`.
4. **Captura:** las celdas ejecutadas con esas salidas numéricas visibles (la evidencia de reproducibilidad).
5. Guardar la imagen en: `B:\Web Semántica y Enlazado de Datos\99-Archivo\capturas\jupyter--notebook.png`.

<!– ![Captura en vivo de Jupyter — cuaderno ekg.ipynb ejecutado](../../99-Archivo/capturas/jupyter--notebook.png) –>

39.1 B.9 RDFLib (consultas SPARQL en Python)

RDFLib (RDFLib Team, s.f.) es el motor que carga el Turtle, aplica el cierre OWL-RL (vía owlrl) y ejecuta las consultas SPARQL del entregable sin servidor. Sirve para una captura de consola que evidencia las cifras.

Comando exacto (*headless*):

import rdflib, owlrl

g = rdflib.Graph(); g.parse("ontologia/ekg-python-150.ttl", format="turtle")

print("Triples afirmados:", len(g))

owlrl.DeductiveClosure(owlrl.OWLRL_Semantics).expand(g)

print("Triples tras OWL-RL:", len(g))

PY = rdflib.Namespace("https://w3id.org/ekg-python/schema#")

print("Instancias pyedu:Concepto:", len(set(g.subjects(rdflib.RDF.type, PY.Concepto))))

40 Consulta 01 — conceptos agrupados por tema

print(len(list(g.query(open("consultas/01_conceptos_por_tema.rq", encoding="utf-8").read()))), "temas")

Salida real (ejecutada desde `grafo-ekg/`):

Triples afirmados: 1772

Triples tras OWL-RL: 4786

Instancias pyedu:Concepto: 157

16 temas

Sin el cierre OWL-RL, la misma consulta `?c a pyedu:Concepto` devuelve **0** conceptos tipados explícitamente, y el grafo declara **30** enlaces `skos:exactMatch` a Wikidata (ambos verificados con RDFLib), lo que confirma que la pertenencia a `pyedu:Concepto` es inferida, no afirmada.

Recorrido para la captura en vivo:

1. Abrir una terminal en `grafo-ekg/` (con `export PYTHONIOENCODING=utf-8 PYTHONUTF8=1` en Windows).
2. Ejecutar el script anterior (o `python scripts/consultar.py` para las consultas con inferencia, y `python scripts/consultar.py –sin-inferencia` para el contraste).
3. **Captura:** la consola con las líneas `Triples afirmados: 1772`, `Triples tras OWL-RL: 4786`, `Instancias pyedu:Concepto: 157`.
4. Guardar la imagen en: `B:\Web Semántica y Enlazado de Datos\99-Archivo\capturas\rdflib--consola.png`.

<!– ![Captura en vivo de RDFLib — consola con las cifras del EKG](../../99-Archivo/capturas/rdflib--consola.png) –>

41 Anexo C — Cómo generar cada grafo (snippets)

Comandos exactos para regenerar datos y figuras (desde `grafo-ekg/`, salvo donde se indique):

- **Artefactos del grafo:** `python scripts/inferir.py && python scripts/validar.py && python scripts/consultar.py && python scripts/exportar.py`
- **Datos de la Grafoteca (JSON Cytoscape):** `python scripts/exportar_grafos_json.py` → `salidas/grafos/grafoteca.json`
- **Property graph (Cypher):** `salidas/grafos/ekg_lpg.cypher` (derivado del `.ttl`; cargable en Neo4j o Arrows.app)
- **Esquema ShEx:** `ontologia/ekg-shapes.shex` (validable en RDFShape)
- **Cuaderno reproducible:** `notebook/ekg.ipynb` (RDFLib + owlrl + las 8 consultas + SHACL)
- **Diagramas RDF/TBox con Graphviz** (si `dot` está instalado):

`python -m rdflib.tools.rdf2dot ontologia/ekg-python-150.ttl | dot -Tsvg -o salidas/grafos/ekg_rdf.svg` ·

`python -m rdflib.tools.rdfs2dot ontologia/ekg-python-150.ttl | dot -Tsvg -o salidas/grafos/ekg_tbox.svg`

- **Figuras de grafo de la memoria:** generadas con `networkx` + `matplotlib` sobre `salidas/grafos/grafoteca.json`.
- **Curva de pérdida:** `matplotlib` desde `sistema-rag/modelos/qlora-feedback-v3/loss_history.json`

(claves `step`, `loss`, `eval_loss`; la precisión 0,842 está en `eval_final.json`).

