Escuela Técnica Superior de Ingeniería

Máster Universitario en Inteligencia Artificial

TRABAJO FIN DE MÁSTER

Asistente Docente IA Integrado en Moodle

Adolfo Peña Marín

Tutores:

Diego Canales Aguilera

Antonio Manuel Durán Rosal

Sevilla, julio de 2026

Diego Canales Aguilera, Profesor Ayudante Doctor del Dpto. de Métodos Cuantitativos y la Escuela Superior de Ingeniería de la Universidad Loyola Andalucía.

Antonio Manuel Durán Rosal, Profesor Titular del Dpto. de Métodos Cuantitativos y la Escuela Superior de Ingeniería de la Universidad Loyola Andalucía.

Informan:

Que el presente Trabajo Fin de Máster titulado *Asistente Docente IA Integrado en Moodle*, constituye la memoria presentada por **Adolfo Peña Marín** para aspirar al título de Máster Universitario en Inteligencia Artificial, y ha sido realizado bajo nuestra dirección en la Escuela Superior de Ingeniería de la Universidad Loyola Andalucía reuniendo, a nuestro juicio, las condiciones necesarias exigidas en este tipo de trabajos.

Y para que conste, se expide y firma el presente certificado en Sevilla, Julio 2026.

Los directores:

Fdo: Prof. Dr. Diego Canales Aguilera

Fdo: Prof. Dr. Antonio M. Durán Rosal

Resumen

Este Trabajo Fin de Máster tiene como objetivo diseñar, implementar y evaluar un asistente docente basado en generación aumentada por recuperación (RAG), integrado en Moodle y orientado a responder preguntas a partir de los materiales reales de una asignatura. El trabajo parte de tres requisitos: que las respuestas puedan rastrearse hasta sus fuentes, que el sistema se abstenga cuando no disponga de evidencia suficiente y que el tratamiento de la información permanezca bajo control institucional mediante una ejecución local o *self-hosted*.

La solución separa la ingesta documental, la consulta y la interfaz de usuario. Moodle actúa como fuente de los recursos docentes; su interfaz REST permite recuperar los documentos; FastAPI expone el servicio de consulta; y los bloques Moodle en PHP funcionan como capa de interacción. El motor RAG se implementa en Python, combina recuperación léxica y densa, almacena las representaciones en ChromaDB y genera las respuestas con un modelo local servido mediante Ollama.

La evaluación se realiza sobre el corpus real de un curso de posgrado y analiza de forma diferenciada la recuperación, la generación, la citación, la abstención y el coste operativo. Los resultados muestran que la recuperación híbrida ofrece un equilibrio adecuado entre cobertura semántica y coincidencia léxica, aunque BM25 mantiene ventajas en algunas métricas de ordenación. El mecanismo de abstención permite rechazar consultas sin evidencia suficiente y la mayoría de las respuestas generadas incorpora fuentes y citas. No obstante, la comprobación realizada es estructural, no semántica, y detecta duplicaciones frecuentes. Asimismo, el reordenamiento avanzado no justifica su coste en la configuración evaluada y la generación concentra la mayor parte de la latencia.

El resultado es un prototipo funcional y modular para el entorno estudiado. Su arquitectura puede servir como base para otros cursos, pero los parámetros, el rendimiento y la utilidad educativa deben recalibrarse y validarse con nuevos corpus, varios anotadores y usuarios reales.

Palabras clave: generación aumentada por recuperación, Moodle, inteligencia artificial educativa, recuperación híbrida, modelos de lenguaje locales, privacidad, trazabilidad.

Universidad LOYOLA

Abstract

This Master's Thesis aims to design, implement and evaluate a teaching assistant based on retrieval-augmented generation (RAG), integrated into Moodle and intended to answer questions from the actual materials of a course. The work is guided by three requirements: answers must be traceable to their sources, the system must abstain when sufficient evidence is unavailable, and information processing must remain under institutional control through local or self-hosted execution.

The solution separates document ingestion, querying and the user interface. Moodle acts as the source of teaching resources; its REST interface retrieves the documents; FastAPI exposes the query service; and the Moodle blocks written in PHP provide the interaction layer. The RAG engine is implemented in Python, combines lexical and dense retrieval, stores the representations in ChromaDB, and generates answers with a local model served through Ollama.

The evaluation uses the real corpus of a postgraduate course and analyses retrieval, generation, citation, abstention and operational cost separately. The results show that hybrid retrieval provides an appropriate balance between semantic coverage and lexical matching, although BM25 retains advantages in some ranking metrics. The abstention mechanism rejects queries for which sufficient evidence is unavailable, and most generated answers include sources and citations. However, the checking performed is structural rather than semantic and frequently detects duplicate markers. In addition, advanced reranking does not justify its cost in the evaluated configuration, while generation accounts for most of the total latency.

The outcome is a functional and modular prototype for the evaluated setting. Its architecture may provide a basis for other courses, but its parameters, performance and educational usefulness require recalibration and validation with new corpora, multiple annotators and real users.

Keywords: retrieval-augmented generation, Moodle, artificial intelligence in education, hybrid retrieval, local language models, privacy, traceability.

Índice general

Índice de figuras

Índice de tablas

I

Parte

Planteamiento del trabajo

1 Introducción

Este capítulo presenta el marco general del Trabajo Fin de Máster (TFM) y delimita el problema de investigación abordado. Para ello, se describe el contexto que motiva el desarrollo del sistema, se formula la hipótesis de trabajo y se establecen los objetivos y las preguntas de investigación que orientan su diseño y evaluación. Asimismo, se concreta el alcance del proyecto, se identifican sus principales aportaciones y se resume la estructura seguida en el resto de la memoria.

1.1. Contexto y motivación

Los sistemas de asistencia inteligente al estudio se han incorporado a los entornos de aprendizaje digital con un ritmo acelerado durante los últimos años, en paralelo a la consolidación de los grandes modelos de lenguaje (*Large Language Models*, LLM) como interfaz conversacional de propósito general [1, 2, 3]. En el ámbito universitario, sin embargo, la integración de esta tecnología con las plataformas institucionales de aprendizaje, con `Moodle` como referencia mayoritaria[1], sigue presentando huecos significativos: la mayoría de los asistentes disponibles operan de forma externa al aula virtual, no utilizan como fuente primaria los materiales del propio curso y no ofrecen garantías verificables sobre el origen de sus respuestas. El marco teórico que sustenta el enfoque adoptado se desarrolla en el Capítulo 2.

Este trabajo se sitúa en ese cruce entre asistencia educativa y gestión documental institucional. Adopta la generación aumentada por recuperación (*Retrieval Augmented Generation*, RAG) [4, 5] como respuesta razonable a la necesidad de ofrecer respuestas trazables sobre el material del curso sin abandonar las restricciones de privacidad propias de una universidad. Su desarrollo se enmarca, además, en la actividad de Loyola Teams, Equipo de IA para la vida universitaria, en la Universidad Loyola Andalucía.

[1] https://stats.moodle.org/

1.2. Problema abordado

El problema concreto que este TFM aborda es la ausencia de asistentes docentes locales capaces de responder únicamente con el material del curso, citando sus fuentes y respetando las restricciones de privacidad de la institución. Las soluciones comerciales actuales resuelven parcialmente la conversación pero introducen dependencias externas, como el envío de contenido institucional a `APIs` de terceros, ausencia de trazabilidad y falta de control sobre las políticas de retención, que resultan incompatibles con los requisitos habituales de una universidad pública o privada con datos protegidos (Reglamento General de Protección de Datos, RGPD) [6].

1.3. Hipótesis del trabajo

La hipótesis que orienta el trabajo sostiene que una arquitectura RAG modular, integrada con `Moodle` mediante interfaces estándar, ejecutada en local o en infraestructura *self-hosted* y dotada de mecanismos de abstención y trazabilidad, puede permitir ofrecer asistencia educativa útil sin comprometer la privacidad ni la verificabilidad de las respuestas. Esta hipótesis se contrasta empíricamente en los capítulos experimentales, a través del diseño descrito en el Capítulo 6 y de los resultados presentados en el Capítulo 7.

1.4. Objetivo general

El objetivo general del trabajo consiste en diseñar, implementar y evaluar una arquitectura RAG multiformato, sensible al contexto y compatible con `Moodle`, capaz de responder a consultas docentes usando únicamente el material del curso, con citación verificable y con mecanismos explícitos de abstención cuando la evidencia disponible es insuficiente.

1.5. Objetivos específicos

Los objetivos específicos que soportan el objetivo general se formalizan en el Capítulo 3. En este capítulo se presentan en forma resumida:

- Revisar de forma crítica el estado del arte en sistemas RAG aplicados a entornos educativos y de gestión documental.

- Definir una arquitectura modular que separe con claridad la ingesta documental, la recuperación, la generación y la integración con `Moodle`.
- Implementar un prototipo funcional que soporte materiales en varios formatos (`PDF`, `DOCX`, `PPTX`, `HTML`, `Markdown`, `notebooks`).
- Construir un conjunto de preguntas de evaluación anotado y reutilizable, alineado con el corpus real de un curso.
- Diseñar y ejecutar un protocolo experimental que cubra recuperación, generación, citación y abstención.
- Documentar las limitaciones del prototipo y las líneas de trabajo posteriores.

1.6. Preguntas de investigación

El trabajo se articula en torno a cuatro preguntas de investigación que se responden empíricamente en los capítulos experimentales:

1. ¿En qué medida una arquitectura RAG con recuperación híbrida y abstención calibrada mejora la calidad de las respuestas frente a *baselines* léxicos o densos aislados sobre el corpus de un curso universitario?
2. ¿Es viable ejecutar el sistema completo con modelos locales *privacy-first* sin degradar la utilidad para el estudiante ni comprometer la latencia percibida?
3. ¿Qué componentes del *pipeline* (recuperación, *re-ranking*, política de *prompt*, abstención) aportan el mayor valor incremental cuando se ajustan de forma individual?
4. ¿Qué límites reales presenta la evaluación asistida por jueces LLM locales frente a las métricas deterministas y a la revisión humana?

1.7. Alcance y exclusiones

El alcance del trabajo se delimita explícitamente para acotar las conclusiones a lo efectivamente realizado.

Se incluye en el alcance: el diseño, la implementación y la evaluación de un prototipo funcional sobre el corpus documental de un curso real de posgrado (*course id*=2, ámbito MLOps); la integración con `Moodle` mediante la `REST API` de `Moodle`, su

interfaz de programación de aplicaciones (*Application Programming Interface*, API) de tipo transferencia de estado representacional (*Representational State Transfer*, REST), como canal de ingesta documental, el servicio `FastAPI` que expone el *endpoint* `/ask` como interfaz de consulta al RAG y un bloque `Moodle` en `PHP` restringido a la interfaz de usuario dentro del entorno educativo.

Queda fuera del alcance: el desarrollo de un *plugin* institucional completo (con administración, permisos y gestión de roles); la integración con otros gestores de contenido educativo distintos de `Moodle`; el despliegue institucional a escala; la evaluación multi-curso; la evaluación con usuarios reales en el aula; y el uso de servicios comerciales o de infraestructura externa de terceros para cualquier parte del *pipeline*, en coherencia con el compromiso *privacy-first* que orienta el trabajo. Los elementos susceptibles de continuación se documentan como líneas de trabajo futuro en el Capítulo 9.

1.8. Aportaciones del trabajo

Las aportaciones principales del trabajo se agrupan distinguiendo lo implementado, lo evaluado y la discusión de limitaciones asociada.

- **Implementadas:** una arquitectura RAG modular reproducible, con separación limpia entre ingesta, recuperación, generación e integración con `Moodle`; una integración funcional con `Moodle` mediante la `REST API` de `Moodle` para la ingesta documental, `FastAPI /ask` como interfaz de consulta al RAG y dos bloques `PHP` de `Moodle` funcionales dentro del alcance del prototipo (chat RAG e indexación) que actúan como interfaces finas sobre `FastAPI`, manteniendo el motor RAG en `Python`; un mecanismo de abstención ante evidencia insuficiente; y una capa de verificación estructural de citas que refuerza la trazabilidad del sistema.
- **Evaluadas:** la calibración empírica del umbral de abstención; y un marco de evaluación multi-fase con métricas deterministas a nivel de fragmento, contrastado con evaluación asistida por jueces LLM locales y con la revisión del autor sobre un corpus real de curso.

Como bloque separado de las aportaciones anteriores, la memoria incluye una discusión honesta de los límites de la evaluación asistida por LLM locales bajo restricciones *privacy-first* y una enumeración de las líneas de trabajo futuro (entre ellas, el plugin institucional completo, la evaluación con usuarios reales y el escalado multi-curso). El desglose detallado y los datos que sostienen cada aportación aparecen en los Capítulos 7 y 9.

1.9. Estructura de la memoria

La memoria se organiza en nueve capítulos. El Capítulo 1 presenta el problema, la hipótesis y las preguntas de investigación. El Capítulo 2 revisa el estado del arte en sistemas RAG y su aplicación en entornos educativos. El Capítulo 3 formaliza los objetivos, los requisitos derivados del problema y la metodología de investigación aplicada. El Capítulo 4 describe la arquitectura de referencia del sistema y su integración con `Moodle`. El Capítulo 5 detalla la implementación del prototipo. El Capítulo 6 presenta el diseño experimental y el protocolo de evaluación. El Capítulo 7 recoge los resultados obtenidos. El Capítulo 8 los interpreta y discute a la luz de las preguntas de investigación. El Capítulo 9 cierra el trabajo con las conclusiones y las líneas futuras. Seis anexos técnicos complementan la memoria con material de referencia sobre despliegue reproducible, contratos de `API`, estructura del repositorio, plantillas de *prompt*, *dataset* de evaluación y resultados extendidos.

2 Estado del arte

Este capítulo presenta el marco teórico y tecnológico que sustenta el desarrollo del sistema propuesto. En primer lugar, se revisan los fundamentos de la generación aumentada por recuperación (RAG), su funcionamiento básico y las limitaciones que pretende mitigar frente al uso aislado de modelos de lenguaje. A continuación, se analizan las principales arquitecturas RAG, las técnicas empleadas para mejorar la recuperación y la selección del contexto, y los métodos utilizados para evaluar la calidad de sus distintos componentes. La revisión se completa con el estudio de los riesgos, las exigencias de privacidad y trazabilidad propias de los entornos educativos, así como con el análisis de antecedentes académicos y técnicos relacionados con la integración de RAG en `Moodle` y otros sistemas de gestión del aprendizaje (*Learning Management System*, LMS). Finalmente, se sintetizan las principales conclusiones del estado del arte y se identifica el hueco que motiva la arquitectura desarrollada en los capítulos posteriores.

2.1. Recuperación aumentada por generación

La generación aumentada por recuperación (RAG) designa una familia de arquitecturas que combinan un modelo de lenguaje preentrenado con una fuente documental externa accesible mediante un mecanismo de recuperación. La respuesta del sistema se produce condicionada por un conjunto de fragmentos seleccionados en el momento de la consulta, en lugar de depender exclusivamente del conocimiento codificado en los parámetros del modelo. La formulación canónica del enfoque corresponde a la propuesta de Lewis et al., 2020 [4], que articula la combinación de un recuperador y un generador en un único marco entrenable de extremo a extremo.

Una manera útil de entender esta arquitectura consiste en contrastarla con el uso aislado de un modelo de lenguaje. Cuando un usuario consulta directamente a un modelo preentrenado, la respuesta procede únicamente de las regularidades estadísticas capturadas en sus parámetros durante el entrenamiento. Esa información, que la literatura denomina memoria paramétrica, presenta limitaciones conocidas: tiene una fecha de corte temporal, no incluye material privado de una organización concreta y no admite revisión

ni inspección directa. RAG introduce una segunda fuente de conocimiento, externa al modelo, en forma de un índice documental consultable. Lewis et al., 2020 [4] describen ese índice como una memoria no paramétrica que puede inspeccionarse y sustituirse sin necesidad de reentrenar el componente generativo, propiedad especialmente útil en dominios cuya base documental se actualiza con frecuencia, como ocurre en los contextos educativos.

El proceso por el que un sistema RAG transforma una consulta del usuario en una respuesta fundamentada se organiza, en su forma básica, en tres fases. La primera, denominada habitualmente *indexing*, prepara la fuente documental: parte de los recursos originales en distintos formatos, los normaliza a texto, los divide en fragmentos documentales de tamaño manejable y los convierte en representaciones vectoriales mediante un modelo de *embeddings* que se almacenan, junto con sus metadatos, en una base vectorial. La segunda fase, el *retrieval*, transforma la consulta del usuario en una representación comparable y recupera los fragmentos documentales más próximos por similitud semántica. La tercera, la *generation*, integra ese contexto recuperado en el *prompt* con el que se invoca al modelo de lenguaje, que produce el texto final apoyándose en él. El estudio sistemático de Gao et al., 2023 [5] presenta esta organización como el patrón *Retrieve-Read* característico del denominado *naïve RAG*, que se asume aquí como referencia conceptual antes de introducir, en los apartados siguientes, las variantes avanzadas y modulares. Manuales recientes orientados a la implementación coinciden en describir el *pipeline* en estas tres fases y aportan vocabulario práctico sobre *loaders*, estrategias de troceado y bases vectoriales [7, 8].

Conviene insistir desde el inicio en que la incorporación de un índice documental no convierte automáticamente al sistema en correcto. La calidad de las respuestas depende del cuidado con el que se ejecuten todas las etapas: una recuperación incompleta, un troceado que rompa la cohesión semántica del documento, unos metadatos pobres o un *prompt* mal construido pueden inducir errores que el componente generativo no será capaz de corregir. RAG permite reducir parcialmente las alucinaciones y facilita la trazabilidad de las respuestas mediante citas a las fuentes recuperadas, pero no las elimina por completo. Esta consideración fija el alcance de las afirmaciones que sobre el enfoque se realizan en el resto de la memoria y motiva la importancia del marco de evaluación que se describe en el capítulo correspondiente.

Una vez establecido el concepto de RAG, conviene examinar con detalle las carencias que motivan su introducción frente al uso aislado de un modelo de lenguaje. Lewis et al., 2020 [4] identifican tres dificultades estructurales de los modelos puramente paramétricos: la imposibilidad de expandir o revisar de forma directa la información almacenada

en sus pesos, la opacidad respecto a la procedencia del conocimiento utilizado y la propensión a generar afirmaciones plausibles pero incorrectas, habitualmente denominadas alucinaciones. El estudio sistemático de Gao et al., 2023 [5] amplía la enumeración añadiendo el desfase temporal inherente a la fecha de corte del corpus de entrenamiento y la dificultad para incorporar dominios específicos sin recurrir a procesos costosos. A estas carencias se añade una observación práctica frecuente: un sistema generativo convencional no dispone de un mecanismo natural para reconocer que no posee información suficiente, y puede responder con apariencia de seguridad incluso en ausencia de evidencia [7].

La ventana de contexto del modelo fija un tope al número de *tokens* que pueden procesarse en una sola consulta y, aunque las versiones más recientes alcanzan tamaños considerables, introducir el contenido íntegro de una asignatura en el *prompt* resulta ineficiente y deteriora la atención del modelo en las posiciones intermedias del texto [7]. Gao et al., 2023 [5] recuerdan que disponer de ventanas largas no exime de la recuperación: una generación basada únicamente en contexto extenso se comporta como una caja negra cuya auditoría resulta difícil, mientras que la recuperación y filtrado de fragmentos documentales relevantes permiten localizar las unidades documentales concretas que sustentan cada respuesta.

Estas carencias adquieren un peso particular en el dominio educativo. Los materiales docentes se actualizan a lo largo del periodo lectivo, los recursos suelen ser visibles o no según el rol del usuario y el momento del curso, y el acceso queda mediado por las estructuras propias de los sistemas de gestión del aprendizaje, organizadas habitualmente en cursos, secciones, actividades y agrupamientos. Un modelo de lenguaje que no haya visto los documentos de la asignatura puede producir respuestas razonables en términos generales pero ajenas al programa concreto, o incluso contrarias a la formulación empleada en apuntes y guías docentes. Un asistente generalista, por construcción, desconoce esta organización y no puede respetar la frontera entre lo accesible para el alumnado y lo restringido por permisos.

Las dos alternativas que habitualmente se contraponen a la recuperación no resuelven el problema por sí solas. El ajuste fino del modelo, o *fine-tuning*, resulta menos adecuado cuando los documentos cambian con frecuencia: cada nueva versión de un apunte o de un enunciado de prácticas obligaría a un ciclo de entrenamiento adicional, con costes y riesgos difíciles de asumir en un calendario docente. Gao et al., 2023 [5] observan, en línea con esta lectura, que la incorporación de conocimiento factual nuevo por la vía del entrenamiento supervisado no estructurado presenta dificultades reproducibles, mientras que la recuperación logra un comportamiento más estable ante conocimiento desactualizado o privado. La alternativa basada en una ventana de contexto muy amplia comparte

la limitación ya señalada de opacidad y coste, y no exime de seleccionar y citar las fuentes pertinentes.

El conjunto de carencias descritas justifica la dirección general adoptada por los enfoques RAG: incorporar una memoria documental externa, inspeccionable, sustituible cuando los recursos cambian y consultable mediante un mecanismo de recuperación capaz de respetar la organización del corpus de origen. La generación queda condicionada por la evidencia documental recuperada, las respuestas pueden acompañarse de citas al material original y el sistema debe contemplar de forma explícita el caso en el que no se localiza evidencia suficiente, en cuyo caso resulta preferible que el componente generativo se abstenga o declare la insuficiencia antes que producir una respuesta no fundamentada. Las técnicas concretas que permiten implementar esta dirección se describen en los apartados siguientes del estado del arte.

2.1.1. *Pipeline* básico: ingesta, indexación, recuperación y generación

Una vez establecidos el concepto general y la motivación de RAG, procede detallar cómo se organiza operativamente un sistema de este tipo. La formulación más extendida lo describe como un *pipeline* con dos momentos diferenciados: una fase de indexación, ejecutada antes de atender consultas, y una fase de respuesta a cada pregunta del usuario [5]. Cada momento responde a restricciones operativas distintas: la indexación admite procesamiento por lotes y revisiones periódicas, mientras que la fase de consulta debe ofrecer una latencia razonable y mantener trazabilidad sobre la información empleada.

La fase de ingesta e indexación parte de la obtención de los documentos de la base de conocimiento, que en el ámbito educativo suelen corresponder a materiales docentes publicados en plataformas de gestión del aprendizaje. Estos materiales llegan en formatos heterogéneos y se normalizan a una representación textual común, lo que permite tratarlos de forma uniforme con independencia de su origen. El texto resultante se divide en fragmentos documentales de tamaño acotado mediante un proceso de fragmentación documental (*chunking*), operación que condiciona la calidad de fases posteriores. Cada fragmento documental se convierte en una representación vectorial mediante un modelo de *embeddings* y se almacena, junto con sus metadatos, en una base vectorial [7, 8].

El detalle sobre representaciones densas, la selección de modelos de *embeddings* y las propiedades de las bases vectoriales se aborda en la subsección siguiente y en el capítulo dedicado al *pipeline* de implementación.

La fase de consulta comienza con la recepción de la pregunta del usuario, que

se transforma en una representación comparable con la del índice. Sobre ella se ejecuta una búsqueda de fragmentos documentales próximos, de modo que el sistema obtenga un conjunto limitado de evidencias pertinentes para la pregunta planteada. Este patrón, recogido en la literatura como *Retrieve-Read*, constituye el flujo más sencillo y sirve de base para variantes más elaboradas que se introducen en secciones posteriores [5]. En dominios con corpus estructurados, los metadatos permiten aplicar filtros derivados del contexto y de los permisos del usuario, así como de la organización documental, de manera que los fragmentos recuperados sean a la vez relevantes y autorizados [9].

La fase de generación toma el contexto recuperado y construye con él la entrada del modelo de lenguaje, que produce el texto final apoyándose en las evidencias documentales aportadas en lugar de depender solo de su memoria interna [4]. Junto a la respuesta, el sistema puede devolver referencias a los fragmentos empleados, lo que habilita la trazabilidad y facilita la verificación por parte del usuario. La relación entre la memoria interna del modelo y la información externa recuperada se examina con mayor detalle más adelante en este mismo capítulo, y las técnicas concretas de construcción del *prompt* se abordan en el capítulo sobre generación e inyección de contexto.

Este *pipeline* constituye la base común sobre la que se construye la mayoría de los sistemas RAG descritos en la literatura. A partir de esta estructura mínima se introducen variantes orientadas a mejorar la calidad de la indexación, la precisión de la recuperación, la selección del contexto y la fidelidad de la generación, que se examinan en los apartados posteriores del estado del arte.

Cabe insistir en que este *pipeline* se comporta como una cadena cuya calidad final depende de la coordinación entre fases: el fallo de cualquiera de ellas, ingesta, troceado, recuperación o generación, puede degradar la respuesta aun cuando las demás funcionen correctamente. Esta dependencia mutua justifica separar, en el marco de evaluación, las métricas de recuperación de las métricas de generación, distinción que se retoma en el capítulo dedicado a evaluación experimental.

La Figura 2.1 resume este esquema conceptual general, con la fase de indexación ejecutada fuera de línea y la fase de consulta en línea conectadas a través del índice; no representa la arquitectura concreta del prototipo, que se aborda en el Capítulo 4.

2.1.2. *Embeddings*, vector *stores* y búsqueda semántica

La fase de recuperación introducida en el apartado anterior se apoya en una representación numérica del texto que haga comparables la consulta y los fragmentos documentales. Esa representación se obtiene mediante modelos de *embeddings*, que transforman

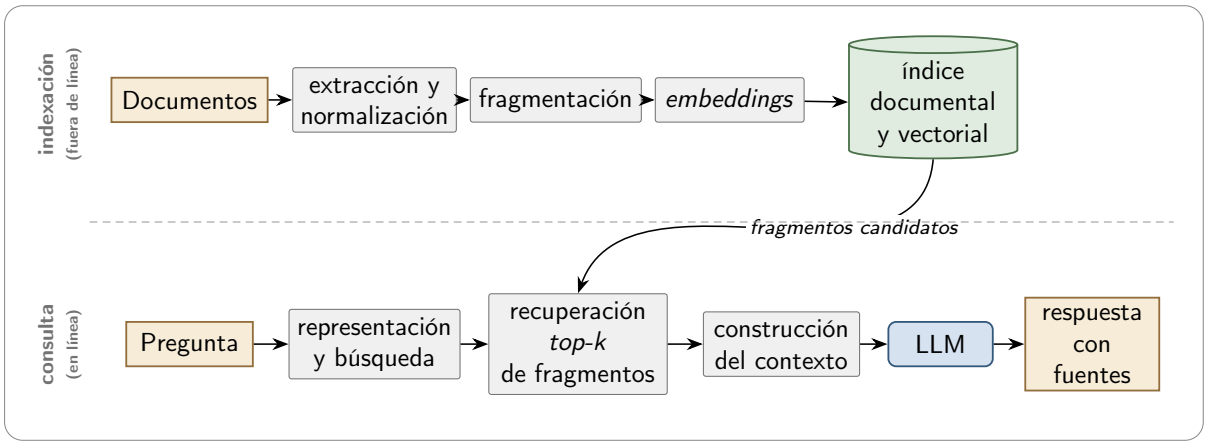

Figura 2.1: Funcionamiento conceptual de un sistema RAG: indexación fuera de línea (documentos, fragmentación y *embeddings*) y consulta en línea (recuperación, construcción del contexto y generación con fuentes). Esquema general, independiente de la implementación del prototipo. Fuente: elaboración propia.

secuencias de *tokens* en vectores de dimensión fija pensados para capturar relaciones semánticas: textos cercanos en significado tienden a ocupar posiciones próximas en el espacio vectorial, mientras que textos no relacionados quedan alejados. Sobre esta propiedad se construye la búsqueda por similitud, que constituye la base operativa de la recuperación semántica en sistemas RAG [7].

La formulación más representativa de esta línea es la recuperación densa propuesta por Karpukhin et al., 2020 [10] en el marco de *Dense Passage Retrieval*. El enfoque parte de dos codificadores entrenados de forma conjunta, uno dedicado a la pregunta y otro a los fragmentos documentales candidatos, ambos derivados de un modelo de lenguaje contextual del tipo BERT [11]. La consulta y los fragmentos se proyectan al mismo espacio vectorial y la relevancia se aproxima mediante el producto escalar entre sus representaciones, lo que permite ordenar los candidatos por afinidad semántica. La diferencia frente a los métodos léxicos clásicos, como TF-IDF o BM25 [12], radica en que la recuperación densa puede emparejar formulaciones distintas que comparten significado aun cuando no coincidan las palabras concretas [10], limitación característica de los enfoques basados en estadísticas de términos; estos métodos léxicos y sus combinaciones con recuperación densa se examinan en apartados posteriores.

Para que la búsqueda resulte eficiente sobre colecciones voluminosas, los *embeddings* se almacenan en un componente especializado denominado base vectorial o *vector store*, que mantiene de forma conjunta cada vector, el fragmento textual del que procede y un conjunto de metadatos asociados. Estos sistemas proporcionan estructuras de índice y algoritmos de búsqueda aproximada que recuperan los vectores más próximos a la representación de la consulta y devuelven, junto a la puntuación de similitud, los fragmentos y

metadatos correspondientes [8]. La literatura práctica señala además que el procesamiento por lotes, la persistencia, el filtrado por metadatos y la actualización incremental del índice son aspectos relevantes en el diseño de este componente, aunque su tratamiento detallado excede el alcance de esta revisión.

La articulación entre modelo de *embeddings*, base vectorial y modelo de lenguaje condiciona la calidad final del sistema. Una elección desafortunada de modelo de *embeddings* puede generar representaciones poco discriminativas para el dominio de interés y, en consecuencia, una recuperación que entregue al generador un contexto irrelevante o incompleto. Gao et al., 2023 [5] recuerdan, en su revisión sistemática, que no existe un modelo de *embeddings* óptimo de forma universal y que su rendimiento depende del corpus, de la longitud típica de los fragmentos y de la naturaleza de las consultas, por lo que la decisión debe sustentarse en evidencia empírica.

La búsqueda semántica basada en *embeddings* constituye, en síntesis, una mejora sustantiva sobre la coincidencia léxica estricta cuando consultas y documentos emplean formulaciones distintas para referirse a los mismos conceptos. No agota, sin embargo, el espacio de soluciones: la recuperación híbrida que combina señales densas y léxicas, el reordenamiento posterior de candidatos y las estrategias de selección y compresión del contexto recuperado se abordan en los apartados siguientes del estado del arte.

2.1.3. Memoria paramétrica y memoria no paramétrica

Una de las contribuciones conceptuales del trabajo de Lewis et al., 2020 [4] consiste en distinguir, dentro de un mismo sistema generativo, dos formas complementarias de almacenamiento de conocimiento. La primera, denominada memoria paramétrica, corresponde a la información que el modelo de lenguaje preentrenado retiene de forma implícita en sus pesos como resultado del entrenamiento sobre grandes corpus. Esta memoria sostiene la capacidad lingüística y generativa del modelo y permite generalizar patrones más allá de los ejemplos vistos durante el ajuste, pero presenta limitaciones conocidas: resulta difícil de inspeccionar, costosa de actualizar y, sobre todo, opaca a la hora de atribuir una afirmación concreta a una fuente verificable.

La segunda forma, denominada memoria no paramétrica, se materializa fuera del modelo en forma de un índice documental consultable en tiempo de inferencia. A diferencia de los parámetros, ese índice puede revisarse, ampliarse o sustituirse sin necesidad de un nuevo entrenamiento, lo que habilita la incorporación de conocimiento actualizado o específico de un dominio y facilita un grado mayor de trazabilidad, dado que los fragmentos documentales utilizados pueden mostrarse o citarse junto a la respuesta. Lewis et al.,

2020 [4] ilustran esta propiedad mediante el reemplazo controlado del índice sin alterar el componente generativo, lo que evidencia la independencia entre ambas memorias.

La articulación de las dos memorias define el rasgo distintivo de RAG. El componente generativo aporta cohesión sintáctica y capacidad de síntesis, mientras que el componente no paramétrico introduce evidencias documentales pertinentes para la consulta concreta. Esta combinación proporciona una vía más flexible que el ajuste fino para incorporar conocimiento cambiante o privado, observación que Gao et al., 2023 recogen en su revisión sistemática al comparar ambos enfoques [5]. La calidad del resultado depende, por tanto, no solo de las capacidades de cada componente, sino también de la coordinación entre la búsqueda de evidencias externas y la generación condicionada por ellas.

Conviene matizar que la presencia de una memoria no paramétrica no garantiza por sí sola la corrección de las respuestas. Si la recuperación selecciona evidencias poco pertinentes o incompletas, el componente generativo puede producir afirmaciones erróneas con apariencia de fundamento documental. Las ventajas y limitaciones derivadas de esta combinación se sintetizan en el apartado siguiente.

2.1.4. Ventajas y limitaciones de RAG

La revisión de los apartados anteriores permite cerrar la sección con un balance sintético de las ventajas y las limitaciones asociadas al paradigma RAG. En el lado de las ventajas, el recurso a una memoria documental externa habilita la actualización del conocimiento sin necesidad de reentrenar el modelo generativo, lo que facilita la incorporación de información reciente o específica de un dominio cerrado. La disponibilidad de fragmentos documentales recuperables sustenta cierto grado de trazabilidad, dado que cada respuesta puede acompañarse de referencias verificables, y contribuye a una reducción parcial, aunque no a la eliminación, de las alucinaciones [4]. Al mismo tiempo, la separación funcional entre recuperación y generación aporta una flexibilidad operativa de la que carecen los enfoques basados exclusivamente en ajuste fino, especialmente cuando los materiales cambian con frecuencia [5].

En el lado de las limitaciones, la calidad del sistema depende directamente de la cobertura, la actualización y la coherencia del corpus disponible, así como de la idoneidad de la fragmentación documental elegida. Una recuperación poco precisa puede entregar evidencias irrelevantes, incompletas o descontextualizadas, y un modelo de *embeddings* mal ajustado al dominio puede producir contextos inadecuados aunque el resto de los componentes funcionen correctamente. A estas dificultades se añade el riesgo de que el componente generativo ignore, mezcle o extrapole más allá del material recuperado, pro-

duciendo respuestas plausibles pero no fundamentadas [7]. De ahí que recuperación y generación deban evaluarse con métricas propias, según se desarrolla en el capítulo de evaluación experimental.

De esta forma, RAG aporta actualización, trazabilidad y reducción parcial de alucinaciones, pero desplaza buena parte del problema a la calidad del corpus, la fragmentación documental, la recuperación, la selección del contexto y la fidelidad de la generación. Las técnicas con las que la literatura aborda estos compromisos se examinan en el apartado siguiente, dedicado a las arquitecturas *naïve*, *advanced* y modular de RAG.

2.2. Arquitecturas RAG: *naïve*, *advanced* y modular RAG

Las arquitecturas RAG han evolucionado desde esquemas lineales sencillos hasta configuraciones formadas por componentes especializados y flujos de procesamiento adaptables. Esta evolución responde a la necesidad de mejorar la calidad de la recuperación, reducir el ruido documental y proporcionar al componente generativo un contexto más pertinente. La literatura distingue habitualmente tres paradigmas principales, *Naïve* RAG, *Advanced* RAG y Modular RAG, que representan diferentes niveles de complejidad y flexibilidad arquitectónica. En esta sección se describen sus características fundamentales, sus principales ventajas y limitaciones, y la relación progresiva que existe entre ellos.

2.2.1. *Naïve* RAG

La revisión sistemática de Gao et al., 2023 [5] agrupa bajo la denominación *Naïve* RAG el paradigma arquitectónico inicial sobre el que se han construido las variantes posteriores. Esta arquitectura formaliza, en su forma más simple, la combinación de un recuperador y un generador originalmente introducida por Lewis et al. 2020 [4], organizándola como una cadena lineal y secuencial en la que cada etapa entrega su salida directamente a la siguiente, sin pasos intermedios de intervención.

La cadena se articula en tres etapas: una indexación previa que prepara la base documental, una recuperación top-k que selecciona fragmentos documentales para la consulta del usuario y una generación condicionada que produce la respuesta a partir del contexto recuperado. Gao et al., 2023 [5] recogen este flujo bajo el patrón *Retrieve-Read*, denominación que enfatiza el carácter de un único paso de recuperación seguido de una

sola pasada generativa, sin reformulación de la consulta ni reordenación posterior de los candidatos.

La principal ventaja de *Naïve* RAG reside en su sencillez. Al carecer de componentes especializados intermedios, resulta directamente implementable a partir de las piezas ya descritas en la sección anterior y permite establecer una línea base conceptual y experimental sobre la que comparar variantes más elaboradas. Esa mínima superficie de decisiones facilita la trazabilidad del comportamiento del sistema y constituye un punto de partida razonable en la mayoría de las propuestas posteriores.

La misma simplicidad explica, sin embargo, sus límites. El comportamiento de *Naïve* RAG depende fuertemente de la calidad de la fragmentación documental y de la idoneidad del recuperador empleado: si la primera produce unidades poco coherentes o el segundo entrega candidatos solo parcialmente pertinentes, el componente generativo hereda directamente ese ruido y carece de mecanismos para corregirlo. Tampoco existen, en este esquema, pasos de reformulación de la consulta, búsqueda híbrida o reordenación de candidatos que mitiguen los errores de recuperación. Gao et al., 2023 [5] señalan que estas carencias motivan la aparición de variantes denominadas *Advanced* RAG, que se examinan en el apartado siguiente.

La Figura 2.2 sintetiza este esquema secuencial de recuperación y lectura.

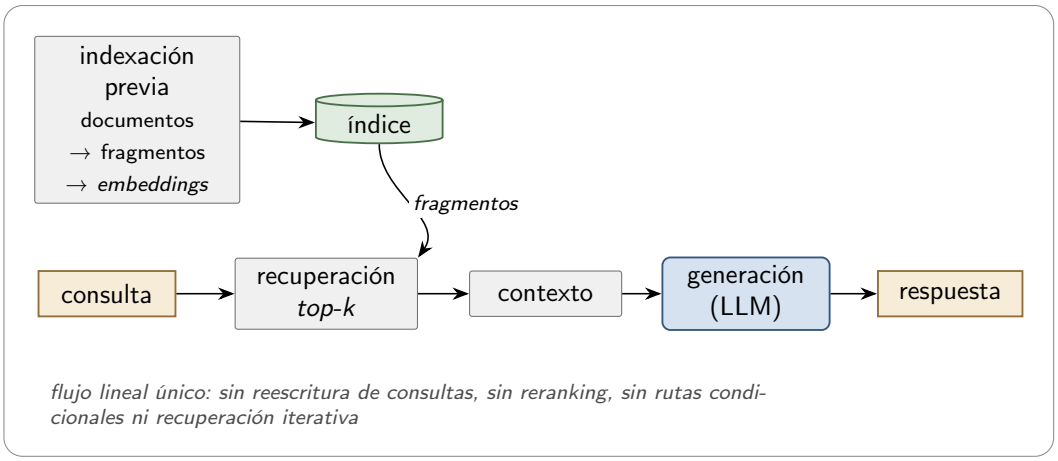

Figura 2.2: *Naïve* RAG: flujo lineal *Retrieve–Read*, sin reescritura de consultas, *reranking* ni recuperación iterativa. Fuente: elaboración propia.