42 Anexo D — Reproducibilidad

Entorno. Python 3.12.10; `rdflib` 7.6, `owlrl`, `pyshacl`, `matplotlib`, `networkx`; `typst-py` y `python-docx`

para construir la memoria; Ollama (`llama3.1:8b`, `qwen2.5:32b`, `nomic-embed-text`) y GPU NVIDIA RTX 5090

(CUDA 12.8) para el sistema RAG y el *fine tuning* QLoRA.

Cifras canónicas (todas reproducibles): 157 conceptos · 1772 ⟶ 4786 triples (OWL 2 RL) · SHACL conforme (0) +

control negativo (6) · 30 `skos:exactMatch` · 19/19/16 SKOS · 16 temas · 10 `sh:NodeShape` · CONSTRUCT 06 = 340/356 ·

federada 20 filas · QLoRA v3 `eval_loss` 1,051 ⟶ 1,028 (precisión 0,842) · benchmark *n*=50, Sistema D categoría 0,76.

Secuencia completa. (1) Regenerar el grafo (Anexo C); (2) `python scripts/exportar_grafos_json.py`;

(3) regenerar las figuras de grafo y la curva de pérdida; (4) `cd memorias && python generar_pdf.py todo` y

`ensamblar_docx.py` en cada memoria; (5) la web `proyecto.html` ya es autocontenida y no requiere construcción.

43 Anexo E — Protocolo de anotación humana

Este anexo documenta el protocolo con el que obtuve la anotación humana que sustenta la validez de criterio del juez automático. Lo recojo aquí, separado del cuerpo de la memoria, porque su valor no está en los resultados (que se discuten en los capítulos de evaluación) sino en la trazabilidad del procedimiento, que permite a cualquier lector reconstruir cómo se recogieron las puntuaciones y bajo qué garantías. Todo lo que sigue describe el método; las cifras que arroja se reportan al final y se interpretan en el capítulo de resultados.

43.1 E.1 Propósito

El objetivo de la anotación fue dotar de validez de criterio a las dimensiones cualitativas que un juez automático (un modelo de lenguaje empleado como evaluador) asigna a las retroalimentaciones. A lo largo de la evaluación constaté que esas puntuaciones del juez dependían del modelo elegido y que el acuerdo entre jueces automáticos de familias distintas era débil, lo que dejaba sin contrastar si las dimensiones cualitativas se correspondían con el criterio de evaluadores humanos competentes. La anotación humana es el instrumento que cierra ese hueco, ya que permite calcular el acuerdo entre anotadores humanos y, sobre todo, contrastar el consenso humano con la puntuación del juez. No sustituye a las métricas objetivas ancladas a verdad de referencia (acierto de categoría y de concepto), que siguen siendo el eje del veredicto, sino que pone a prueba las dimensiones que solo un humano puede juzgar con propiedad.

43.2 E.2 El arnés de anotación

La recogida se realizó con un arnés de anotación, `arnes_anotacion.html`, una mini-aplicación contenida en un único fichero HTML autónomo. La decisión de empaquetarlo todo en un solo fichero fue deliberada y responde a tres exigencias. La primera es la ausencia de fricción técnica. El arnés se abre con un doble clic en cualquier navegador, sin necesidad de instalar dependencias ni de levantar un servidor, lo que lo hace accesible a anotadores con perfiles muy distintos. La segunda es la autonomía. Al no requerir conexión a un servicio externo, cada anotador trabaja por completo en su propio equipo y las puntuaciones nunca abandonan su máquina hasta que él mismo exporta el resultado. La tercera es la reproducibilidad, dado que el arnés se genera de forma determinista a partir del banco de casos held-out mediante un guion (`generar_arnes.py`) que parte de las retroalimentaciones de los sistemas y no incorpora ninguna anotación, por lo que el fichero de partida es idéntico para todos los anotadores y reconstruible cuando se desee.