2.2.2. *Advanced* RAG

A partir del esquema secuencial descrito en el apartado anterior, la literatura denomina *Advanced* RAG al conjunto de variantes arquitectónicas que introducen mecanismos adicionales sin alterar la estructura de fondo del *pipeline*. La revisión sistemática de Gao

et al., 2023 [5] organiza esos mecanismos en dos grupos según el momento en que actúan: aquellos que intervienen antes de la recuperación, agrupados como *pre-retrieval*, y aquellos que operan sobre los candidatos ya recuperados, agrupados como *post-retrieval*. Esta dicotomía permite incorporar mejoras de forma gradual, manteniendo el resto del *pipeline* como referencia conocida.

Las técnicas de *pre-retrieval* persiguen mejorar la calidad de la información que llegará a la fase de generación al actuar sobre el índice y sobre la consulta. Entre ellas figuran la optimización de la estructura del índice y de la granularidad del fragmento documental, el enriquecimiento con metadatos descriptivos y la reformulación o expansión de la consulta para acercarla a la formulación habitual en los documentos. Estas técnicas se desarrollan con detalle en el apartado dedicado a *pre-retrieval* y aquí basta con señalar que constituyen la línea de actuación previa al motor de recuperación propiamente dicho.

Las técnicas de *post-retrieval*, en cambio, actúan sobre el conjunto de candidatos devuelto por la recuperación inicial. Su finalidad es filtrar, reordenar o comprimir esos candidatos antes de entregar el contexto al componente generativo, con el objetivo de reducir redundancia, descartar fragmentos poco pertinentes y respetar las restricciones de la ventana de contexto. El *re-ranking*, la selección selectiva del contexto y las estrategias de compresión son ejemplos característicos de esta línea y se examinan en el apartado posterior dedicado a *post-retrieval*.

Desde una perspectiva práctica, los manuales orientados a la implementación reflejan este planteamiento al describir el *pipeline* RAG como una composición de componentes intercambiables, cada uno de los cuales admite mejoras locales sin necesidad de rediseñar el sistema en su conjunto [8]. Esa modularidad operativa es la que permite, en la práctica, adoptar mejoras *Advanced* RAG de forma incremental sobre una base *Naïve* RAG ya funcional.

El propósito común de estas mejoras es reducir el ruido en el contexto, aumentar la pertinencia de las evidencias documentales entregadas al componente generativo y, en última instancia, sostener una generación más fiel a las fuentes recuperadas. Conviene matizar, no obstante, que cada componente añadido incrementa la complejidad del sistema y los posibles puntos de fallo: la incorporación de mecanismos avanzados solo resulta defendible cuando puede evaluarse su efecto sobre la calidad final, distinción que motiva el marco de evaluación discutido más adelante [5]. Las arquitecturas Modular RAG, examinadas en el apartado siguiente, generalizan esta lógica de mejora por componentes hacia esquemas más flexibles y componibles.

La Figura 2.3 muestra cómo estas mejoras *pre-retrieval* y *post-retrieval* se incorporan alrededor del flujo base sin sustituirlo; se trata de una representación del estado

del arte, no de la configuración final del prototipo.

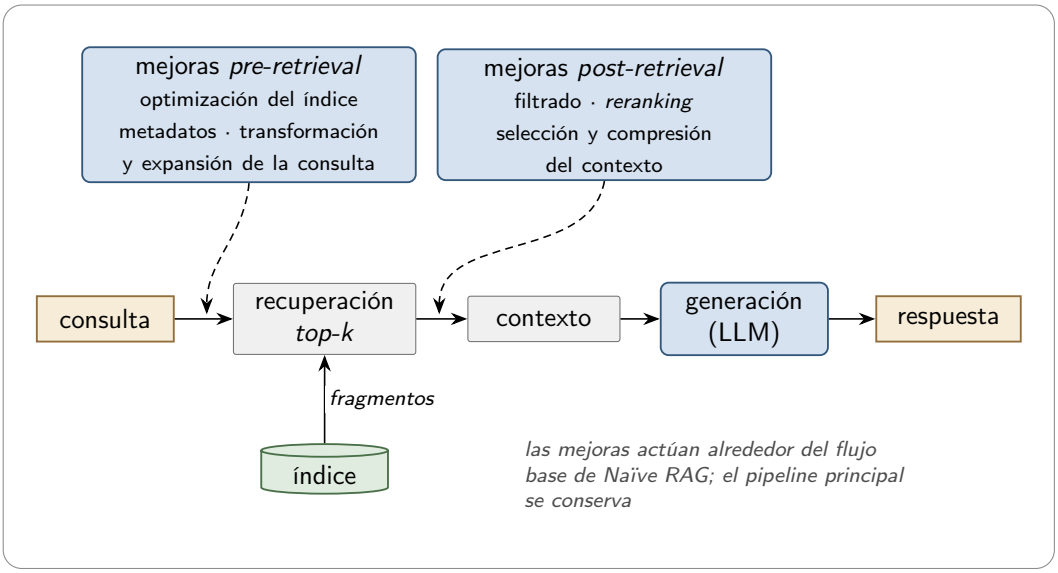

Figura 2.3: *Advanced* RAG: mejoras *pre-retrieval* (optimización del índice, metadatos, transformación de la consulta) y *post-retrieval* (filtrado, *reranking*, compresión del contexto) alrededor del flujo base. Figura de estado del arte; no describe la configuración del prototipo. Fuente: elaboración propia.

2.2.3. Modular RAG

La distinción introducida en el apartado anterior entre técnicas *pre-retrieval* y *post-retrieval* recoge solo parte de la diversidad arquitectónica descrita en la literatura. Gao et al., 2023 [5] denominan Modular RAG al paradigma que generaliza esa lógica de mejora por componentes hasta concebir el sistema como un conjunto de módulos especializados que pueden combinarse y sustituirse de forma relativamente independiente. A diferencia de *Advanced* RAG, que refuerza puntos concretos de un *pipeline* secuencial, Modular RAG ofrece un marco para componer arquitecturas en las que el orden y las relaciones entre etapas son ellos mismos objeto de diseño.

El catálogo de módulos descrito en la literatura incluye, entre otros, búsqueda, enrutamiento, reformulación de la consulta, recuperación propiamente dicha, reordenación, lectura o generación, memoria y fusión de resultados procedentes de varias fuentes [5, 13]. Sobre ese conjunto se construyen patrones más flexibles que la cadena lineal *Retrieve-Read*: recuperación iterativa, en la que el sistema puede volver al índice tras una primera generación; recuperación recursiva, que descompone consultas complejas en subconsultas; recuperación adaptativa, que ajusta dinámicamente la estrategia de búsqueda; y enrutamiento, que dirige cada consulta hacia el módulo o la fuente más adecuada.

Desde una perspectiva de implementación, los manuales prácticos describen el *pipeline* RAG como una composición de *loaders*, fragmentadores, modelos de *embeddings*, bases vectoriales, recuperadores, plantillas de *prompt*, cadenas y, en su evolución más reciente, agentes capaces de orquestar herramientas y decisiones de recuperación [8]. Esa modularidad operativa facilita incorporar mejoras de forma incremental y realizar estudios de ablación que aíslen el efecto de cada componente sobre la calidad final.

La principal ventaja de este enfoque es la flexibilidad: el sistema puede adaptarse a dominios documentales heterogéneos o a varias fuentes sin rediseñar el *pipeline* completo. La contrapartida, sin embargo, no es menor. Cada módulo introducido añade complejidad, puntos de fallo, coste computacional y carga de mantenimiento, y exige procedimientos de evaluación que combinen métricas locales y globales para distinguir mejoras reales de cambios sin efecto demostrable.

La lectura modular permite cerrar la revisión arquitectónica: RAG deja de entenderse como una cadena única y pasa a verse como un espacio de diseño compuesto por módulos seleccionables. A partir de esta base, el apartado siguiente no vuelve a clasificar arquitecturas, sino que examina las técnicas concretas con las que la literatura interviene sobre la recuperación y sobre la selección del contexto. La elección de cada una de ellas debe sustentarse en su aportación a la calidad del sistema y en su coste, antes que en una acumulación acrítica de componentes.

La Figura 2.4 representa este paradigma modular, con módulos intercambiables coordinados mediante enrutamiento y flujos no estrictamente lineales.

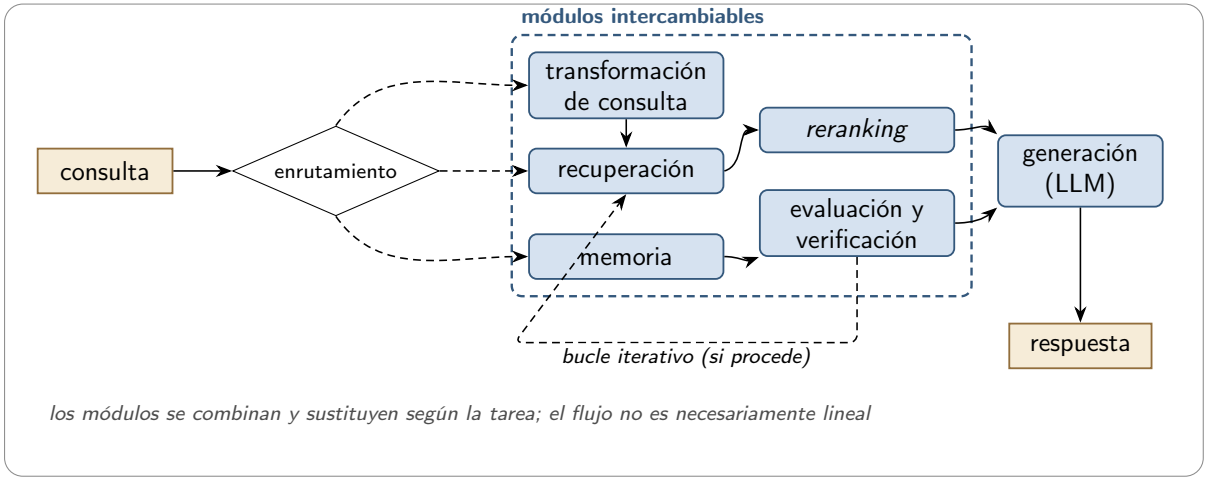

Figura 2.4: Modular RAG: módulos intercambiables (transformación de consulta, recuperación, memoria, *reranking*, evaluación) coordinados mediante enrutamiento, con posibles bucles iterativos. Representación general del paradigma. Fuente: elaboración propia.

2.3. Técnicas de recuperación y mejora del contexto

La eficacia de un sistema RAG depende en gran medida de las técnicas empleadas para preparar el corpus, formular la consulta, recuperar las evidencias y seleccionar el contexto que recibe el componente generativo. Estas técnicas pueden intervenir antes de la recuperación, mediante la optimización de la indexación, la fragmentación y los metadatos, durante la búsqueda, a través de estrategias densas, léxicas o híbridas, y después de ella, mediante mecanismos de reordenación, selección y compresión. Esta sección examina dichas líneas de actuación, junto con las técnicas de transformación de consultas y las variantes arquitectónicas que amplían el proceso de recuperación mediante estructuras jerárquicas, grafos o componentes agénticos. El objetivo es identificar las posibilidades y limitaciones de cada enfoque, así como los criterios que deben orientar su incorporación a un sistema RAG concreto.

2.3.1. Optimización *pre-retrieval*: indexación, granularidad y metadatos

Las técnicas agrupadas bajo el término *pre-retrieval* engloban las intervenciones aplicadas antes de consultar el índice documental, con el propósito de mejorar la información disponible para la recuperación posterior. Gao et al. 2023 [5] recogen en su revisión sistemática esta línea como una de las dos vías características de los enfoques *Advanced* RAG y la describen como una optimización conjunta del índice y de la consulta. La idea de fondo es que la calidad de la recuperación no depende únicamente del modelo de búsqueda elegido: si los documentos están mal preparados, mal fragmentados o descritos con metadatos pobres, la recuperación posterior arrastrará ruido y omisiones aunque el modelo de *embeddings* sea adecuado.

La primera línea de actuación es la indexación, entendida como fase de preparación documental previa a la consulta. En la formulación habitual de los manuales orientados a la implementación, esa fase incluye la obtención de los materiales en sus formatos originales, la extracción y normalización del texto, la generación de representaciones adecuadas para la recuperación y su almacenamiento en una estructura consultable [7, 8]. Esta fase suele ejecutarse fuera de línea, lo que permite revisar y modificar el corpus sin afectar al tiempo de respuesta de las consultas posteriores y, al mismo tiempo, deja margen para aplicar transformaciones costosas que no serían viables durante la interacción.

La granularidad de la unidad documental indexada constituye el segundo eje del *pre-retrieval*. La fragmentación documental (*chunking*) consiste en dividir cada documento

original en unidades más manejables sobre las que opera el modelo de búsqueda. La decisión es delicada: fragmentos demasiado pequeños pueden perder el contexto que da sentido a una afirmación, mientras que fragmentos demasiado grandes tienden a diluir la relevancia al mezclar varias ideas y a sobrecargar la posterior selección de evidencias. La literatura práctica subraya que esta decisión no se reduce a fijar una longitud uniforme, sino que conviene aprovechar la estructura del material, secciones, encabezados, páginas o agrupaciones temáticas, para preservar la coherencia interna [8].

Junto a la fragmentación, los metadatos asociados a cada fragmento documental amplían la información disponible para la recuperación más allá del contenido textual. Permiten filtrar por fuente, tipo documental, sección, página, fecha o ámbito de validez, contextualizar los resultados respecto al perfil del usuario, ordenar los candidatos por criterios no puramente semánticos y, en particular, conservar la procedencia que sostiene la trazabilidad de la respuesta [7, 8]. En entornos documentales estructurados o sujetos a control de acceso, los metadatos resultan especialmente útiles porque combinan recuperación semántica con restricciones explícitas sobre qué material puede entregarse a una consulta concreta.

Las intervenciones *pre-retrieval* no generan, en sí mismas, ninguna respuesta: lo que hacen es condicionar qué evidencias podrán recuperarse y con qué precisión, así como qué información de procedencia las acompañará. Una vez preparado el índice, el problema inmediato es cómo formular la búsqueda sobre él.

2.3.2. Recuperación densa, léxica e híbrida

Una vez construido el índice descrito en el apartado anterior, la recuperación de fragmentos puede abordarse, en términos generales, mediante tres familias de estrategias: una basada en la similitud semántica entre representaciones densas, otra basada en la coincidencia ponderada de términos entre consulta y documento y una tercera que combina ambas señales. La revisión sistemática de Gao et al., 2023 [5] recoge esta clasificación dentro de la línea *Advanced* RAG y la presenta como un espacio de decisión, no como una opción única que convenga a todos los corpus o consultas.

La recuperación densa toma como referencia técnica la formulación de *Dense Passage Retrieval* propuesta por Karpukhin et al., 2020 [10] en la que dos codificadores entrenados de forma conjunta proyectan, respectivamente, la pregunta y los fragmentos documentales candidatos a un mismo espacio vectorial. La relevancia se aproxima mediante el producto escalar entre las representaciones resultantes, lo que permite ordenar los candidatos por afinidad semántica con independencia de las palabras concretas em-

pleadas. Su ventaja característica es la capacidad para emparejar formulaciones distintas que comparten significado: dos textos que abordan la misma idea con vocabularios diferentes pueden quedar próximos en el espacio vectorial y, por tanto, ser recuperados conjuntamente sin necesidad de coincidencia literal.

La recuperación léxica responde a una lógica complementaria. Su referencia clásica es la función BM25 introducida en el sistema Okapi por Robertson et al., 1995 [14] que estima la relevancia a partir de la frecuencia de los términos de la consulta en cada documento, modulada por la rareza de cada término dentro de la colección y normalizada respecto a la longitud documental. Esta combinación, sostenida en el modelo probabilístico de recuperación, otorga mayor peso a los términos poco frecuentes y evita favorecer sistemáticamente a los documentos más extensos. Su utilidad práctica se manifiesta sobre todo cuando la pregunta contiene elementos cuya coincidencia exacta resulta decisiva, siglas, nombres propios, identificadores, fórmulas, fechas o vocabulario técnico, porque la búsqueda semántica densa puede no priorizar coincidencias literales que para el usuario son determinantes.

Los enfoques híbridos se justifican precisamente porque las dos familias anteriores fallan en escenarios complementarios. La recuperación densa puede omitir documentos en los que un término exacto resulta determinante; la recuperación léxica puede pasar por alto reformulaciones cuya coincidencia léxica es baja pese a una fuerte equivalencia semántica. La combinación de ambas señales, mediante esquemas de fusión o ponderación, permite atenuar estas limitaciones, según recoge la literatura RAG en su examen de estrategias de recuperación [5]. La utilidad concreta de la combinación no se sigue, sin embargo, de su mera adopción. El balance entre señales semánticas y léxicas, el método de fusión y los parámetros de ponderación dependen del corpus, del tipo de consultas esperadas y de los resultados que arroje su evaluación empírica. Aunque el patrón híbrido se ha consolidado en sistemas RAG recientes, su adopción acrítica no garantiza mejoras: la combinación puede introducir ruido si una de las señales domina indebidamente o si la ponderación no se ajusta al perfil de las consultas observadas.

La elección entre recuperación densa, léxica o híbrida no debe plantearse como una preferencia universal, sino como una decisión empírica condicionada por la naturaleza del corpus y por las características de las consultas previstas. A esa decisión se suma el problema, no menos relevante, de cómo se formula la búsqueda sobre el índice: una misma necesidad de información puede admitir distintas representaciones de consulta que afectan a los resultados con independencia del modelo de recuperación elegido.

2.3.3. Transformación de consultas: *query rewriting*, *multi-query* y HyDE

La calidad de la recuperación no depende únicamente del índice y del modelo de búsqueda elegidos: también está mediada por la propia formulación de la consulta. Una misma necesidad de información puede expresarse con un vocabulario distinto al empleado en los documentos, de forma incompleta, ambigua o demasiado breve para que el recuperador encuentre evidencias pertinentes. La revisión sistemática de Gao et al., 2023 [5] sitúa en este punto una familia específica de técnicas, agrupadas bajo el término transformación de consultas, cuyo propósito común es intervenir sobre la consulta antes de ejecutar la búsqueda o utilizarla como base para generar consultas alternativas.

Dentro de esta familia, la reescritura de la consulta busca reformular el enunciado original para acercarlo a la formulación esperable en el corpus, sustituyendo expresiones coloquiales o ambiguas por términos más alineados con el material indexado. La expansión añade términos relacionados o información de contexto que enriquece la consulta sin sustituirla, mientras que las estrategias *multi-query* producen varias formulaciones alternativas a partir de una misma pregunta y combinan los resultados obtenidos por cada una. Los manuales orientados a la implementación describen estas variantes como módulos que pueden acoplarse al recuperador sin alterar el resto del *pipeline* [8], lo que permite incorporarlas de forma incremental.

Un caso particular que conviene examinar por su carácter representativo es *Hypothetical Document Embeddings* (HyDE), propuesto por Gao et al., 2022 [15] en un trabajo específico sobre recuperación densa *zero-shot*. La técnica consiste en pasar primero la consulta original por un modelo de lenguaje instruccional para generar un documento hipotético, codificar después ese documento mediante un *encoder* denso y utilizar la representación vectorial resultante como punto de búsqueda en el índice documental. El documento hipotético no se trata como evidencia final, sino como una representación intermedia que aproxima el tipo de texto en el que la respuesta podría aparecer; los documentos efectivamente entregados al componente generativo son los reales, recuperados a partir de la vecindad de ese vector.

Estas intervenciones sobre la consulta admiten ganancias plausibles en cobertura y en pertinencia, pero comportan riesgos que deben tenerse en cuenta. La reformulación o la expansión pueden introducir deriva semántica si modifican el sentido de la pregunta original; las estrategias *multi-query* pueden recuperar evidencias parcialmente contradictorias entre formulaciones; y, en el caso particular de HyDE, el documento hipotético generado puede contener información no fundamentada, riesgo que el propio trabajo identifica como

límite inherente al método [15]. Por estas razones, la incorporación de técnicas de transformación de consultas en un sistema RAG no debe asumirse como mejora garantizada y exige evaluación empírica sobre el corpus y las consultas concretas a las que se aplica.

Las intervenciones aquí descritas actúan antes de que el componente generativo reciba contexto y modifican lo que se introduce en la búsqueda, no lo que se entrega al modelo de lenguaje. Una vez obtenidos los candidatos, persiste el problema de decidir cuáles incluir, en qué orden y con qué nivel de compresión, decisión que se sitúa en una fase distinta del *pipeline*.

2.3.4. Optimización *post-retrieval*: *re-ranking*, selección y compresión de contexto

La búsqueda sobre el índice devuelve un conjunto de candidatos recuperados, pero no todos ellos resultan igualmente útiles para construir el contexto que recibirá el componente generativo. La lista inicial puede contener ruido, fragmentos parcialmente redundantes, evidencias solo tangencialmente relevantes o unidades demasiado extensas para la ventana de contexto del modelo. La revisión sistemática de Gao et al., 2023 [5] agrupa bajo el término *post-retrieval* las técnicas que actúan sobre ese conjunto antes de entregarlo al generador, con el propósito común de mejorar la calidad del contexto efectivamente empleado en la generación.

Una primera familia consiste en seleccionar y filtrar los candidatos recuperados. La selección *top-k* fija el número máximo de fragmentos que entrarán en el contexto, mientras que los filtros adicionales descartan candidatos por debajo de umbrales de relevancia, eliminan duplicados o fragmentos solapados y, cuando los metadatos lo permiten, priorizan las evidencias más específicas frente a las más genéricas. La literatura práctica subraya que estas decisiones no son neutras: el ajuste de *top-k* y de los umbrales condiciona tanto la cobertura como el ruido del contexto final [7, 8].

El *re-ranking* constituye una intervención más sofisticada. Parte del supuesto, habitual en sistemas de recuperación de información, de que conviene separar dos fases: una primera búsqueda sobre el índice optimizada por eficiencia, que entrega un conjunto candidato relativamente amplio, y una segunda fase que aplica sobre ese subconjunto un criterio más preciso de comparación entre consulta y fragmento. El reordenamiento permite situar en las primeras posiciones los candidatos efectivamente más pertinentes, con la contrapartida de un mayor coste computacional y un incremento de latencia que debe valorarse caso a caso [5, 8].

La compresión del contexto recuperado responde a una restricción distinta: la can-

tidad de información que el modelo de lenguaje puede procesar en una sola llamada está acotada por su ventana de contexto y, además, un volumen excesivo de texto tiende a degradar la atención sobre las posiciones intermedias del *prompt*. Las técnicas de compresión seleccionan, resumen o reformulan los fragmentos recuperados para conservar lo esencial en menos espacio, evitando que documentos redundantes o fragmentos prescindibles desplacen a las evidencias verdaderamente útiles [7, 8]. Conviene matizar que una compresión demasiado agresiva puede eliminar información relevante, por lo que la elección del nivel de compresión es, una vez más, una decisión sujeta a evaluación.

Un ejemplo técnico característico dentro de esta línea es la propuesta de ColBERT introducida por Khattab et al., 2020 [16] que codifica consulta y documento por separado pero retrasa la interacción fina entre sus representaciones contextualizadas hasta una fase posterior denominada *contextualized late interaction*. Este planteamiento permite acercarse a la precisión de modelos que comparan todos los pares de *tokens* entre consulta y documento sin asumir su coste completo, y se utiliza tanto en *ranking* inicial como, sobre todo, en escenarios de *re-ranking*. ColBERT no constituye un requisito de los sistemas RAG, sino un caso ilustrativo de cómo una interacción más rica entre representaciones puede integrarse en la fase *post-retrieval*. En conjunto, las técnicas descritas en este apartado pueden mejorar la calidad del contexto entregado al generador, pero comportan un incremento de coste, latencia y complejidad, por lo que su incorporación a un sistema concreto debe sustentarse en evidencia experimental antes que en su atractivo formal.

2.3.5. RAG jerárquico, *GraphRAG* y *Agentic* RAG

Las técnicas examinadas en los apartados anteriores cubren la preparación del corpus, las estrategias de recuperación, la transformación de la consulta y la optimización del contexto entregado al componente generativo. Junto a este conjunto, la literatura RAG propone variantes arquitectónicas de mayor alcance que reorganizan el flujo completo del sistema o introducen niveles adicionales de control sobre la recuperación. La revisión sistemática de Gao et al., 2023 [5] recoge estas variantes como parte de la evolución hacia arquitecturas modulares y patrones avanzados de recuperación, dejando constancia de que constituyen extensiones del paradigma básico y no sustituciones.

El RAG jerárquico organiza la información indexada en varios niveles que pueden corresponder, según el corpus, a documentos completos, secciones, resúmenes o agrupaciones temáticas. La búsqueda puede operar primero sobre un nivel de mayor granularidad y refinarse después en niveles más específicos, o combinar varias capas para sostener una respuesta que requiera evidencias de distinto detalle. Esta organización resulta especial-

mente útil cuando el corpus es extenso o presenta una estructura natural en niveles, ya que permite articular recuperación local y agregación de contexto sin recurrir a una única búsqueda plana [5].

GraphRAG, propuesto por Edge et al., 2024 [17] representa una variante distinta en la que el índice deja de ser una colección de fragmentos vectorizados para convertirse en un grafo de entidades, relaciones y comunidades extraídas del corpus, sobre el que se generan resúmenes jerárquicos. Este planteamiento se orienta a preguntas globales que requieren una visión de conjunto sobre el material, situaciones en las que la recuperación local de fragmentos próximos a la consulta resulta insuficiente. La contrapartida es un aumento significativo de la complejidad: la construcción del grafo, la detección de comunidades, la generación de resúmenes y su validación introducen coste, mantenimiento y dificultades de evaluación que exceden el alcance de un RAG documental básico, razón por la que su consideración encaja mejor como evolución futura que como componente de un sistema inicial.

Por último, *Agentic* RAG introduce agentes o controladores capaces de decidir, en tiempo de inferencia, qué herramientas utilizar, cuándo acudir al índice, si reformular la consulta, si lanzar una nueva búsqueda tras un primer intento o si combinar varias acciones para resolver consultas complejas. Los manuales recientes describen este enfoque como una extensión de la lógica modular discutida en apartados anteriores, en la que los módulos no se ejecutan en orden fijo sino bajo la dirección de un componente que orquesta decisiones [8]. Estas variantes amplían el espacio de diseño del paradigma RAG, pero introducen incertidumbre, coste, latencia, superficie de fallo y dificultad de validación; por ello su adopción debe sostenerse en una necesidad demostrada del problema y en evidencia experimental, en lugar de plantearse como requisito de partida.

2.4. Evaluación de sistemas RAG

La evaluación de los sistemas RAG requiere analizar de forma diferenciada la calidad de la recuperación, la generación de las respuestas, la trazabilidad de las evidencias y el comportamiento global del sistema. Cada una de estas dimensiones responde a objetivos distintos y exige métricas específicas, ya que un buen rendimiento en una de ellas no garantiza necesariamente resultados adecuados en las demás. Esta sección presenta las principales métricas empleadas para evaluar los componentes de un sistema RAG, así como las magnitudes operativas, las estrategias de evaluación automática y las limitaciones metodológicas que deben considerarse en un análisis extremo a extremo.

2.4.1. Evaluación de la recuperación

La evaluación de un sistema RAG no debería tratar como una única caja negra al recuperador y al componente generativo, ya que una respuesta deficiente puede deberse tanto a una recuperación que no aporta evidencias adecuadas como a una generación que las ignora o las distorsiona. La tradición de la recuperación de información, recogida en obras de referencia como la de Manning et al., 2008 [12] establece un marco metodológico transferible al caso RAG: para medir la calidad del recuperador se necesitan un conjunto de consultas o necesidades de información, una colección documental sobre la que opera el sistema, los resultados que este devuelve y, sobre todo, juicios de relevancia que indiquen, para cada consulta, qué fragmentos de la colección deberían considerarse relevantes.

Las dos métricas básicas son *precision* y *recall*. La primera mide la proporción de resultados recuperados que son relevantes; la segunda, la proporción de elementos relevantes de la colección que el sistema ha conseguido recuperar:

$$\text{Precision} = \frac{|R \cap R_{\text{rel}}|}{|R|}, \qquad \text{Recall} = \frac{|R \cap R_{\text{rel}}|}{|R_{\text{rel}}|}.$$

donde R designa el conjunto de fragmentos recuperados y R_{rel} el conjunto de fragmentos efectivamente relevantes para la consulta. Las dos magnitudes mantienen una tensión conocida: recuperar pocos candidatos muy filtrados tiende a elevar *precision* a costa de *recall*, mientras que recuperar muchos puede mejorar *recall* pero introduce ruido y, por tanto, degrada *precision*. Esta tensión obliga a no interpretar ninguna de ellas de forma aislada.

En sistemas RAG, el examen de la lista completa de candidatos resulta menos relevante que el comportamiento del sistema en las primeras posiciones del *ranking*, dado que solo un número limitado de fragmentos se incorpora al contexto del componente generativo. Las versiones de *precision* y *recall* acotadas al *top-k*, denominadas *Precision@K* (*Precision at K*) y *Recall@K* (*Recall at K*), responden directamente a esta restricción [12]:

$$\text{Precision}\,@K = \frac{|R_K \cap R_{\text{rel}}|}{K}, \qquad \text{Recall}\,@K = \frac{|R_K \cap R_{\text{rel}}|}{|R_{\text{rel}}|}.$$

con R_K entendido como el subconjunto formado por los K primeros resultados devueltos por el recuperador. *Precision@K* refleja la proporción de fragmentos útiles dentro de ese corte, mientras que *Recall@K* indica qué parte de las evidencias relevantes consigue situarse dentro de él; en un sistema RAG ambos valores condicionan directamente la calidad del contexto que recibirá el modelo.

Las métricas anteriores no incorporan información sobre la posición exacta de los resultados relevantes dentro del *top-k*. Cuando una sola evidencia correcta puede bastar para responder a la consulta, una métrica útil es el *MRR* (*Mean Reciprocal Rank*) introducido por Voorhees y Tice en el marco de la evaluación de sistemas de pregunta-respuesta del TREC-8 [18]. Para cada consulta se localiza la posición del primer resultado relevante y se calcula el promedio de sus inversos sobre el conjunto de consultas evaluadas:

$$\mathrm{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\mathrm{rank}(q)}.$$

donde Q es el conjunto de consultas y $\mathrm{rank}(q)$ representa la posición del primer resultado relevante para la consulta q. *MRR* resulta útil cuando importa, ante todo, que una evidencia correcta aparezca pronto en el *ranking*, pero queda limitada cuando la respuesta requiere integrar varios fragmentos relevantes, ya que ignora los resultados posteriores al primero.

La consideración conjunta de la posición y del grado de relevancia condujo a Järvelin y Kekäläinen, 2002 a proponer las métricas de ganancia acumulada descontada [19]. La idea de partida es que los fragmentos relevantes no son todos igual de útiles: pueden distinguirse niveles de relevancia, por ejemplo no relevante, parcialmente útil o directamente útil para responder. Sobre esa base, *DCG@K* (*Discounted Cumulative Gain at K*) acumula la ganancia obtenida al recorrer las K primeras posiciones del *ranking*, descontando el aporte de los resultados que aparecen más abajo. Por su parte, *nDCG@K* (*Normalized Discounted Cumulative Gain at K*) normaliza esa ganancia frente a la del *ranking* ideal, para expresar el rendimiento como una fracción del óptimo alcanzable:

$$\mathrm{DCG}\,@K = \sum_{i=1}^{K} \frac{\mathrm{rel}_i}{\log_2(i+1)}, \qquad \mathrm{nDCG}\,@K = \frac{\mathrm{DCG}\,@K}{\mathrm{IDCG}\,@K}.$$

donde rel_i representa el grado de relevancia del resultado situado en la posición i y $\mathrm{IDCG}\,@K$ corresponde a la ganancia acumulada descontada del *ranking* ideal, es decir, el valor de *DCG@K* que se obtendría si los resultados se ordenaran de mayor a menor grado de relevancia. *nDCG@K* resulta especialmente informativa cuando los fragmentos pueden clasificarse en varios niveles de utilidad, ya que penaliza tanto que los fragmentos más útiles aparezcan tarde como que el *ranking* acumule resultados solo parcialmente pertinentes en las primeras posiciones.

El conjunto formado por *precision*, *recall*, sus variantes acotadas al *top-k*, *MRR* y *nDCG@K* cubre razonablemente la calidad de la recuperación entendida como tarea clásica de localización y ordenación de candidatos. En un sistema RAG, sin embargo, esta

evaluación no agota la pregunta relevante: importa además si el contexto efectivamente entregado al componente generativo resulta adecuado para producir una respuesta fundamentada, dimensión que en la literatura RAG suele denominarse precisión contextual y que conecta la calidad de la recuperación con la calidad del contexto recuperado. Las métricas específicas que evalúan esa conexión, así como la fidelidad y la relevancia de la respuesta producida, configuran dimensiones complementarias del marco de evaluación de sistemas RAG.

2.4.2. Evaluación de la generación y la trazabilidad

Una recuperación adecuada de fragmentos relevantes no garantiza, por sí sola, que la respuesta generada sea correcta. El componente generativo puede ignorar parte del contexto recuperado, distorsionar sus contenidos, omitir afirmaciones necesarias, introducir elementos que no se siguen del material disponible o emitir citas que no respaldan lo que afirman. Los marcos automáticos de evaluación RAG propuestos por Es et al., 2023 [20] bajo la denominación *RAGAS* (*Retrieval Augmented Generation Assessment*), y por Saad-Falcon et al., 2023 [21] bajo la denominación *ARES* (*Automated RAG Evaluation System*), comparten precisamente esta orientación: la calidad de la respuesta debe medirse mediante dimensiones específicas que separen la fidelidad respecto al contexto, la pertinencia frente a la pregunta y la idoneidad del contexto efectivamente empleado.

La primera de estas dimensiones es la *faithfulness*, o fidelidad de la respuesta al contexto recuperado. Es et al. la estiman descomponiendo la respuesta en afirmaciones discretas y verificando, para cada una, si puede inferirse a partir del contexto entregado al generador:

$$\text{Faithfulness} = \frac{|V|}{|S|}.$$

donde S es el conjunto de afirmaciones extraídas de la respuesta y V es el subconjunto de afirmaciones verificadas como respaldadas por el contexto [20]. *ARES* recoge una dimensión análoga bajo la denominación *answer faithfulness*, lo que refuerza la idea de que la evaluación RAG debe contemplar de forma explícita la posibilidad de alucinaciones aun en presencia de contexto pertinente [21].

Una segunda dimensión, distinta de la anterior, es la *answer relevance*: una respuesta puede estar bien fundamentada en el contexto y no responder, sin embargo, a la pregunta planteada, ya sea por ser incompleta, por desviarse hacia información tangencial o por incluir contenido redundante. *RAGAS* aproxima esta dimensión generando

preguntas a partir de la respuesta y midiendo su similitud semántica con la pregunta original:

$$\text{AnswerRelevance} = \frac{1}{n}\sum_{i=1}^{n} \text{sim}(q, q_i).$$

donde q es la pregunta original, q_i son las preguntas generadas a partir de la respuesta y $\text{sim}(q, q_i)$ representa la similitud semántica entre ambas [20].

La tercera dimensión, denominada *context relevance* y a menudo identificada con la idea de precisión contextual, actúa como puente entre la recuperación y la generación. A diferencia de las métricas del apartado anterior, no se aplica sobre el *ranking* completo, sino sobre el contexto efectivamente entregado al componente generativo, evaluando qué proporción de su contenido resulta útil para responder a la pregunta:

$$\text{ContextRelevance} = \frac{|F_{\text{rel}}|}{|F|}.$$

donde F es el conjunto de frases del contexto entregado al generador y F_{rel} es el subconjunto de frases relevantes para responder a la pregunta [20, 21]. Una recuperación con buenas métricas en términos de *Precision@K*, *Recall@K* o *nDCG@K* puede traducirse, sin embargo, en un contexto con baja precisión contextual si el proceso de selección o compresión introduce ruido o si los fragmentos más útiles quedan desplazados por otros menos pertinentes.

Junto a estas tres dimensiones, la trazabilidad puede operacionalizarse mediante métricas específicas que, sin estar todavía consolidadas como estándares universales, resultan útiles para sistemas que se comprometen a citar evidencias. Una primera métrica operativa es la *citation accuracy*, definida como la proporción de citas emitidas por el sistema que apuntan a fragmentos documentales que realmente soportan la afirmación citada:

$$\text{CitationAccuracy} = \frac{|C_{\text{valid}}|}{|C|}.$$

donde C es el conjunto de citas emitidas y C_{valid} es el subconjunto de citas que apuntan a fragmentos que respaldan la afirmación correspondiente. Una segunda métrica operativa, definida a nivel de respuesta, es la consistencia de citas, que comprueba si el conjunto de citas de cada respuesta es coherente con su contenido:

$$\text{CitationConsistency} = \frac{|A_{\text{consistent}}|}{|A_{\text{cited}}|}.$$

donde A_{cited} es el conjunto de respuestas que incluyen al menos una cita y $A_{\text{consistent}}$ es el subconjunto cuyas citas son coherentes con el contenido generado. y se corresponden con las afirmaciones que pretenden justificar. Ninguna de estas dos magnitudes debe presentarse como estándar consagrado en la literatura: se trata de métricas operativas que permiten formalizar requisitos de trazabilidad en escenarios con exigencia explícita de verificabilidad.

Conviene, por último, tratar de forma diferenciada el caso de las preguntas para las que no existe evidencia suficiente en el corpus. Si el sistema produce respuestas fluidas en estos casos, aun en ausencia de fundamentación, se compromete la fiabilidad global del RAG. Como métrica operativa cabe definir la exactitud de abstención sobre el subconjunto de preguntas sin evidencia:

$$\text{AbstentionAccuracy} = \frac{|Q^{\text{abs}}_{\text{noev}}|}{|Q_{\text{noev}}|}.$$

donde Q_{noev} es el conjunto de preguntas para las que no existe evidencia suficiente en el corpus y $Q^{\text{abs}}_{\text{noev}}$ es el subconjunto de esas preguntas ante las que el sistema produce una abstención explícita en lugar de una respuesta. Una exactitud de abstención elevada indica que el sistema reconoce su propia falta de información en lugar de inventarla. Las dimensiones examinadas hasta aquí permiten analizar la respuesta generada y su trazabilidad, pero no agotan la evaluación del sistema completo, que también depende de latencia, coste, robustez, incertidumbre de los jueces automáticos y validación humana parcial.

2.4.3. Evaluación extremo a extremo y limitaciones

Las métricas examinadas hasta aquí evalúan componentes específicos del sistema, la recuperación, la generación o la trazabilidad, pero no agotan la pregunta sobre el comportamiento del sistema entendido como una solución completa. Un RAG puede recuperar bien y generar respuestas fieles en casos controlados y, sin embargo, resultar inviable en producción si su latencia es excesiva, si su coste computacional desborda los recursos disponibles, si falla ante consultas ambiguas o si las puntuaciones automáticas que reporta no son lo bastante estables como para sostener decisiones de diseño. La evaluación extremo a extremo añade, por tanto, una capa de magnitudes operativas y de cautelas metodológicas

sobre las métricas descritas en los apartados anteriores.