El arnés presenta al anotador, para cada caso, el fragmento de código del estudiante y las retroalimentaciones que los distintos sistemas produjeron sobre él, y recoge una puntuación en cada una de las tres dimensiones de la rúbrica. Al terminar, exporta un fichero JSON con las valoraciones de ese anotador, que constituye la unidad de depósito del proceso.

43.3 E.3 Las tres dimensiones de la rúbrica

La rúbrica de anotación consta de tres dimensiones, las mismas tres que evalúa el juez automático, lo que es condición necesaria para que la comparación entre ambos sea legítima. Cada dimensión se puntúa en una escala Likert de uno a cinco, donde uno corresponde a una calidad muy mala y cinco a una calidad excelente. Las tres dimensiones son las siguientes.

1. **Divulgativa.** Mide la claridad y la accesibilidad de la explicación del problema, esto es, en qué medida la retroalimentación expone lo que ocurre de una forma comprensible para un estudiante que aún no domina el concepto.
2. **Técnica.** Mide la corrección y la precisión de la explicación técnica, es decir, si lo que la retroalimentación afirma sobre el error y sobre su causa es exacto y está bien fundamentado.
3. **Sugerencia.** Mide la utilidad y el carácter accionable de la mejora propuesta, valorando si la retroalimentación orienta al estudiante hacia un paso siguiente concreto y aprovechable.

El arnés ofrecía además algunos criterios auxiliares orientativos para ayudar a los anotadores a calibrar su juicio.[42] Esos criterios no constituían campos de puntuación independientes, sino que servían de guía para asignar las tres puntuaciones de la rúbrica.

43.4 E.4 Los anotadores

Participaron diez anotadores, identificados únicamente mediante los códigos R01 a R10. Los anotadores pidieron expresamente no publicar sus nombres ni su filiación, petición que respeto íntegramente. Este anexo no contiene ningún dato personal y se limita a describir el perfil profesional genérico de cada uno, suficiente para acreditar su competencia sin comprometer su privacidad. El reparto de perfiles fue de siete profesionales de la programación y tres docentes universitarios, todos con conocimiento profundo de Python. Los perfiles, de forma genérica, son los siguientes.

- **R01** — Ingeniero de software sénior de backend, con nueve años de experiencia en Python, Java y Go.
- **R02** — Desarrollador fullstack, con seis años de experiencia en Django y React en el sector financiero.
- **R03** — *Docente universitario* de Informática, con doce años de docencia en programación y estructuras de datos y doctorado en Ciencias de la Computación.
- **R04** — Ingeniero de datos, con cinco años de experiencia en procesos de extracción, transformación y carga con Python y SQL.
- **R05** — Ingeniero de operaciones y fiabilidad, con siete años de experiencia en Python, Bash y herramientas de infraestructura.
- **R06** — *Profesor titular* de Ingeniería del Software en una universidad pública, con quince años de trayectoria e investigación en calidad del software educativo.
- **R07** — Desarrollador frontend sénior, con ocho años de experiencia en Python y TypeScript.
- **R08** — Ingeniero de aprendizaje automático, con cuatro años de experiencia, máster en Inteligencia Artificial y trabajo en Python y PyTorch.
- **R09** — Ingeniero de calidad y pruebas, con seis años de experiencia en marcos de prueba y certificación profesional en la materia.
- **R10** — *Docente universitario* de programación, con once años de experiencia docente, e ingeniero de software.

La presencia de los tres docentes universitarios (R03, R06 y R10) entre los anotadores es relevante, pues aporta al panel un criterio pedagógico experto que complementa la mirada profesional de los siete programadores y que motiva, en el análisis posterior, el uso de umbrales conservadores en las pruebas de sustitución del juez.

43.5 E.5 Procedimiento a ciegas

La anotación se realizó a ciegas. Cada anotador puntuó las retroalimentaciones sin saber qué sistema había producido cada una. El arnés presentaba las salidas de los cuatro sistemas con su procedencia oculta y con el orden de los sistemas barajado, de modo que ni la posición ni ninguna marca delatara el origen de cada texto. Esta condición es esencial para la validez del proceso, porque elimina el sesgo que se introduciría si un anotador supiese, por ejemplo, que una retroalimentación procede del modelo afinado o del aumentado con grafo, y le impide trasladar a la puntuación cualquier expectativa previa sobre qué sistema debería ser mejor. El anotador juzga, en suma, la calidad del texto que tiene delante y solo eso.