La primera de estas magnitudes es la latencia, entendida como el tiempo total transcurrido entre el envío de una consulta y la entrega de la respuesta al usuario:

$$\text{Latency} = t_{\text{response}} - t_{\text{query}}.$$

Esta magnitud puede descomponerse en los tiempos parciales de recuperación, reordenación, construcción del contexto, generación y post-proceso, sin que su análisis exija detallar la implementación concreta. Junto a la latencia, el coste computacional refleja los recursos consumidos por consulta o por lote de evaluación y depende de factores heterogéneos como el tiempo de cálculo, la memoria, el número de llamadas al modelo, los *tokens* procesados y el espacio de almacenamiento e indexación:

$$\text{Cost} = f\left(t_{\text{compute}}, m, n_{\text{tokens}}, n_{\text{calls}}\right).$$

No se trata de una métrica universal cerrada, sino de una magnitud operativa cuya formulación concreta depende del despliegue y de los recursos disponibles. Por último, la robustez describe la estabilidad del comportamiento del sistema frente a consultas problemáticas: ambiguas, incompletas, fuera del dominio del corpus, sin evidencia suficiente, que mezclan varios temas o que emplean vocabulario divergente respecto al material indexado. Puede expresarse operativamente como la proporción de consultas de estrés en las que el sistema responde de acuerdo con el criterio esperado:

$$\text{Robustness} = \frac{|Q_{\text{stress}}^{\text{correct}}|}{|Q_{\text{stress}}|}.$$

donde Q_{stress} designa el conjunto de consultas de estrés y $Q_{\text{stress}}^{\text{correct}}$ es el subconjunto de consultas en las que el sistema responde de acuerdo con el criterio esperado. Su valor depende, evidentemente, del diseño del propio conjunto de prueba.

La evaluación automática de respuestas complejas suele apoyarse, además, en la utilización de modelos de lenguaje como jueces, una práctica conocida como *LLM-as-a-judge*. Zheng et al., 2023 [22] distinguen tres modalidades principales: la comparación por pares, en la que el juez recibe una pregunta y dos respuestas candidatas y elige la mejor; la puntuación individual, en la que se asigna una nota a una respuesta aislada; y la evaluación guiada por una respuesta de referencia, que incorpora una solución esperada para orientar el juicio. Cada modalidad presenta compromisos distintos en escalabilidad, sensibilidad a pequeñas diferencias entre respuestas y dependencia de respuestas humanas

previas.

El uso de jueces LLM ofrece ventajas claras, escalabilidad, rapidez, posibilidad de obtener una explicación del propio juicio y comparación sistemática entre configuraciones del sistema, pero está acompañado de limitaciones documentadas en la propia literatura. Zheng et al., 2023 [22] identifican varios sesgos relevantes: el sesgo de posición (*position bias*), por el que el juez puede favorecer sistemáticamente la respuesta situada en una posición concreta; el sesgo de verbosidad (*verbosity bias*), por el que respuestas más extensas pueden recibir puntuaciones más altas sin aportar información adicional; y un posible sesgo de autorefuerzo (*self-enhancement bias*), por el que el juez puede favorecer respuestas producidas por su propio modelo o por modelos cercanos. A estos sesgos se suman limitaciones específicas en tareas que requieren razonamiento o cálculo, donde el juez puede errar incluso en juicios para los que el propio modelo dispondría de capacidad de resolución. Los autores sugieren varias medidas de mitigación: intercambiar el orden de las respuestas y aceptar el resultado solo cuando es consistente, incorporar ejemplos de calibración, recurrir a razonamiento explícito antes del juicio o emplear evaluación guiada por referencia cuando exista una solución esperada. La revisión humana parcial sigue siendo, en cualquier caso, una garantía complementaria difícilmente sustituible.

A los sesgos del juez se añade la incertidumbre estadística de la propia evaluación. Saad-Falcon et al., 2023 [21] plantean, en el marco *ARES*, la conveniencia de combinar la evaluación automática con un conjunto reducido de anotaciones humanas para calibrar los juicios automáticos y, sobre todo, para acompañar las puntuaciones de intervalos de confianza que reflejen la incertidumbre asociada a cada comparación entre sistemas o configuraciones. Esta combinación de magnitudes operativas, métricas específicas del *pipeline*, jueces automáticos y validación humana parcial dibuja una evaluación de sistemas RAG necesariamente compuesta: ninguna familia de métricas basta por sí sola y, en dominios donde la corrección conceptual depende de criterios expertos, la revisión humana sigue siendo el referente último frente al que se valida la fiabilidad de las métricas automáticas.

2.5. IA generativa educativa: riesgos, privacidad y trazabilidad

La revisión de los apartados anteriores ha caracterizado RAG como arquitectura, técnicas y marco de evaluación, pero ha dejado fuera una pregunta previa: en qué condiciones resulta legítimo y defendible integrar inteligencia artificial (IA) generativa en contextos educativos. Kasneci et al., 2023 [1] revisan, desde una perspectiva inter-

disciplinar, las oportunidades educativas de los modelos de lenguaje grandes y apuntan a aplicaciones como generación de preguntas y explicaciones, elaboración de materiales, apoyo a la escritura, asistencia individualizada al estudiantado y soporte al profesorado en planificación o evaluación. La Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura (UNESCO) [2] recoge una valoración similar al describir el potencial de los sistemas de IA generativa en entornos educativos y de investigación, pero acompaña esa valoración de una cautela explícita: la integración debe estar guiada por un propósito pedagógico claro, no por la mera disponibilidad técnica.

La cautela anterior no responde solo a consideraciones éticas: tiene raíces en limitaciones funcionales documentadas en la literatura. Los modelos de lenguaje grandes producen con frecuencia afirmaciones plausibles que no están respaldadas por evidencia, presentan sesgos heredados de los datos de entrenamiento, exhiben fragilidad ante reformulaciones menores de la consulta y dificultan la verificación posterior de sus salidas, ya que el texto generado es habitualmente fluido y persuasivo incluso cuando contiene errores. En entornos educativos, estas propiedades se combinan con riesgos de uso, como la dependencia acrítica por parte del estudiantado y las dificultades para preservar la integridad académica, además de con condiciones de acceso desiguales entre centros y contextos [1, 2]. Ninguno de estos problemas se resuelve mediante una mejora aislada del modelo subyacente: todos imponen requisitos sobre el sistema que lo envuelve, en forma de mecanismos de verificación, delimitación del corpus consultable, conservación de la procedencia documental y supervisión en el bucle de uso.

A esos requisitos funcionales se suma una exigencia de gobernanza pedagógica. UNESCO sitúa la responsabilidad humana y la agencia del usuario como principios rectores de la integración de IA generativa en educación [2], y las orientaciones europeas sobre IA y datos en enseñanza y aprendizaje recogen un planteamiento equivalente al asignar al profesorado y a los equipos directivos un papel activo en la selección, supervisión y revisión de cualquier herramienta basada en IA, además de subrayar la necesidad de una alfabetización específica en sus oportunidades, riesgos y límites [23]. Para un sistema concreto, este enfoque se traduce en requisitos operativos verificables: el asistente debe ser inspeccionable por el profesorado, debe permitir revisar la respuesta y sus fuentes, debe presentarse como apoyo y no como sustituto del juicio docente, y debe dejar margen para que el usuario que asume la responsabilidad pedagógica intervenga sobre el material indexado, las preguntas admitidas o el alcance de las respuestas.

Las decisiones de privacidad se desplazan, en consecuencia, desde el plano declarativo al de la arquitectura de datos. Los principios del Reglamento General de Protección de Datos (RGPD) actúan como fundamento de varias decisiones técnicas concretas [6].

El RGPD establece principios como la licitud, lealtad y transparencia, la limitación de la finalidad, la minimización, la exactitud, la limitación del plazo de conservación, la integridad y confidencialidad, la responsabilidad proactiva, la privacidad desde el diseño y por defecto, y la seguridad del tratamiento adecuada al riesgo. Implican, en la práctica, limitar qué datos se transmiten a cada componente del sistema, restringir el envío de consultas, materiales del curso o información de roles a servicios externos sin control institucional, registrar solo la información estrictamente necesaria para la trazabilidad y favorecer despliegues locales o autoalojados cuando los recursos permitan mantener el tratamiento dentro del perímetro institucional. La consecuencia operativa es una arquitectura *privacy-first*, en la que el flujo de datos queda explícitamente acotado por construcción y no por buena voluntad del operador.

A las cautelas sobre la información tratada se añade la cuestión, igualmente operativa, de qué usos del sistema resultan admisibles. El Reglamento europeo sobre IA introduce una aproximación basada en riesgos: las obligaciones reforzadas de gestión de riesgos, gobernanza de datos, documentación, trazabilidad, transparencia, supervisión humana, precisión, robustez y ciberseguridad se aplican a los sistemas que pueden afectar de forma significativa a derechos fundamentales o a ámbitos sensibles, e identifica explícitamente como usos potencialmente de alto riesgo determinadas aplicaciones educativas relacionadas con acceso, admisión, evaluación de resultados de aprendizaje o vigilancia de conductas durante pruebas [24]. Esta diferencia no es meramente clasificatoria: marca una frontera de alcance funcional. Un asistente documental supervisado por profesorado, limitado a consultar materiales docentes con citación verificable, ocupa una posición distinta a la de un sistema que evalúa, califica, decide acceso a programas o monitoriza conductas en pruebas, y la admisibilidad del primero no puede utilizarse para justificar el segundo. La Comisión Europea recoge un criterio análogo al pedir que cualquier herramienta de IA en enseñanza se valore atendiendo a su contexto de uso, nivel de autonomía e impacto sobre el alumnado [23].

Las exigencias examinadas son el enfoque centrado en la persona, la supervisión docente, la alfabetización en IA, la minimización de datos, la privacidad desde el diseño, la gestión proporcional del riesgo y la transparencia. En conjunto, convergen sobre una propiedad técnica concreta: la trazabilidad de las respuestas generadas. Si la herramienta debe ser inspeccionable, revisable, limitada al material autorizado y defendible ante criterios pedagógicos y regulatorios, su salida no puede depender únicamente del conocimiento interno del modelo. Esta constatación conecta directamente con la motivación técnica del paradigma RAG examinada en los apartados anteriores y reforzada en los manuales de implementación cuando subrayan el valor de recuperar fuentes internas y de hacer explícitas las evidencias empleadas en la generación [7]. Desde esta perspectiva,

las arquitecturas RAG resultan especialmente relevantes para entornos LMS, porque permiten vincular la generación a materiales docentes controlados, recuperables y citables, manteniendo el alcance del asistente dentro de un corpus institucionalmente definido y verificable por quienes asumen la responsabilidad pedagógica.

2.6. RAG en Moodle y sistemas LMS

La revisión técnica y metodológica de los apartados anteriores permite examinar a continuación antecedentes específicos sobre la integración de IA generativa y RAG en Moodle y otros sistemas LMS. La literatura y los desarrollos disponibles se agrupan, en términos generales, en dos vías: por un lado, propuestas académicas que documentan arquitecturas RAG integradas con Moodle y aportan evidencia experimental sobre casos de uso concretos; por otro, soluciones técnicas y repositorios orientados a la integración práctica, con distinto grado de madurez y de documentación pública. El examen que sigue se centra en el alcance declarado, el tipo de integración, la fuente documental empleada, la madurez del desarrollo, las garantías de privacidad, la trazabilidad documental y la evidencia disponible.

2.6.1. Arquitecturas académicas RAG integradas en Moodle

Evaluación automática en Moodle: Vangelova y Aleksieva-Petrova

El primer antecedente académico considerado es la propuesta de Vangelova y Aleksieva-Petrova, 2026 [25], que describe una arquitectura integrada para la evaluación automática de respuestas abiertas en Moodle apoyada en RAG y agentes LLM. El sistema recibe entregas del alumnado en Moodle, recupera contexto procedente de materiales docentes, aplica una rúbrica analítica definida en el propio LMS y devuelve calificación estructurada y comentarios al cuaderno de notas, lo que sitúa el trabajo en la intersección entre RAG, evaluación y gestión académica.

La arquitectura combina varias capas. Moodle actúa como fuente de entregas, rúbricas y materiales; la comunicación con la capa externa se realiza mediante la API de Moodle Web Services y un *plugin* local de *webhook* que intercepta eventos de entrega. Un orquestador basado en n8n encadena la recuperación de metadatos, la consulta a una base vectorial implementada sobre Supabase con pgvector y la invocación del modelo generativo, en este caso OpenAI GPT-4.1-mini para texto y OpenAI GPT-4.1 para una rama multimodal aplicada a diagramas del lenguaje unificado de modelado (*Unified Mo-*

deling Language, UML). Un agente instrumentalizado coordina razonamiento, uso de herramientas y producción de un resultado estructurado que se escribe en `Moodle` mediante funciones estándar como la grabación de calificaciones del módulo de tareas.

En cuanto a datos y privacidad, el sistema procesa respuestas y adjuntos del alumnado, rúbricas analíticas y materiales docentes en formatos ofimáticos habituales. Los autores documentan medidas defendibles para un experimento académico: configuración de tipo *shadow-grading* en la que las calificaciones generadas no sustituyen a las oficiales, seguridad de la capa de transporte (*Transport Layer Security*, `TLS`) 1.3, cuenta de `Moodle Web Services` con privilegios limitados, credenciales cifradas, pseudonimización mediante identificadores numéricos y retención acotada de las trazas de ejecución. La documentación revisada reconoce, sin embargo, que las evaluaciones formales de impacto en protección de datos y en derechos fundamentales quedan pendientes para un despliegue más amplio, y que el sistema depende de servicios *cloud* externos como `OpenAI` y `Supabase`. La trazabilidad se orienta al proceso de evaluación, contexto recuperado, rúbrica aplicada, salida estructurada y verificación de la calificación registrada, y no a una conversación con citas visibles para el estudiante.

La evidencia empírica disponible procede de un estudio sobre 32 estudiantes y 160 tareas en dos secciones lingüísticas, en el que se comparan las calificaciones del sistema con las del profesor. Los autores reportan una aceleración estimada de 81,36× y una reducción del tiempo de evaluación del 98,77 %, con coeficientes *quadratic weighted kappa* por tarea entre 0,612 y 0,931 y correlaciones de Spearman de 0,894 en búlgaro y 0,826 en inglés, además de un sesgo medio cercano a cero [25]. El alcance declarado se centra, en cualquier caso, en un caso institucional controlado: una sola asignatura, muestra reducida, una única referencia humana sin evaluación *multi-rater* y robustez lingüística limitada a las lenguas analizadas.

Tutoría RAG y aprendizaje asistido en `Moodle`: Ostrowska et al.

El segundo antecedente académico considerado se desplaza del marco de la evaluación al de la asistencia educativa. Ostrowska et al., 2026 [26] presentan un sistema de tutoría RAG integrado en `Moodle`, denominado por los autores `AI Teaching & Learning Assistant`, cuyo propósito es ofrecer apoyo de aprendizaje y de docencia sobre materiales verificados por el profesorado. La propuesta combina una vertiente orientada al estudiantado, con *chatbot* interactivo, tutoría socrática y generación de cuestionarios, y una vertiente orientada al profesorado con generación supervisada de materiales bajo un flujo de revisión humana previa a la publicación. No se centra, por tanto, en calificar respuestas abiertas, sino en mediar la consulta y la elaboración pedagógica sobre los contenidos del

curso.

La arquitectura se organiza en tres capas. Un *plugin* de `Moodle` desarrollado en `PHP` actúa como punto de entrada y un *frontend* basado en `Next.js` y `React` se embebe en el LMS mediante un elemento `iframe` para alojar el *chatbot*, los cuestionarios y el panel de progreso. Un *backend* desarrollado en `Python` sobre `FastAPI` coordina el *pipeline* RAG, reconoce la intención de la consulta mediante un modelo de *embeddings* ligero y la enruta hacia el modo correspondiente. La capa de datos combina `ChromaDB` para los fragmentos vectorizados, `PostgreSQL` para los datos estructurados del curso y `MinIO` como almacenamiento compatible con el servicio de almacenamiento simple (*Simple Storage Service*, `S3`) para los ficheros originales. El *backend* admite modelos compatibles con `OpenAI` y `Ollama`, lo que permite plantear tanto configuraciones basadas en servicios externos como configuraciones locales o autoalojadas.

El sistema se construye sobre materiales docentes en formatos `PDF`, `DOCX` y `PPTX`. Una de las funcionalidades más relevantes para esta revisión es el denominado `Source Reader`, que permite al estudiante desplegar la respuesta de la IA y consultar los fragmentos documentales y las páginas en las que se apoya, junto con el documento de origen. A esta trazabilidad documental se añade un flujo de revisión humana en el lado docente, en el que los materiales y cuestionarios generados permanecen marcados como no revisados hasta su aprobación. La documentación describe un diseño que busca aislar datos entre cursos y admitir despliegue local, aunque no desarrolla con el mismo detalle medidas formales de privacidad ni análisis específicos de RGPD.

La evaluación se realiza en dos planos complementarios. En el plano técnico, el sistema se mide con `RAGAS` sobre las dimensiones de *faithfulness*, *answer relevance*, *context recall* y *context precision*, con valores de *faithfulness* en torno a 0,97 en configuraciones de ciencia, tecnología, ingeniería y matemáticas (*Science, Technology, Engineering and Mathematics*, STEM) y *context recall* cercano a 1,00 en humanidades para ajustes determinados de fragmentación y temperatura; los autores observan, además, que aumentar el *top-k* más allá de diez no mejora el *recall* y degrada la precisión contextual por incremento de ruido. En el plano de usuario, un estudio preliminar con dieciocho participantes valora la resistencia percibida a alucinaciones, la utilidad para iniciar conversación, la relevancia de las respuestas y la recomendación global, con puntuaciones medias favorables y comentarios cualitativos que destacan el `Source Reader` y el control docente [26]. El alcance declarado se centra en un prototipo institucional, con muestra reducida y sin validación longitudinal en cursos reales.

2.6.2. Soluciones técnicas y repositorios RAG para Moodle

Terus RAG

El primer antecedente considerado dentro de las soluciones técnicas y repositorios es Terus RAG, un bloque de Moodle disponible en el directorio oficial de Moodle Plugins como block_terusrag y mantenido en un repositorio público asociado [27]. A diferencia de los trabajos académicos revisados en el bloque anterior, no se trata de un *paper* con validación empírica, sino de un *plugin* de bloque integrado en el ecosistema oficial de extensiones de Moodle, lo que lo sitúa en la frontera entre prototipo y producto comunitario. Su consideración resulta pertinente porque su caso de uso declarado se aproxima a la consulta documental asistida por IA generativa sobre contenidos del curso.

Desde el punto de vista funcional, la documentación pública describe una arquitectura RAG organizada en capas internas del propio bloque: extracción y fragmentación de contenido, generación de *embeddings*, almacenamiento vectorial y ensamblado de contexto. El *plugin* declara soporte para varios proveedores LLM, Google Gemini, OpenAI y Ollama, y para distintos almacenes vectoriales, incluyendo la propia base de datos de Moodle, ChromaDB o Supabase, así como un mecanismo de *ranking* híbrido basado en BM25 y similitud coseno y tareas programadas de indexación. La integración con Moodle es directa: el *plugin* se instala como bloque dentro de un curso o del panel principal, sin requerir un servicio externo independiente para alojar la lógica RAG.