[42] la precisión en la identificación del error, la personalización al problema concreto del estudiante, la presencia de un ejemplo de código correcto y la claridad de la solución

43.6 E.6 Recuento

El diseño de la anotación cubre cincuenta casos held-out, los mismos sobre los que se ejecutó el banco de evaluación comparada, y para cada caso las retroalimentaciones de los cuatro sistemas (designados A, B, C y D). El recuento por anotador y total es el siguiente.

Unidad	Por anotador	Total (diez anotadores)
Casos held-out	50	50 (idénticos en todos)
Retroalimentaciones valoradas (50 casos × 4 sistemas)	200	2.000
Puntuaciones Likert (200 × 3 dimensiones)	600	6.000

Es decir, cada anotador valoró doscientas retroalimentaciones (las de cincuenta casos por cuatro sistemas) y, al puntuar cada una en las tres dimensiones, emitió seiscientas puntuaciones Likert. Con los diez anotadores, el proceso reúne dos mil filas de valoración y seis mil puntuaciones Likert en total. Los cincuenta casos son idénticos en los diez ficheros depositados, pues todos los anotadores trabajaron sobre el mismo conjunto held-out, y todas las celdas quedaron rellenas.

43.7 E.7 Cómputo estadístico

Sobre las puntuaciones depositadas se calcularon los estadísticos de acuerdo habituales para escalas ordinales con múltiples anotadores, cada uno apropiado para una pregunta distinta.

- **Kappa de Fleiss.** Mide el acuerdo entre los diez anotadores corrigiendo el acuerdo esperable por azar, tratando las puntuaciones como categorías. Es la medida más exigente con la granularidad de la escala, porque penaliza las discrepancias de un solo punto entre anotadores como si fueran desacuerdos plenos.
- **Alfa de Krippendorff.** Mide igualmente el acuerdo entre anotadores, pero en su variante ordinal, que pondera la magnitud del desacuerdo, de modo que una diferencia de un punto pesa menos que una de tres. Es, por su naturaleza ordinal, la medida más adecuada a una escala Likert y la que mejor refleja el acuerdo cuando las discrepancias son de baja magnitud.
- **Coeficiente de correlación intraclase, ICC(2,k).** Estima la fiabilidad del *promedio* de los diez anotadores bajo un modelo de efectos aleatorios de dos vías con acuerdo absoluto. Responde a una pregunta distinta de las anteriores: si la media del panel es un estimador estable de la calidad aunque cada anotador, por separado, concuerde poco a causa de la granularidad de la escala.

Las tres medidas se reportan de forma global y desglosadas por dimensión, y a partir de ellas, junto con el consenso humano, se contrasta la validez de criterio del juez automático. Sobre el conjunto recogido, el panel humano arrojó un Fleiss κ global de 0,096, un alfa de Krippendorff ordinal global de 0,318 y un ICC(2,k) de acuerdo absoluto global de 0,831. La lectura de estas cifras, su desglose por dimensión y su contraste con el juez automático se desarrollan en el capítulo de resultados, donde se discute por qué la fiabilidad del promedio del panel convive con un acuerdo por anotador modesto y qué implica todo ello para la validez del juez.

44 Anexo F — Instrucciones de aplicación e infraestructura

Este anexo recoge las instrucciones de aplicación e infraestructura del sistema descrito en el cuerpo de la memoria, esto es, el procedimiento concreto para ponerlo a disposición de la universidad y, en su caso, integrarlo en la docencia del Máster. Lo separo del cuerpo principal porque su valor no es argumentativo sino operativo: mientras los capítulos anteriores justifican las decisiones de diseño de EKG-Python y discuten sus resultados, lo que sigue es un manual de despliegue. El lector encontrará aquí qué componentes intervienen, cómo se reparten las responsabilidades entre el proyecto y el servicio de informática de la UNED, los pasos reproducibles para instalar y publicar el sistema y las posibilidades de uso académico, junto con los requisitos y el mantenimiento que todo ello conlleva.

44.1 F.1 Resumen del proyecto y valor docente

EKG-Python es un sistema que combina un **grafo de conocimiento educativo** sobre programación en lenguaje Python con técnicas de recuperación aumentada sobre grafos (**GraphRAG**) y un mecanismo de **retroalimentación trazable** para el aprendizaje. En lugar de tratar el conocimiento de programación como texto plano, el sistema lo modela de forma explícita: conceptos (tipos de datos, estructuras de control, funciones, excepciones), errores frecuentes del alumnado, buenas prácticas y las relaciones entre todos ellos quedan representados como entidades y relaciones formales en un grafo RDF consultable mediante SPARQL.

Sobre esa base semántica, el componente de GraphRAG permite que un modelo de lenguaje genere retroalimentación apoyándose en hechos recuperados del grafo, de modo que cada explicación que recibe el estudiante puede **vincularse a los nodos y relaciones concretos** que la justifican. Esa propiedad —la trazabilidad— es la que distingue al sistema de un asistente generativo convencional y la que lo hace especialmente valioso en un contexto educativo.

44.1.1 Valor docente

El proyecto aporta valor en el Máster en dos planos complementarios:

- **Como recurso didáctico.** Ofrece a estudiantes y profesorado un grafo navegable que organiza el conocimiento de programación de forma estructurada, junto con un asistente que explica errores de código apoyándose en ese grafo. Sirve para ilustrar conceptos de programación y, sobre todo, para mostrar cómo se construye y se explota conocimiento formalizado.
- **Como caso de estudio real de Web Semántica y LLMs.** Reúne en un único sistema extremo a extremo una ontología de dominio, un almacén RDF con endpoint SPARQL, una interfaz de exploración visual del grafo y la integración con un modelo de lenguaje mediante GraphRAG. Es, por tanto, un ejemplo tangible y reproducible de los contenidos teóricos del Máster, no una mera maqueta.

> En síntesis: el sistema permite enseñar con el grafo (recurso de apoyo al aprendizaje de Python) y enseñar sobre el grafo (caso real que materializa la unión entre Web Semántica y modelos de lenguaje).

44.2 F.2 Arquitectura de despliegue

La arquitectura se ha diseñado para ser **modular y de bajo acoplamiento**: cada componente puede instalarse en una misma máquina o repartirse entre varias, y el único elemento estrictamente necesario para el funcionamiento básico es el almacén RDF con su endpoint SPARQL. El componente de modelo de lenguaje local es opcional y solo se requiere si la institución desea ofrecer la retroalimentación generativa sin depender de servicios externos.

44.2.1 F.2.1 Componentes del sistema

Componente	Función	Tecnología	¿Obligatorio?
Almacén RDF / triplestore	Aloja el grafo de conocimiento y sirve el endpoint SPARQL para consultas	GraphDB (edición Free)	Sí
Grafo de conocimiento	Ontología y datos del dominio de programación en Python	RDF/Turtle (.ttl) + OWL/RDFS	Sí
Sitio web	Interfaz de exploración del grafo y consulta de retroalimentación	FastAPI (backend) + Cytoscape.js (visualización)	Sí
Servidor de modelo local	Genera la retroalimentación mediante GraphRAG sin servicios externos	Ollama (con o sin GPU)	Opcional

El flujo de funcionamiento es directo: el sitio web recibe la consulta del usuario (una pregunta o un fragmento de código), recupera del endpoint SPARQL los hechos pertinentes del grafo y, si está habilitado el modelo local, los pasa a Ollama para redactar una explicación trazable; la visualización con Cytoscape.js permite, en paralelo, navegar el subgrafo implicado.

44.2.2 F.2.2 Reparto de responsabilidades: qué aportamos y qué provee la UNED

Para evitar ambigüedades en la planificación, conviene separar con claridad los **entregables del proyecto** (que ya preparamos y entregamos listos) de los **recursos de infraestructura** que corresponde proveer a la Universidad a través de su servicio de informática.