Desde la perspectiva de privacidad y control, la flexibilidad de proveedor condiciona el comportamiento del sistema: la presencia de Ollama permite una configuración local o autoalojada, mientras que la selección de Gemini u OpenAI implica el uso de API externas. La documentación revisada no desarrolla, sin embargo, una estrategia explícita de minimización de datos, gestión de *logs* ni análisis específico de RGPD. Tampoco consta con suficiente precisión si el sistema devuelve a la persona usuaria fuentes verificables, fragmentos citados o identificadores concretos de recurso de Moodle como soporte de cada respuesta.

Como evidencia, la fuente revisada se limita a documentación funcional, repositorio público y configuraciones por parámetros, sin que conste una evaluación académica del *plugin* ni métricas de recuperación, fidelidad o satisfacción. Algunos comentarios disponibles en la propia página del directorio sugieren además un mantenimiento desigual entre versiones. Su valor como antecedente reside, por tanto, en su proximidad al ecosistema de Moodle y en su carácter de solución aplicada distribuible, más que en una validación empírica contrastada.

Microsoft moodle-ai-assistant

El repositorio microsoft/moodle-ai-assistant es el segundo antecedente técnico considerado, que distribuye un *plugin* local de Moodle denominado local_aichat concebido como acelerador para desplegar un asistente conversacional basado en IA generativa dentro del LMS [28]. La documentación lo identifica como versión 0.1.0 *Alpha*, lo que aconseja interpretar su madurez con cautela: no se trata de un producto consolidado ni de una solución académicamente evaluada, sino de un repositorio técnico procedente de un actor industrial relevante, útil como referencia de arquitectura aplicada para la integración LMS + IA/RAG.

Desde el punto de vista funcional, el *plugin* añade a Moodle un *widget* flotante de *chatbot* en páginas de curso y actividad, con respuestas en tiempo real entregadas mediante *Server-Sent Events*. El procesamiento se concentra en el propio *plugin*, que integra módulos para la interacción con Azure OpenAI, sanitización de entrada y salida, limitación de tasa, un componente tipo *circuit breaker* y un *pipeline* RAG. Este *pipeline* extrae contenido de un amplio conjunto de tipos de recursos y actividades de Moodle, genera *embeddings* con la API de Azure OpenAI, los almacena en la propia base de datos del LMS, los recupera por similitud coseno e inyecta el contexto de la página activa en la consulta. La reindexación se automatiza mediante tareas programadas con detección de cambios por *hash* de contenido.

En cuanto a privacidad y trazabilidad, la propuesta incorpora un aviso de privacidad superpuesto, mecanismos de exportación y eliminación de datos coherentes con la lógica RGPD, un proveedor de privacidad propio del *plugin* y un visor anonimizado de *logs* de conversación, junto con limitación de tasa por usuario y defensa frente a *prompt injection*. El despliegue se proporciona mediante Dockerfile y docker-compose para desarrollo, instalación manual y empaquetado en formato ZIP. La arquitectura se construye, sin embargo, en torno a Azure OpenAI, lo que implica una dependencia explícita de servicios *cloud* externos y desaconseja interpretarla como solución local o autoalojada por defecto. La trazabilidad operativa está bien documentada mediante hilos, exportación de transcripciones, *feedback* de respuestas y paneles docente y administrativo, pero la documentación pública no permite establecer con precisión si cada respuesta presenta a la persona usuaria fuentes verificables, fragmentos citados o identificadores concretos de recurso de Moodle.

Como evidencia, la fuente revisada aporta un repositorio activo publicado por Microsoft, documentación técnica con guías separadas para usuario, administración, desarrollo, arquitectura, API y seguridad, y una primera *release* pública. No se ha localizado, en cambio, evaluación académica, métricas RAGAS, estudio de usuarios ni validación

institucional asociada al repositorio. Su valor como antecedente reside, por tanto, en la completitud de la arquitectura aplicada que documenta y en su proximidad al ecosistema de `Moodle`, más que en evidencia empírica contrastada; el propio etiquetado como versión 0.1.0 *Alpha* refuerza la cautela con la que conviene tratarlo en una revisión del estado del arte.

2.7. Síntesis crítica y hueco identificado

La revisión técnica de los apartados anteriores muestra que RAG ofrece una vía consistente para fundamentar la generación en evidencias documentales recuperadas en tiempo de inferencia. El paradigma permite actualizar el conocimiento sin reentrenar el modelo generativo, reducir parcialmente las alucinaciones, condicionar la respuesta por contexto externo y sostener cierta trazabilidad mediante las fuentes empleadas, además de admitir una evaluación diferenciada por componentes, recuperación, generación y trazabilidad, en lugar de tratar el sistema como una caja negra única. Estas propiedades, sin embargo, no son automáticas: dependen de un conjunto de decisiones técnicas concretas que recorren la preparación del corpus, la granularidad y los metadatos de los fragmentos documentales, la elección del modelo de *embeddings* y de la base vectorial, la estrategia de recuperación, la selección o reordenación del contexto, la construcción del *prompt* y los procedimientos con los que se evalúa cada configuración.

A esas decisiones técnicas se superponen las exigencias específicas del contexto educativo. Una IA generativa integrada en un entorno docente no se justifica solamente por su capacidad para producir respuestas fluidas: requiere supervisión docente, minimización de datos, limitación de finalidad, privacidad desde el diseño, transparencia, trazabilidad verificable y una delimitación funcional cuidadosa entre asistencia, tutoría, evaluación y decisión académica. Los marcos institucionales y regulatorios revisados refuerzan, además, la conveniencia de mantener intervención humana en cualquier punto donde la decisión pueda afectar al alumnado. La integración en un LMS añade un componente estructural propio: los materiales están organizados por cursos, secciones, actividades y recursos, y los usuarios disponen de roles y visibilidades distintas, de modo que el sistema debería operar respetando esa organización en lugar de tratar el corpus como una colección plana.

Los antecedentes específicos revisados sobre RAG en `Moodle` y otros LMS cubren piezas relevantes de este problema, pero con objetivos distintos y de forma parcial. La propuesta de Vangelova y Aleksieva-Petrova integra `Moodle`, RAG y agentes LLM para la evaluación automática de respuestas abiertas, con evidencia empírica sobre eficiencia y concordancia, pero se centra en calificación y depende de servicios *cloud* externos. Os-

trowska et al. se aproximan más a la consulta sobre materiales docentes mediante un sistema de tutoría que incorpora `Source Reader` y revisión humana del profesorado, evaluado con `RAGAS` y un estudio preliminar, aunque en un prototipo institucional con validación todavía limitada. `Terus RAG` ofrece un *plugin* de `Moodle` aplicado con varios proveedores y almacenes vectoriales, pero sin evaluación académica ni desarrollo detallado de privacidad y trazabilidad documental. El repositorio `microsoft/moodle-ai-assistant` aporta una arquitectura técnica más completa, con *dashboards* y mecanismos de seguridad operativa, pero está construido en torno a `Azure OpenAI` y se distribuye explícitamente como versión 0.1.0 *Alpha* [25, 26, 27, 28].

El hueco que se desprende de esta revisión no es la ausencia de propuestas RAG para `Moodle`, ya que los antecedentes analizados muestran distintas aproximaciones académicas y técnicas al problema. La carencia principal se sitúa en que no se observan combinadas, de forma simultánea y suficientemente documentada, una ejecución local o autoalojada que evite la dependencia estructural de `API` comerciales, una integración `Moodle`/LMS sensible al contexto de curso, sección, actividad, recurso, usuario, rol y visibilidad, y una separación clara de responsabilidades entre el LMS como fuente documental, la capa de orquestación, el motor RAG, el almacén vectorial, el modelo generativo y la interfaz. Tampoco aparece suficientemente documentada una ingesta multiformato de `PDF`, `DOCX`, `PPTX`, texto plano y otros materiales docentes que los transforme en una representación común, enriquecida con metadatos educativos que conserven procedencia, contexto y visibilidad de cada fragmento documental, ni una trazabilidad documental con citas verificables que vincule cada respuesta con evidencias concretas. Queda igualmente abierta la integración de una evaluación técnica reproducible sobre recuperación, generación, fidelidad, relevancia, precisión de citas, abstención y robustez, junto con un mecanismo de actualización de la base vectorial por parte del profesorado que permita reindexar los materiales cuando cambian los recursos del curso y mantener bajo control docente qué conocimiento pasa a formar parte del RAG. La arquitectura que se desarrolla en los capítulos siguientes se orienta precisamente a explorar esa combinación integrada.

II

Parte

Marco de investigación y diseño

3 Objetivos, preguntas de investigación y metodología

Este capítulo concreta el propósito del TFM y establece el marco que orienta el diseño, la implementación y la evaluación del sistema propuesto. Para ello, se presentan el objetivo general y los objetivos específicos, se formulan las preguntas de investigación y se describe la metodología aplicada, basada en principios de investigación en diseño (*design science*). Asimismo, se identifican los requisitos funcionales y no funcionales, las restricciones del proyecto, los criterios utilizados para validar el prototipo y las principales amenazas que condicionan la interpretación y la generalización de los resultados.

3.1. Objetivo general

El objetivo general del trabajo consiste en diseñar, implementar y evaluar una arquitectura RAG modular, multiformato y sensible al contexto que pueda integrarse con `Moodle` y ofrecer respuestas docentes trazables sobre el material del curso. La separación de responsabilidades adoptada es explícita: la `API REST` de `Moodle` se utiliza como canal de ingesta documental; el servicio `FastAPI` expone el *endpoint* `/ask` como interfaz de consulta al RAG; el bloque de `Moodle` en `PHP` se restringe a la interfaz de usuario dentro del LMS; y el motor RAG, implementado íntegramente en `Python`, concentra las responsabilidades de recuperación, generación, citación, abstención y evaluación. Toda la inferencia se ejecuta con modelos locales sobre un despliegue *self-hosted*, sin recurrir a `API` comerciales, alineando el sistema con la restricción *privacy-first*. La formulación de este objetivo se apoya en el análisis crítico del estado del arte presentado en el Capítulo 2 y se traduce en un compromiso técnico verificable mediante un prototipo funcional y un marco de evaluación reproducible.

3.2. Objetivos específicos

El objetivo general se desglosa en los seis objetivos específicos (OE) siguientes:

OE1. Revisar y sistematizar el estado del arte en sistemas RAG aplicados a entornos

educativos y a la gestión documental, identificando los patrones arquitectónicos y las prácticas de evaluación relevantes para el caso de `Moodle`.

OE2. Definir una arquitectura modular *privacy-first* que separe con claridad la ingesta documental, la recuperación, la generación, la citación y la integración con `Moodle`.

OE3. Implementar un prototipo funcional del sistema con soporte multiformato (`PDF`, `DOCX`, `PPTX`, `HTML`, `Markdown`, *notebooks*) y despliegue reproducible mediante contenedores.

OE4. Construir un conjunto de preguntas de evaluación anotado sobre el corpus real del curso, con granularidad de fragmento (*chunk-level*) para permitir un análisis fino de la recuperación.

OE5. Diseñar y ejecutar un protocolo experimental que cubra recuperación, generación, citación y abstención, contrastando métricas deterministas con evaluación asistida por jueces LLM locales y con la revisión del autor.

OE6. Documentar las limitaciones del prototipo, las líneas de trabajo posteriores y las decisiones de diseño relevantes de forma trazable y justificable.

La verificación de estos objetivos se articula desde el Capítulo 4 hasta el Capítulo 9. La Tabla 3.1 sintetiza la relación entre cada objetivo específico, las preguntas de investigación (*Research Questions*, RQ) asociadas, los criterios de validación aplicables (§3.6) y el capítulo en el que se verifica.

3.3. Preguntas de investigación

El trabajo se articula en torno a cuatro preguntas de investigación, que se responden empíricamente en los capítulos de resultados y discusión:

RQ1. ¿En qué medida una arquitectura RAG con recuperación híbrida y abstención calibrada mejora la calidad de las respuestas frente a *baselines* léxicos o densos aislados sobre el corpus de un curso universitario?

RQ2. ¿Es viable ejecutar el sistema completo con modelos locales (*privacy-first*) sin degradar la utilidad para el estudiante ni comprometer la latencia percibida?

RQ3. ¿Qué componentes del *pipeline* (recuperación, *re-ranking*, política de *prompt*, abstención) aportan el mayor valor incremental cuando se ajustan de forma individual?

RQ4. ¿Qué límites reales presenta la evaluación asistida por jueces LLM locales frente a las métricas deterministas y a la revisión humana?

Cada pregunta se conecta con una sección específica del Capítulo 7 y con el análisis interpretativo del Capítulo 8. La correspondencia entre estas preguntas y los objetivos específicos que las abordan se resume en la Tabla 3.1.

Tabla 3.1: Matriz de trazabilidad: objetivos específicos, preguntas de investigación, criterios de validación y capítulos de verificación.

OE	**RQ**	**Criterio (§3.6)**	**Verificación**
OE1	—	Marco preparatorio; no aplica criterio operativo.	Cap. 2
OE2	RQ2	*Privacy-first*; trazabilidad de decisiones.	Cap. 4
OE3	RQ1, RQ2	Funcionamiento extremo a extremo; coste local.	Cap. 5, Cap. 7
OE4	RQ1	Métricas de recuperación sobre anotación *chunk-level*.	Cap. 6, Cap. 7
OE5	RQ1, RQ3, RQ4	Recuperación, abstención, ausencia de *prompt leakage*, latencia.	Cap. 7
OE6	RQ4	Trazabilidad técnica.	Cap. 8, Cap. 9

3.4. Metodología de investigación aplicada

El trabajo se enmarca metodológicamente como investigación aplicada con componentes de investigación en diseño (*design science*) [29]. El proceso seguido combina las siguientes fases, ejecutadas de forma iterativa y alineadas con metodologías de investigación en diseño para sistemas de información [30]:

1. Revisión crítica del estado del arte (Capítulo 2) y análisis de las soluciones RAG existentes para LMS.
2. Diseño de una arquitectura de referencia (Capítulo 4) alineada con los requisitos derivados del problema y con las restricciones *privacy-first*.
3. Implementación iterativa del prototipo (Capítulo 5) con decisiones técnicas documentadas de forma trazable y con instrumentación explícita para evaluación.
4. Construcción del conjunto experimental y del protocolo de evaluación (Capítulo 6), incluyendo la anotación *chunk-level* revisada por el autor.

5. Ejecución de la evaluación (Capítulo 7) con métricas deterministas, jueces LLM locales y revisión del autor.

6. Análisis interpretativo (Capítulo 8) y consolidación de conclusiones y líneas futuras (Capítulo 9).

Cada iteración se cierra con criterios de aceptación explícitos y con la actualización correspondiente de la traza de implementación, lo que permite reproducir la secuencia de decisiones sin depender de memoria implícita.

3.5. Requisitos derivados del problema

Los requisitos que orientan el diseño se han derivado del problema descrito en el Capítulo 1 y del análisis del estado del arte (Capítulo 2). Se clasifican en requisitos funcionales, no funcionales y restricciones.

3.5.1. Requisitos funcionales

Los requisitos funcionales (RF) del prototipo se enumeran a continuación para facilitar su trazabilidad con la arquitectura (Capítulo 4), la implementación (Capítulo 5) y los resultados (Capítulo 7):

RF1. Ingesta documental multiformato del corpus del curso.

RF2. Recuperación filtrada por contexto de curso y rol.

RF3. Generación de respuestas con citas verificables.

RF4. Abstención explícita ante evidencia insuficiente.

RF5. Integración con `Moodle` mediante `API REST`.

RF6. Instrumentación completa para la evaluación experimental.

Cada requisito funcional se vincula con las secciones del Capítulo 5 que documentan su realización.

3.5.2. Requisitos no funcionales

Los requisitos no funcionales (RNF) prioritarios se enumeran a continuación:

RNF1. Privacidad: ejecución local, sin `API` comerciales.

RNF2. Reproducibilidad: entornos versionados y contenedores.

RNF3. Modularidad: separación clara de responsabilidades.

RNF4. Latencia percibida aceptable en la consulta educativa.

RNF5. Tolerancia a fallos del modelo de inferencia mediante abstención controlada.

3.5.3. Restricciones

Las restricciones (RES) bajo las que se ejecuta el trabajo son firmes en materia de privacidad y despliegue y se formalizan como sigue:

RES1. Prohibición explícita de `API` comerciales para generación o evaluación.

RES2. Ausencia de servicios en la nube externa.

RES3. Integración con `Moodle` únicamente por canales oficiales, como la `API REST`, la interoperabilidad de herramientas de aprendizaje (*Learning Tools Interoperability*, `LTI`) 1.3 o `Moodle Backup`.

RES4. Despliegue mediante contenedores gestionables por equipos docentes técnicos.

Estas restricciones acotan las decisiones de diseño y se reflejan en la justificación arquitectónica del prototipo.

3.6. Criterios de validación

Los criterios de validación aplicados al prototipo son los siguientes:

- funcionamiento extremo a extremo sobre el corpus real de un curso, con integración funcional entre `Moodle`, `FastAPI` y RAG verificada;

- cumplimiento de umbrales mínimos en las métricas deterministas de recuperación, *Precision@5* (P@5), *Recall@5* (R@5), MRR y nDCG@5, sobre el conjunto anotado *chunk-level*;
- robustez frente a preguntas fuera del temario mediante abstención correcta;
- ausencia de fugas de plantilla en el *prompt* (*prompt leakage*) verificada por inspección estructural;
- trazabilidad de las decisiones técnicas relevantes, incluyendo su motivación, alternativas consideradas e impacto sobre la arquitectura;
- latencia observada del flujo de consulta, incluyendo media y percentiles cuando estén disponibles;
- coste computacional asociado a la ejecución local del prototipo, considerando las restricciones de un despliegue *self-hosted*.

Los criterios alcanzados, parcialmente alcanzados o reservados como línea futura se detallan en el Capítulo 9. La Tabla 3.1 traza cada criterio al objetivo específico que lo motiva y al capítulo en el que se verifica.

3.7. Amenazas a la validez

El diseño experimental está sujeto a varias amenazas a la validez que se declaran explícitamente para no exagerar el alcance de las conclusiones. Entre las más relevantes:

- la evaluación se ejecuta sobre un único curso, por lo que la generalización a otros dominios no queda demostrada (validez externa);
- La anotación fue realizada por el autor, por lo que no se estima acuerdo interanotador (validez de constructo);
- los jueces LLM locales presentan un acuerdo con humanos limitado en dimensiones de citación material, tal como se documenta en el marco de suficiencia (validez interna);
- los tiempos de latencia dependen del entorno concreto de ejecución y no son comparables directamente con instalaciones ajenas (validez de conclusión);

- el conjunto de preguntas, aunque diseñado para cubrir los temas del curso, refleja el conocimiento previo del autor sobre el corpus, lo que introduce un sesgo inherente al proceso de construcción.

El análisis interpretativo del Capítulo 8 retoma estas amenazas y discute sus implicaciones para las respuestas a las preguntas de investigación.

4 Arquitectura RAG *privacy-first* integrada en `Moodle`

Este capítulo presenta la arquitectura del sistema RAG propuesto y concreta las decisiones de diseño derivadas de los objetivos, los requisitos y el estado del arte. En primer lugar, se describen los requisitos arquitectónicos y la separación de responsabilidades entre `Moodle`, `FastAPI` y el motor RAG implementado en `Python`. A continuación, se detallan el modelo de contexto, los metadatos y permisos, y los flujos de indexación y consulta. Finalmente, se exponen los mecanismos de citación, trazabilidad y abstención, así como las garantías *privacy-first* adoptadas para mantener el procesamiento y el almacenamiento dentro de un entorno controlado.

4.1. Requisitos arquitectónicos derivados del estado del arte

El análisis crítico del Capítulo 2 identifica cuatro requisitos arquitectónicos fundamentales que orientan el diseño del sistema: separación estricta entre el LMS y el motor RAG, integración por interfaces oficiales, ejecución local o *self-hosted* y trazabilidad documental de las respuestas. Estos requisitos concretan los requisitos funcionales, no funcionales y las restricciones formalizadas en el Capítulo 3 (§3.5), sin duplicarlos, y condicionan tanto la vista lógica del sistema como su vista de despliegue, cuyos componentes se presentan a continuación en §4.2.

4.2. Arquitectura general del sistema

La arquitectura se estructura en un conjunto reducido de componentes con responsabilidades bien delimitadas y contratos explícitos entre ellos. Las comunicaciones entre servicios se realizan mediante el protocolo de transferencia de hipertexto (*Hypertext Transfer Protocol*, `HTTP`). La Tabla 4.1 resume esa vista de componentes al nivel de detalle que necesita este capítulo.

Tabla 4.1: Vista de componentes de la arquitectura del prototipo.

Componente	Rol arquitectónico	Ubicación operativa
Moodle	LMS institucional y fuente documental.	Instancia *self-hosted*.
Bloque Moodle PHP	Interfaz de usuario de integración: chat RAG e indexación.	Dentro del propio Moodle.
FastAPI	Contrato HTTP de consulta e ingesta hacia el motor RAG.	Contenedor Docker.
RAG Python	Recuperación, generación, citación, abstención y evaluación.	Contenedor Docker.
ChromaDB	Persistencia vectorial local del prototipo.	Embebida en el contenedor RAG con volumen persistente.
Ollama	Servidor de inferencia local (LLM *self-hosted*).	Proceso en el anfitrión.

La Figura 4.1 sintetiza los principales flujos de datos entre estos componentes, con la separación entre el canal de ingesta, apoyado en la API REST de Moodle, y el canal de consulta expuesto por FastAPI mediante /ask.

La modularidad por contratos entre estos componentes permite sustituir o ampliar piezas sin afectar al resto del sistema, condición imprescindible para el diseño experimental descrito en el Capítulo 6 y para la evolución futura de la arquitectura. Las herramientas principales que soportan la plataforma se apoyan en documentación oficial de cada proyecto [31, 32, 33].

4.3. Separación de responsabilidades entre Moodle, FastAPI y RAG en Python

La arquitectura establece una separación explícita entre canal de consulta y canal de ingesta, y entre interfaz de usuario y motor inteligente:

- **API REST de Moodle**: canal oficial de ingesta documental [34]. El motor RAG utiliza esta interfaz para descargar los materiales del curso, no para atender consultas del usuario.
- **Bloque de Moodle en PHP para el *chat* RAG**: interfaz de usuario integrada en Moodle. Recoge la pregunta y el contexto docente y consume el *endpoint* /ask.
- **Bloque de Moodle en PHP para la indexación**: interfaz funcional para lanzar y facilitar procesos de indexación desde Moodle. Consume los *endpoints* de indexación

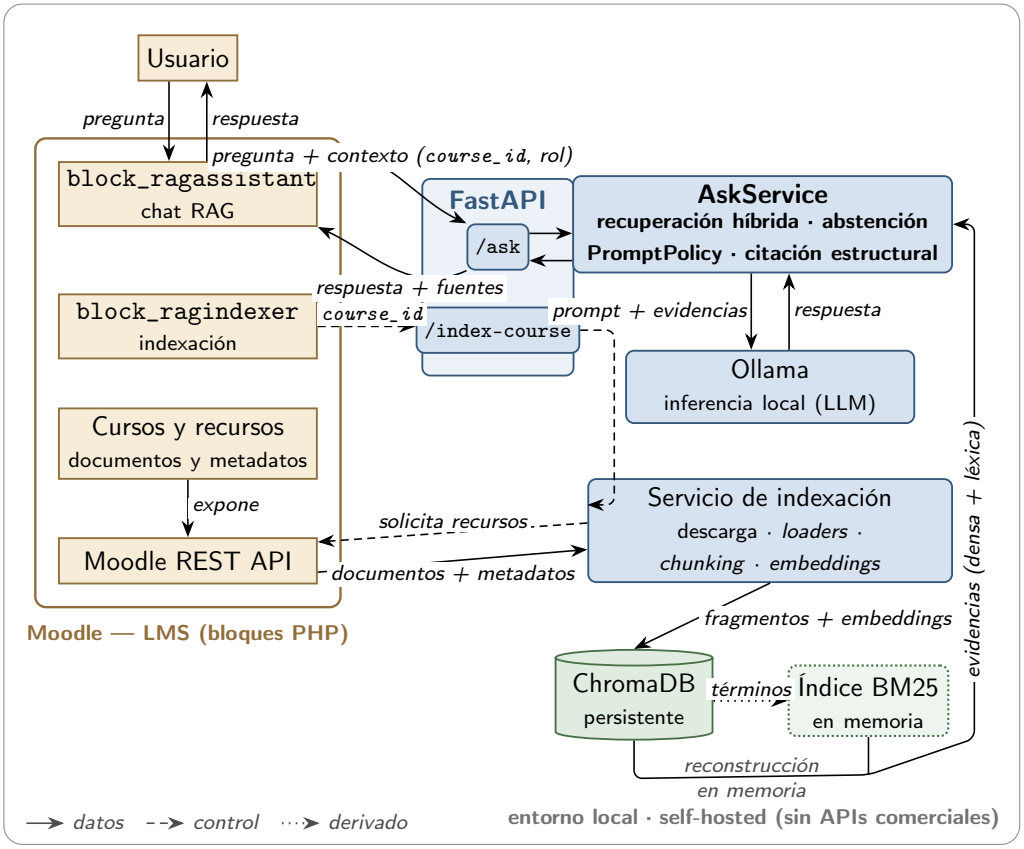

Figura 4.1: Flujo general de datos entre `Moodle` y el motor RAG: bloques `PHP` como interfaz, contratos `HTTP` de consulta e ingesta, motor RAG en `Python`, inferencia local con `Ollama` y persistencia vectorial en `ChromaDB`. Fuente: elaboración propia.

de FastAPI.

- FastAPI /ask: contrato HTTP de consulta al RAG.
- ***Endpoints* de indexación de FastAPI**: entrada HTTP hacia el *pipeline* de ingesta.
- **RAG en Python**: núcleo inteligente del sistema. Concentra las responsabilidades de recuperación, generación, citación, abstención y evaluación.
- Ollama: inferencia local sobre modelos *self-hosted*, sin dependencia de API comerciales.
- ChromaDB: persistencia vectorial local del prototipo.

El camino principal de consulta del usuario, en consecuencia, es bloque de PHP Moodle → FastAPI en Docker Compose → RAG en Python → Ollama. La API REST de Moodle queda reservada para la ingesta documental y no interviene en la consulta interactiva. El RAG no se implementa en PHP: los bloques de Moodle no ejecutan recuperación, generación, citación, abstención, *reranking* ni evaluación. Su papel es de interfaz, de construcción del contexto docente a partir de la sesión de Moodle, de envío de solicitudes HTTP y, en el caso del bloque de indexación, de disparo funcional del *pipeline*. Esta separación mantiene el motor RAG mantenible y evolucionable de forma independiente al ciclo de vida del LMS y reduce sensiblemente la superficie sobre la que puede materializarse un incidente de privacidad, al concentrar la lógica sensible en un único servicio bien delimitado.

4.4. Modelo de contexto de Moodle

El modelo de contexto que utiliza el sistema combina, de forma conceptual, la información de la sesión de Moodle en la que se origina la consulta: identificación del curso y la sección, la actividad y el recurso desde el que se accede al asistente, el rol pedagógico del usuario, su idioma preferente y la política de visibilidad aplicable. Este contexto se propaga a lo largo del *pipeline* y actúa como filtro previo a la recuperación densa, contribuyendo a restringir la búsqueda al material del curso asociado a la sesión y a la política de visibilidad aplicable. Sobre el mismo contexto se aplica una política de perfiles pedagógicos que distingue perfil de estudiante y perfil de profesorado, con el perfil de administración tratado como profesorado a efectos de generación. Este perfil condiciona el tono y el nivel de detalle de la respuesta en la fase de generación, sin alterar el conjunto de fragmentos recuperados.

4.5. Modelo de metadatos y permisos

Cada fragmento indexado se acompaña de un conjunto de metadatos ricos que soportan tanto la recuperación filtrada como el control de acceso documental. Estos metadatos se agrupan conceptualmente en cinco clases:

- **Identificación documental**: identificadores únicos del documento y del fragmento, junto con el resumen criptográfico del contenido para deduplicación y detección de cambios.
- **Contexto de Moodle**: identificadores originales de curso, sección, actividad y recurso, que permiten reconstruir la procedencia del fragmento dentro del LMS.
- **Origen y tipo de recurso**: fuente, tipo de recurso y tipo de extensiones multipropósito de correo de Internet (*Multipurpose Internet Mail Extensions*, MIME) del documento original, sin exponer rutas internas del sistema de ficheros.
- **Visibilidad**: política de visibilidad heredada del recurso de Moodle, usada como filtro previo a la búsqueda.
- **Trazabilidad y normalización**: sugerencia de título, indicación de idioma y elementos de posición estructural, como sección, subsección, página o celda, que sostienen la citación verificable de la respuesta.

El modelo de permisos se aplica como filtro previo a la recuperación vectorial (*prefilter*), no como filtrado a posteriori sobre resultados ya obtenidos. Este diseño permite aplicar filtros por curso, actividad y visibilidad, facilita restringir la recuperación al ámbito autorizado del usuario y se orienta a evitar la exposición indebida de material fuera de contexto. Su comportamiento efectivo depende de la corrección de los metadatos importados desde Moodle y de la consistencia entre la política de visibilidad del LMS y la sesión en la que se origina la consulta.

4.6. Flujo de indexación

El flujo de indexación se ejecuta de forma programática. La descarga y catalogación de recursos se resuelve mediante la API REST de Moodle: se identifican los materiales del curso, se descargan los recursos autorizados y se convierten a texto plano según su formato original, con soporte para materiales en Markdown, texto plano, PDF, DOCX, PPTX,

HTML y *notebooks* de Jupyter. Los documentos resultantes se dividen en fragmentos con un *chunking* consciente de la estructura del propio documento y se enriquecen con los metadatos descritos en §4.5. Cada fragmento se transforma en su representación vectorial mediante un modelo de *embeddings* local y se almacena en la base vectorial junto con sus metadatos.

El bloque de Moodle en PHP para la indexación se limita a actuar como interfaz funcional dentro del propio LMS: permite lanzar o facilitar la indexación de un curso sin que el operador salga de Moodle, delegando de forma íntegra la conversión, el *chunking*, el cálculo de *embeddings* y el almacenamiento en el *pipeline* de Python. No sustituye al *pipeline*, no implementa el motor RAG y no contiene lógica de recuperación ni de generación. La descripción operativa concreta del *pipeline* y de la configuración de cada etapa se desarrolla en el Capítulo 5.

4.7. Flujo de consulta

El flujo de consulta se ejecuta como respuesta a una petición /ask. El bloque de Moodle en PHP construye el contrato conceptual de la consulta a partir de la sesión activa de Moodle: pregunta del usuario y contexto docente. FastAPI lo recibe y delega en el motor RAG, que ejecuta el siguiente camino *end-to-end*:

1. El usuario formula la pregunta en el bloque de Moodle integrado en el curso.
2. El bloque construye el contexto docente y consume el contrato HTTP expuesto por FastAPI.
3. FastAPI valida la petición y delega en el motor RAG.
4. El motor aplica el filtrado por permisos y contexto sobre la base vectorial.
5. Se ejecuta la recuperación híbrida, combinando búsqueda léxica mediante BM25 y búsqueda densa mediante fusión de rangos recíprocos (*Reciprocal Rank Fusion*, RRF).
6. Se evalúa la política de abstención sobre la evidencia recuperada.
7. Se construye el *prompt* condicionado al perfil pedagógico del usuario mediante la política de *prompt* del sistema.
8. El servidor Ollama genera la respuesta con el modelo local configurado.

9. El motor verifica de forma estructural las citas de la respuesta.
10. El resultado se devuelve al bloque de Moodle en forma de respuesta con fuentes y metadatos operativos que el bloque presenta al usuario.

La Figura 4.2 ofrece la vista dinámica de este flujo, con sus fases de recepción, recuperación, decisión de abstención, generación y entrega.

La configuración operativa final adoptada en el prototipo tras la fase experimental fija los siguientes valores por defecto: modo de recuperación híbrida (RAG_RETRIEVAL_MODE=hybrid), parámetro de RRF RAG_HYBRID_RRF_K=20, umbral mínimo de abstención RAG_ABSTENTION_MIN_SCORE=0.52, *re-ranking* desactivado (RAG_RERANK_ENABLED=false), política de *answerability* desactivada (RAG_ANSWERABILITY_POLICY_ENABLED=false) y política de *prompt* activa. Esta configuración se declara aquí como decisión de diseño; su justificación empírica se desarrolla en el Capítulo 7.

4.8. Citación, trazabilidad y abstención

Toda respuesta emitida por el sistema debe apoyarse en los fragmentos recuperados y citar sus fuentes mediante marcadores numéricos [1], [2], etc., asociados de forma posicional con el contexto que se envía al modelo generativo. Sobre la respuesta producida se ejecuta una verificación estructural de las citas que actúa como capa de defensa y contempla, de forma narrativa, cuatro situaciones anómalas: citas ausentes, citas no reconocidas, citas malformadas y citas duplicadas. La detección de cualquiera de ellas se traslada al mensaje devuelto en forma de advertencia operativa asociada a la respuesta, permitiendo al usuario y al docente valorar la fiabilidad de la salida.

Cuando la evidencia recuperada es insuficiente para sostener una respuesta, el sistema se abstiene de forma explícita y emite un mensaje uniforme que informa al usuario de la ausencia de material suficiente en el corpus. Esta abstención no elimina el riesgo de respuesta no soportada, pero lo reduce sensiblemente al desplazar la decisión de responder a un umbral cuantitativo sobre la similitud de la evidencia disponible. La calibración empírica del umbral de abstención se desarrolla en el Capítulo 7.

4.9. Garantías *privacy-first*

La arquitectura asume un compromiso *privacy-first* que se traduce en medidas concretas ancladas a componentes reales del despliegue del prototipo. En primer lugar,

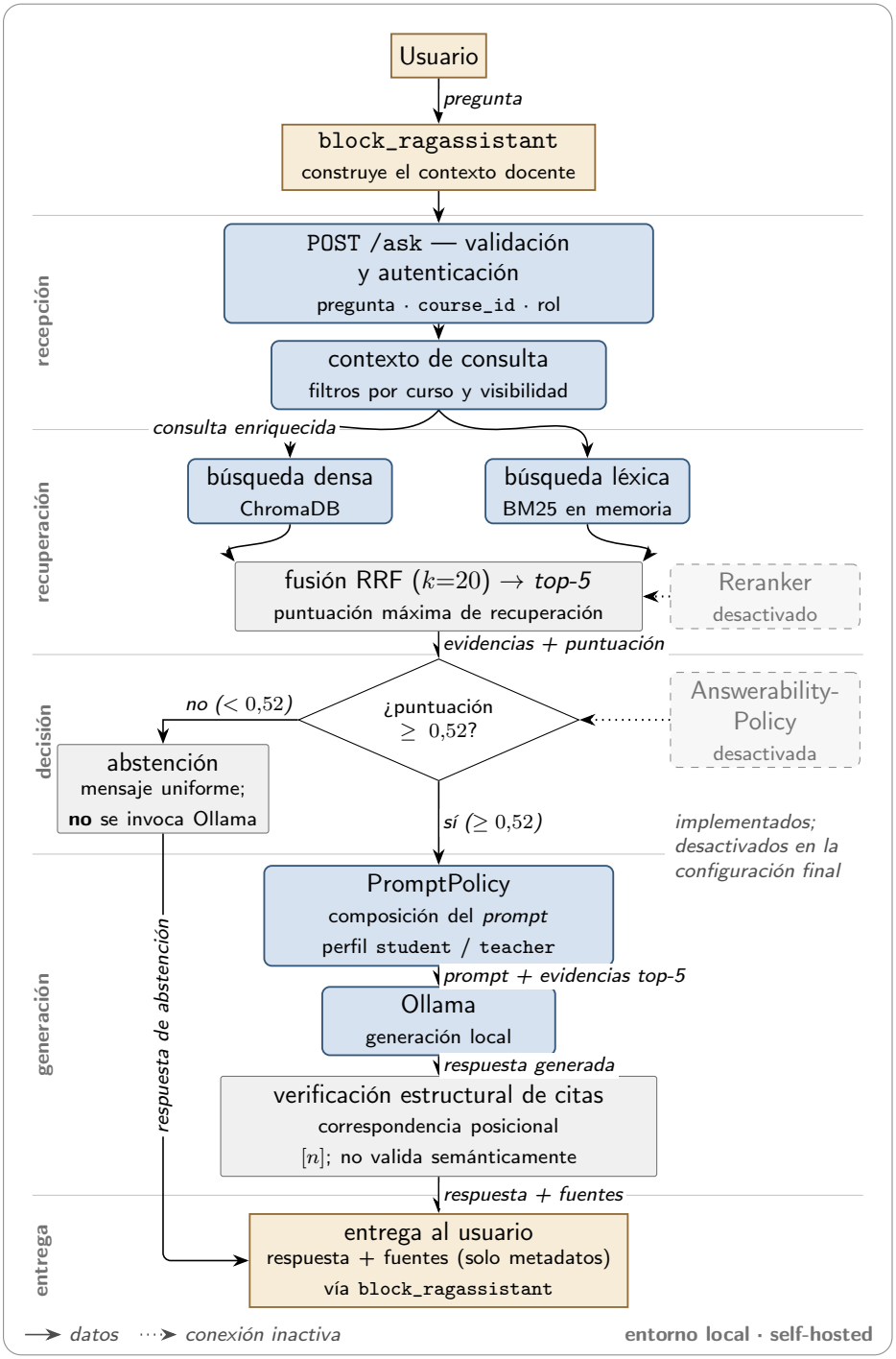

Figura 4.2: Flujo de datos durante el procesamiento de una consulta `/ask`, por fases: recepción y contexto, recuperación híbrida, decisión de abstención, generación local y entrega con fuentes. Los componentes en gris discontinuo están implementados pero desactivados en la configuración final. Fuente: elaboración propia.

los modelos de generación y de juicio se ejecutan localmente mediante Ollama, alcanzados desde el contenedor RAG a través del anfitrión, sin llamadas a API comerciales; el propio despliegue contempla la ejecución alternativa de Ollama como servicio dentro de la orquestación cuando así se requiera. En segundo lugar, el corpus documental no sale del entorno controlado: la base vectorial ChromaDB reside en un volumen persistente local del contenedor [32, 33], y la descarga de materiales de Moodle se conserva en un directorio operativo dentro del mismo entorno. En tercer lugar, los registros operativos no incluyen preguntas completas, respuestas ni fragmentos, salvo cuando se activan modos de inspección explícitos y controlados; el bloque correspondiente de Moodle registra únicamente resúmenes criptográficos de la consulta, sin conservar la pregunta ni la respuesta en claro. Por último, los *tokens* de autenticación entre Moodle, FastAPI y el motor RAG se inyectan mediante variables de entorno y no se almacenan en el repositorio ni se transmiten en *logs*. Estas garantías son verificables por inspección de la configuración del despliegue, descrita en el Anexo A.

Estas medidas delimitan el alcance del prototipo evaluado. La extensión a despliegues institucionales, escenarios con varios cursos, monitorización operativa o almacenes vectoriales orientados a mayor escala queda fuera del alcance de esta arquitectura experimental y se aborda como evolución futura en el Capítulo 9.