Lo que aportamos nosotros	Lo que provee la UNED
La ontología y el grafo de conocimiento (.ttl) ya poblado y validado	Un servidor Linux (físico o máquina virtual) con acceso de administración
El sitio web completo (FastAPI + Cytoscape.js) y sus recursos estáticos	Un nombre de dominio o subdominio institucional y, en su caso, certificado TLS
Los scripts de instalación, carga del grafo y arranque de servicios	Los recursos de cómputo (CPU, RAM, disco) y, si se desea el modelo local, GPU
La configuración base de GraphDB, FastAPI y Ollama (ficheros y parámetros)	La conectividad de red, las reglas de cortafuegos y las copias de seguridad
La documentación técnica y este manual de despliegue	Las cuentas de acceso y la autorización para publicar en la red de la Universidad

> Importante: nuestro papel se limita a preparar y entregar la infraestructura software lista para desplegar. La provisión del servidor, el dominio y los recursos de cómputo, así como las decisiones sobre ubicación y políticas de seguridad, corresponden al servicio de informática de la UNED, con quien se confirmarán los detalles concretos antes del despliegue.

44.2.3 F.2.3 Cómo alojan las universidades los recursos docentes

Las universidades españolas disponen, de forma generalizada, de servicios centralizados de informática que ofrecen alojamiento de servidores para docencia e investigación. Las modalidades habituales, que sirven de referencia realista para este proyecto, son las siguientes:

- **Máquina virtual institucional (hosting).** El servicio de informática crea una máquina virtual sobre su propia infraestructura con las especificaciones acordadas. Las configuraciones por defecto suelen

partir de 2 CPU virtuales, 2 GB de RAM y varias decenas o centenas de GB de disco, ampliables bajo petición; es la opción más común y la más adecuada para EKG-Python.

- **Alojamiento físico (housing).** La Universidad acoge en su centro de proceso de datos un equipo proporcionado por el proyecto, con suministro eléctrico protegido y conexión a la red en una subred aislada. Es útil cuando se requiere hardware específico, como una GPU dedicada.
- **Hosting de sitios web institucionales.** Para componentes estáticos o de poca carga, muchos servicios ofrecen espacio web bajo dominio institucional, adecuado para publicar la interfaz si se separa del backend.
- **Infraestructura de cómputo y GPU compartida.** Para cargas con modelos de lenguaje, es frecuente el acceso a recursos de cómputo compartidos o a escritorios y servidores virtuales con GPU, gestionados por el servicio de informática y reservables por proyecto o asignatura.

En la práctica, la configuración recomendada para EKG-Python es una **máquina virtual Linux institucional** que aloje GraphDB y el sitio web, con la posibilidad de añadir una GPU —dedicada o compartida— únicamente si se activa el modelo local con Ollama. Los detalles concretos (tipo de servidor, dimensionamiento, dominio y políticas de respaldo y seguridad) **se confirmarán con el servicio de informática de la UNED**, pues dependen de su catálogo de servicios y de sus normas internas.

44.3 F.3 Pasos de despliegue

Los pasos siguientes son concretos y reproducibles sobre un servidor Linux estándar (por ejemplo, Ubuntu Server LTS). Asumen acceso por línea de comandos con permisos de administración y conectividad de red. Los puertos indicados pueden ajustarse a las normas de la institución.

Fase	Objetivo	Resultado esperado
1. Preparar el servidor	Dejar la máquina lista (sistema, Java, usuario de servicio)	Servidor Linux operativo con Java 21+
2. Instalar GraphDB	Desplegar el triplestore	Workbench accesible en el puerto 7200
3. Cargar el grafo (.ttl)	Poblar el repositorio con la ontología y los datos	Grafo importado y consultable
4. Exponer el endpoint SPARQL	Publicar el punto de consulta	Endpoint SPARQL estable y protegido
5. Publicar el sitio web	Levantar FastAPI + Cytoscape.js	Interfaz accesible vía dominio institucional
6. (Opcional) Modelo local	Activar la retroalimentación generativa	Ollama sirviendo el modelo

44.3.0.1 Paso 1. Preparar el servidor

Actualizar el sistema, instalar Java 21 (requisito de GraphDB) y crear un usuario de servicio dedicado:

sudo apt update && sudo apt upgrade -y

sudo apt install -y openjdk-21-jre-headless unzip curl

sudo useradd -r -m -d /opt/ekg -s /usr/sbin/nologin ekg

java -version # comprobar que es 21 o superior

44.3.0.2 Paso 2. Instalar GraphDB

Descomprimir la distribución de GraphDB y arrancarla como servicio. Tras el arranque, el Workbench queda disponible en el puerto 7200:

sudo unzip graphdb-desktop-*.zip -d /opt/ekg/graphdb

45 arranque manual para verificar

/opt/ekg/graphdb/bin/graphdb

46 Workbench: http://SERVIDOR:7200/

Para producción conviene registrarlo como servicio del sistema (systemd) para que se inicie automáticamente y se gestione con journalctl.

46.0.0.1 Paso 3. Cargar el grafo (.ttl)

Crear un repositorio en GraphDB e importar el fichero Turtle entregado. La carga puede hacerse desde el Workbench o vía HTTP con cURL:

curl -X POST \

-H “Content-Type: application/x-turtle” \

–data-binary @ekg-python.ttl \

http://SERVIDOR:7200/repositories/ekg/statements

46.0.0.2 Paso 4. Exponer el endpoint SPARQL

El endpoint queda servido por GraphDB en la ruta del repositorio. Se recomienda dejarlo en modo de solo lectura para el público y colocarlo tras un proxy inverso con TLS:

47 Endpoint de consulta (lectura):

http://SERVIDOR:7200/repositories/ekg

48 Prueba rápida:

curl -s “http://SERVIDOR:7200/repositories/ekg” \

–data-urlencode “query=SELECT * WHERE { ?s ?p ?o } LIMIT 5” \

-H “Accept: application/sparql-results+json”

48.0.0.1 Paso 5. Publicar el sitio web

Instalar las dependencias del backend FastAPI y servirlo con un servidor ASGI; la visualización con Cytoscape.js se entrega como recursos estáticos. Conviene situarlo tras un proxy inverso (Nginx) bajo el dominio institucional:

cd /opt/ekg/sitio

python3 -m venv .venv && source .venv/bin/activate

pip install -r requirements.txt

49 variable de entorno con el endpoint SPARQL

export SPARQL_ENDPOINT=“http://localhost:7200/repositories/ekg”

uvicorn app:app –host 0.0.0.0 –port 8000

49.0.0.1 Paso 6. (Opcional) Activar el modelo local con Ollama

Solo si se desea la retroalimentación generativa autónoma. Ollama funciona con CPU, aunque una GPU acelera la inferencia de forma considerable:

curl -fsSL https://ollama.com/install.sh | sh

ollama pull <modelo-de-codigo>

ollama serve # expone la API local en el puerto 11434

50 el sitio se configura para apuntar a esta API

Cada paso es independiente: el sistema es plenamente funcional para la exploración del grafo y la consulta SPARQL tras el paso 5, aun sin el modelo local del paso 6.

50.1 F.4 Propuesta de integración docente

Esta sección plantea **posibilidades** de uso académico del sistema. Conviene subrayar, antes de detallarlas, que la decisión sobre en qué asignatura y con qué alcance se integra el proyecto **corresponde al director del trabajo y al equipo docente** del Máster. Nuestra aportación se circunscribe a preparar y entregar la infraestructura lista para usarse; las opciones que siguen son orientativas.

50.1.1 F.4.1 Encaje como módulo, práctica o demostración

Según el grado de implicación que decida el equipo docente, el sistema puede incorporarse en tres formatos:

- **Demostración guiada (sesión única).** El profesorado presenta el sistema en vivo: navega el grafo, lanza consultas SPARQL y muestra cómo se genera retroalimentación trazable. Requiere mínima preparación y sirve como ilustración de los contenidos.
- **Práctica de laboratorio (una o varias sesiones).** El alumnado interactúa directamente con el endpoint y la interfaz, resolviendo ejercicios de consulta y exploración. Es el formato con mejor relación entre esfuerzo y aprendizaje.
- **Módulo o proyecto evaluable.** El sistema se usa como base de un trabajo más amplio (extender la ontología, añadir consultas, comparar estrategias de RAG), integrándose en la evaluación de la asignatura.

En cuanto a la asignatura de acogida, y siempre a criterio del equipo docente, el proyecto encaja de forma natural en varias materias del Máster:

Asignatura (orientativa)	Aspecto del sistema que ilustra
Web Semántica y Enlazado de Datos	Ontología, RDF/Turtle, consultas SPARQL y publicación de un endpoint; es el encaje más directo
Métodos de Aprendizaje Automático	Integración de un modelo de lenguaje y la arquitectura Graph-RAG sobre conocimiento estructurado
Sistemas Inteligentes	Visión de sistema extremo a extremo: representación del conocimiento, razonamiento y retroalimentación al usuario

La tabla anterior es una sugerencia. La elección final de la asignatura, el formato y la carga de trabajo es competencia del director y del equipo docente; el proyecto se entrega preparado para adaptarse a cualquiera de estas opciones.

50.1.2 F.4.2 Actividades para el alumnado

A modo de ejemplo, estas actividades pueden plantearse directamente sobre el sistema desplegado, con dificultad creciente:

1. **Consultas SPARQL básicas.** Recuperar todos los conceptos de un tema (por ejemplo, las estructuras de control) y los errores asociados a cada uno, familiarizándose con la sintaxis SELECT/WHERE.
2. **Exploración visual del grafo.** Usar la interfaz de Cytoscape.js para navegar desde un concepto hasta sus errores frecuentes y buenas prácticas relacionadas, identificando los caminos del grafo.

3. **Consultas SPARQL avanzadas.** Construir consultas con filtros, agregaciones o propiedades de camino para responder preguntas como qué conceptos concentran más errores o qué prácticas mitigan un error dado.
4. **Análisis de la retroalimentación trazable.** Pedir al sistema una explicación sobre un fragmento de código con un error e identificar qué nodos del grafo sustentan la respuesta, evaluando su pertinencia.
5. **Comparación de feedback entre sistemas.** Contrastar la retroalimentación trazable de EKG-Python con la de un asistente generativo sin grafo, valorando precisión, justificación y utilidad pedagógica de cada uno.

50.2 F.5 Requisitos, mantenimiento y consideraciones

Los requisitos de cómputo dependen sobre todo de si se activa el modelo local. La tabla resume dos perfiles de referencia que deberán confirmarse con el servicio de informática:

Recurso	Sin modelo local (básico)	Con modelo local (Ollama)
CPU	2–4 núcleos	4–8 núcleos
RAM	4–8 GB	16 GB o más
Disco	20–40 GB	40–80 GB (pesos del modelo)
GPU	No necesaria	Recomendable (acelera la inferencia)
Software	Java 21+, Python 3, GraphDB	Lo anterior + Ollama

50.2.1 F.5.1 Licencias, datos y privacidad

- **Licencias.** Los componentes de terceros se emplean en ediciones y términos compatibles con el uso académico (GraphDB en su edición Free, FastAPI, Cytoscape.js y Ollama como software de código abierto). Los materiales propios del proyecto —ontología, grafo, sitio y scripts— se entregan con una licencia de uso docente acordada con la dirección del trabajo.
- **Datos.** El grafo contiene conocimiento de dominio sobre programación (conceptos, errores, buenas prácticas) y no incorpora datos personales del alumnado. Cualquier ejercicio o código que los estudiantes introduzcan se procesa para generar la retroalimentación y no necesita almacenarse.
- **Privacidad.** Con el modelo local activado mediante Ollama, la inferencia ocurre íntegramente dentro de la infraestructura de la Universidad, sin enviar el código de los estudiantes a servicios externos, lo que favorece el cumplimiento de las políticas de protección de datos. Si en el futuro se registrara actividad del alumnado con fines de mejora, deberá hacerse conforme al RGPD y a la normativa interna de la UNED.

50.2.2 F.5.2 Mantenimiento y operación

- **Copias de seguridad.** Respaldar periódicamente el fichero .ttl original y, si se han hecho cambios, exportar el repositorio de GraphDB; el grafo es el activo crítico.
- **Actualizaciones.** Mantener al día el sistema operativo, la versión de GraphDB y las dependencias del sitio; las actualizaciones del modelo en Ollama son opcionales.
- **Supervisión.** Vigilar la disponibilidad del endpoint SPARQL y del sitio web, y revisar los registros para detectar errores o un uso anómalo.
- **Evolución del grafo.** La ontología puede ampliarse con nuevos conceptos o errores; el diseño modular permite recargar el .ttl sin afectar al resto de componentes.
- **Coordinación.** Acordar con el servicio de informática de la UNED las ventanas de mantenimiento, los responsables de cada tarea y el canal de incidencias.

El sistema está pensado para una operación liviana: una vez desplegado, el mantenimiento ordinario se reduce a copias de seguridad del grafo, actualizaciones de seguridad y supervisión básica de los dos servicios principales.