5 Implementación del prototipo

Este capítulo describe la implementación concreta del prototipo a partir de la arquitectura presentada en el capítulo anterior. En primer lugar, se expone la organización general de la aplicación y se detallan los módulos responsables de la ingesta, la normalización, la indexación, la recuperación y la generación. A continuación, se presentan las decisiones adoptadas para el almacenamiento vectorial, la recuperación híbrida, la citación y la abstención ante evidencia insuficiente. Finalmente, se describe la integración funcional con `Moodle` y el despliegue reproducible de los componentes mediante contenedores.

5.1. Descripción general del prototipo

El prototipo materializa la arquitectura descrita en el Capítulo 4 como una aplicación en `Python` organizada en torno a `FastAPI`, con `ChromaDB` como base vectorial local y `Ollama` como servidor de modelos locales. El código se distribuye en un paquete principal cuyas subdivisiones corresponden directamente a las capas funcionales del *pipeline*. En particular, se identifican los siguientes módulos:

- **ingesta**: descarga y *parsing* multiformato de los materiales del curso.
- **indexación**: cálculo de *embeddings*, persistencia vectorial y catalogación de fragmentos.
- **recuperación**: búsqueda densa, léxica e híbrida con fusión por rango recíproco y una etapa opcional de *re-ranking*.
- **generación**: composición del *prompt*, invocación del modelo local, verificación estructural de citas y política de abstención.
- `API`: exposición de los contratos `HTTP` mediante `FastAPI` y su cableado con los módulos anteriores.
- **integraciones**: cliente contra la `API REST` de `Moodle` y modelo de manifiesto documental derivado de `Moodle`.

- **evaluación**: instrumentación reproducible del protocolo experimental descrito en el Capítulo 6.

La Figura 5.1 presenta la organización modular del prototipo y concreta, a nivel de implementación, la arquitectura general descrita en el Capítulo 4: muestra las dependencias funcionales entre módulos, herramientas y parámetros, no la topología de despliegue.

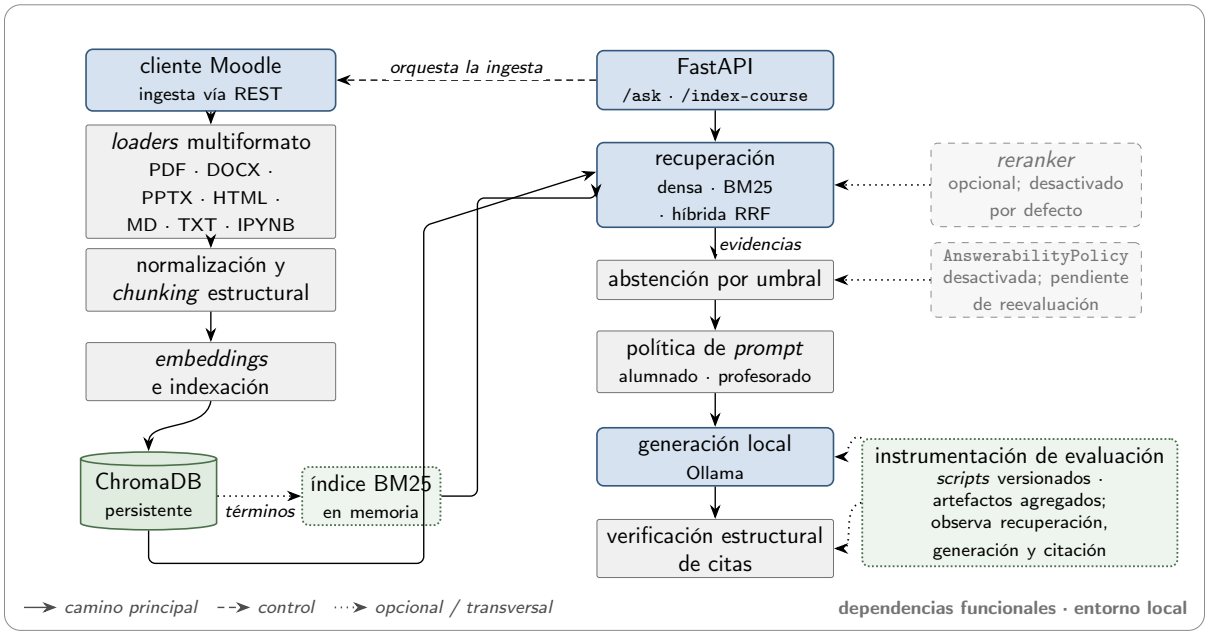

Figura 5.1: Módulos implementados del prototipo y sus dependencias funcionales: ingesta multiformato e indexación, recuperación densa, léxica e híbrida, abstención, política de *prompt*, generación local y verificación estructural de citas. Los componentes en gris discontinuo están implementados pero desactivados en la configuración final. Fuente: elaboración propia.

Este capítulo describe cada componente desde la perspectiva de su realización concreta, sin descender al detalle operativo, que queda reservado para el Anexo A, dedicado a las variables y al arranque, y para el Anexo B, dedicado a los contratos completos de la `API`.

5.2. Ingesta documental multiformato

El módulo de ingesta soporta materiales en los formatos habituales de un curso universitario: `Markdown`, texto plano (`TXT`), `PDF`, `DOCX`, `PPTX`, `HTML` y `Jupyter Notebooks`. Cada formato se procesa mediante un cargador especializado (*loader*) que preserva la estructura documental original, como secciones, diapositivas, celdas o páginas, para que la

fase posterior de fragmentación pueda aprovechar esa organización semántica. Los documentos se enriquecen con los metadatos de `Moodle` obtenidos del contexto de descarga, como curso, sección, actividad, recurso, tipo de recurso y política de visibilidad.

La Figura 5.2 detalla, como vista dinámica complementaria a la organización modular de la Figura 5.1, las transformaciones aplicadas desde la recuperación del recurso en `Moodle` hasta su incorporación a los índices.

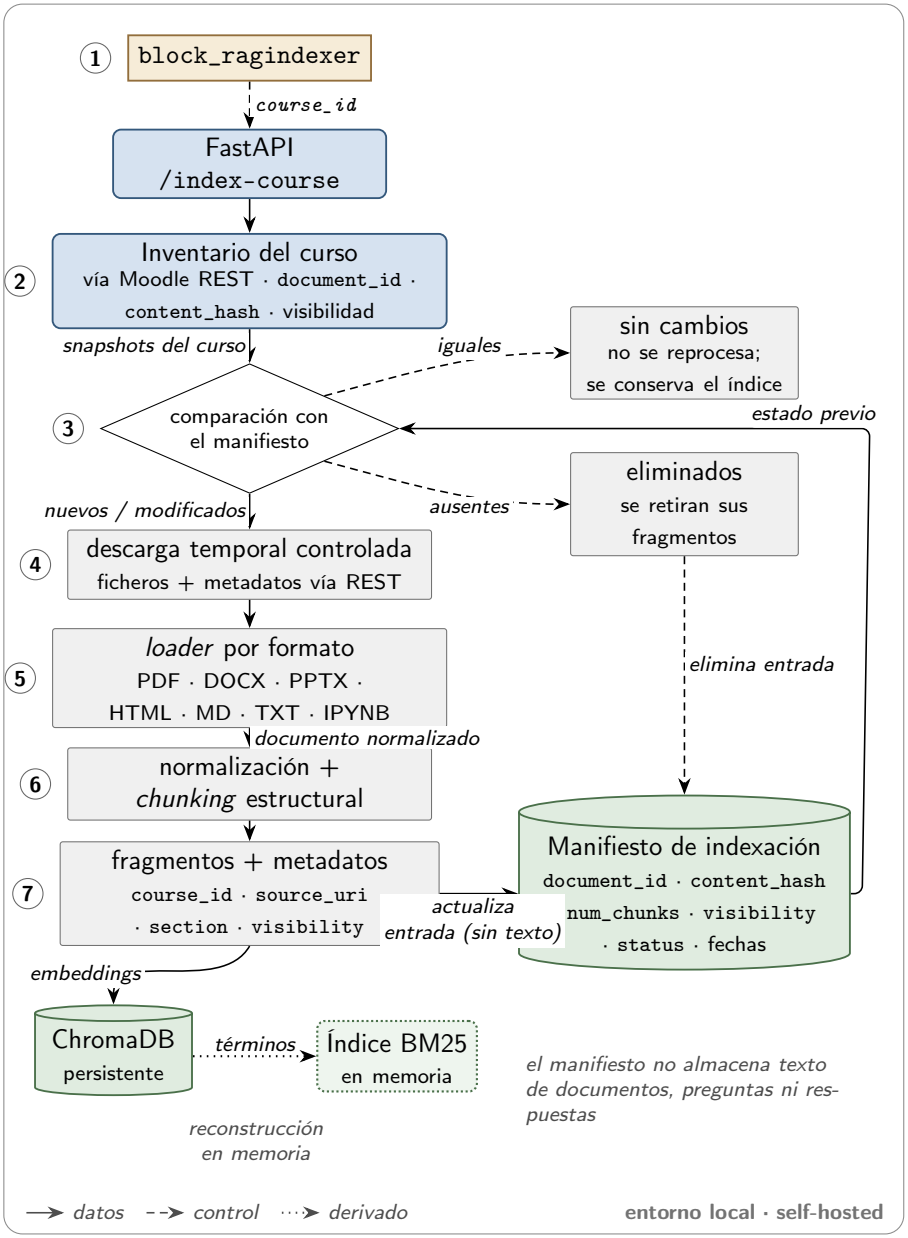

Figura 5.2: Flujo de ingesta, transformación e indexación de los recursos docentes recuperados desde `Moodle`, con clasificación incremental frente al manifiesto de indexación. Fuente: elaboración propia.

La ingesta presenta limitaciones honestas que conviene declarar. El contenido pura-

mente visual, como imágenes, esquemas escaneados o capturas, no se aprovecha porque el *pipeline* no integra un módulo local de reconocimiento óptico de caracteres (*Optical Character Recognition*, `OCR`); el reconocimiento óptico y la descripción automática de imágenes se dejan como evolución futura. Los archivos `PDF` y `PPTX` con estructura irregular, como textos organizados en cuadros, fondos gráficos o diapositivas sin marcadores de sección, pueden dar lugar a fragmentos con contexto empobrecido; el prototipo prioriza la fidelidad al texto extraído y no fuerza reconstrucciones estructurales artificiales.

5.3. Normalización, *chunking* y metadatos

Tras el *parsing*, cada documento se somete a una fase de normalización textual, que incluye limpieza, homogeneización de codificación y extracción de referencias explícitas a figuras, y a una fase de *chunking* consciente de la estructura documental. La fragmentación se apoya en las unidades semánticas naturales de cada formato, como encabezados `ATX` en `Markdown`, diapositivas en `PPTX`, celdas en *notebooks* y páginas o secciones en `PDF`, en lugar de aplicar un tamaño fijo desacoplado del contenido.

Cada fragmento resultante se etiqueta con un `chunk_id` único, un `content_hash` para deduplicación y detección robusta de cambios, la posición estructural en el documento y los metadatos de `Moodle` heredados. Este esquema soporta simultáneamente tres necesidades: la trazabilidad de las citas hasta el fragmento concreto que las sostiene, el filtrado previo a la búsqueda por contexto y visibilidad, y la actualización incremental del índice cuando cambia un documento en `Moodle`. La política de tamaño y solape del *chunking* se ha ajustado como parte de las decisiones técnicas del prototipo, documentadas de forma trazable y discutidas en el Capítulo 8.

5.4. *Embeddings* y almacenamiento vectorial

La representación vectorial de los fragmentos se calcula con un modelo local de *embeddings* multilingüe de la familia `Sentence Transformers` [35]. En el prototipo se utiliza `paraphrase-multilingual-MiniLM-L12-v2`, con dimensión 384, ejecutado dentro del propio contenedor de la `API` y sin salida a servicios externos. Esta elección concreta el compromiso *privacy-first* enunciado en §4.9: los materiales del curso y las representaciones que se derivan de ellos se mantienen en el entorno controlado del prototipo.

Los vectores se almacenan en `ChromaDB`, escogida como base vectorial de prototipo por su sencillez de despliegue, su persistencia mediante un volumen local del contenedor y

su ajuste con la escala de un curso individual. La evolución hacia soluciones más escalables, como `Qdrant` *self-hosted*, entre otras, se documenta como línea futura en el Capítulo 9: no se plantea como *refactor* inmediato, sino como paso natural cuando la escala de despliegue lo justifique.

5.5. Recuperación densa, léxica e híbrida

El sistema implementa tres modos de recuperación configurables por variable de entorno: recuperación densa (`vector`) por similitud coseno sobre los *embeddings*, recuperación léxica pura (`bm25`) sobre un índice construido en memoria a partir del mismo corpus, y recuperación híbrida (`hybrid`) que combina ambas mediante RRF [36]. El modo híbrido es el modo por defecto en la configuración final operativa del prototipo, con un parámetro de fusión $k = 20$ y un tamaño de resultado final de cinco fragmentos por consulta.

La recuperación se combina con un filtrado previo por contexto y visibilidad (*prefilter*) que restringe la búsqueda al material accesible según el objeto `QueryContext` propagado desde la petición. La comparación experimental entre los tres modos y la justificación de $k = 20$ se desarrollan en el Capítulo 7.

5.6. *Re-ranking*

El *pipeline* soporta una etapa opcional de *re-ranking* mediante un *cross-encoder* local. Como implementación de referencia se ha integrado `Qwen3-Reranker-0.6B`, con parámetros operativos por defecto de veinte candidatos iniciales y cinco resultados finales tras el reordenamiento. La etapa se activa o desactiva mediante variables de entorno y no introduce coste adicional cuando queda desactivada.

En la configuración final operativa del prototipo, el *re-ranking* está desactivado (`RAG_RERANK_ENABLED=false`) porque la evaluación no ha justificado su activación por defecto sobre el corpus considerado. La evidencia empírica que sostiene esta decisión se recoge en el Capítulo 7.

5.7. Generación local y construcción del *prompt*

La generación se realiza con modelos locales servidos por `Ollama`. El *prompt* del sistema se compone de un bloque de instrucciones comunes, como responder únicamente a partir del contexto proporcionado, citar cada afirmación relevante con marcadores `[n]`, no inventar contenido y no reproducir marcadores estructurales del propio *prompt*, y de un bloque específico según el perfil pedagógico del usuario. El perfil de alumnado prioriza un tono didáctico con definiciones guiadas, mientras que el perfil de profesorado prioriza un tono técnico y la cobertura completa del material disponible. La política `PromptPolicy` normaliza el rol de `Moodle` recibido en el `QueryContext` a uno de los dos perfiles operativos; el rol de administración se trata como profesorado a efectos de generación.

Como capa defensiva, el *pipeline* aplica un *guard* estructural frente a fugas de plantilla (*prompt leakage*) que inspecciona la respuesta generada antes de devolverla, con el objetivo de detectar la reproducción indebida de fragmentos de la plantilla o de marcadores internos. Las plantillas completas de alumnado y profesorado, así como el *prompt* base y las instrucciones de citación y abstención, se recogen en el Anexo D.

5.8. Citación y verificación estructural

Las respuestas incluyen marcadores numéricos `[1]`, `[2]`, etc., que se asocian de forma posicional con los fragmentos incorporados al contexto de la generación. Sobre la respuesta emitida por el modelo se ejecuta una verificación estructural de citas que actúa en dos direcciones: por un lado, detecta la ausencia de marcadores donde el mensaje contiene afirmaciones factuales; por otro, identifica marcadores que apuntan a fragmentos inexistentes, marcadores mal formados o marcadores duplicados en la misma respuesta. Las anomalías detectadas se trasladan al mensaje devuelto en forma de advertencias operativas que el bloque de `Moodle` puede exponer al usuario.

Conviene señalar el alcance de esta verificación de forma explícita: se trata de una comprobación estructural que garantiza la consistencia posicional entre marcadores y contexto recuperado, pero **no** valida materialmente que cada afirmación de la respuesta esté literalmente sostenida por el pasaje citado. La validación semántica de cada afirmación queda fuera del alcance del prototipo y se aborda como límite conocido en el Capítulo 8.

5.9. Abstención por evidencia insuficiente

Antes de invocar al modelo generador, el sistema evalúa una política de abstención que compara la mejor puntuación de recuperación con un umbral empíricamente ajustado. Cuando la evidencia recuperada no supera ese umbral, el *pipeline* se abstiene de forma explícita, devuelve un mensaje uniforme que informa al usuario de la insuficiencia de material y no llega a invocar al modelo. Esta política reduce el riesgo de respuesta no soportada cuando el corpus no contiene material relevante, sin pretender eliminar todas las respuestas erróneas del sistema. La configuración operativa final fija el umbral en `RAG_ABSTENTION_MIN_SCORE=0.52`; la justificación empírica de este valor y su comportamiento frente a preguntas dentro y fuera del temario se documentan en el Capítulo 7.

De forma complementaria, se ha implementado una política de *answerability* (`AnswerabilityPolicy`) que contempla respuestas parciales y turnos de clarificación ante preguntas mixtas o ambiguas. En la configuración final del prototipo esta política permanece **desactivada por defecto** (`RAG_ANSWERABILITY_POLICY_ENABLED=false`); no se ha reevaluado bajo la configuración final y se conserva como línea futura de evaluación controlada.

5.10. Integración con `Moodle` y despliegue reproducible

La integración con `Moodle` se resuelve mediante tres piezas complementarias que preservan la separación de responsabilidades establecida en §4.3. La `API REST` de `Moodle` actúa como fuente documental oficial para la ingesta: el motor RAG la utiliza para enumerar los recursos del curso y descargar los materiales autorizados según su política de visibilidad. Los bloques de `Moodle` desarrollados en `PHP` son componentes funcionales de integración estables dentro del alcance del trabajo y actúan exclusivamente como interfaz dentro del propio LMS. Se distinguen dos:

- **Bloque de *chat* RAG**: interfaz de consulta. Recoge la pregunta del usuario, construye el contexto docente, formado por curso, sección, actividad, recurso, rol derivado de las capacidades de `Moodle`, idioma y visibilidad, a partir de la sesión activa, invoca el *endpoint* `/ask` de `FastAPI` mediante un cliente `HTTP` interno, y presenta la respuesta acompañada de la lista de fuentes y de las advertencias operativas que devuelva la `API`.
- **Bloque de indexación**: interfaz funcional para lanzar la indexación desde `Moodle`. Consume el *endpoint* `/index-course` y muestra el resultado agregado devuelto por

la API, incluidos los contadores de documentos y fragmentos, las advertencias y los errores resumidos.

Ambos bloques se comportan como transportes finos: no implementan recuperación, generación, cálculo de *embeddings*, citación, abstención, *reranking* ni evaluación. Toda la lógica del sistema RAG permanece en el motor de Python. La autenticación entre los bloques y FastAPI se gestiona mediante configuración de administración sin exponer los valores en el propio Moodle. Junto a los dos *endpoints* anteriores, FastAPI expone además *endpoints* auxiliares de seguimiento, sincronización y diagnóstico. El seguimiento de indexación se apoya en /index-status/course_id. La sincronización de visibilidad se canaliza mediante /sync-visibility. Los *endpoints* /health y /health/deep se reservan para comprobaciones de disponibilidad y diagnóstico. Estos *endpoints* no son necesariamente consumidos por los bloques de Moodle.

El despliegue del prototipo se orquesta mediante Docker Compose: un servicio activo empaqueta la API y el motor RAG, monta un volumen persistente para el índice vectorial y accede al servidor Ollama del anfitrión. Las variables de entorno relevantes, los procedimientos de arranque y los esquemas completos de los contratos HTTP se recogen en el Anexo A y en el Anexo B.

III

Parte

Evaluación experimental

6 Diseño experimental y protocolo de evaluación

Este capítulo presenta el diseño experimental utilizado para evaluar el prototipo y establecer la configuración final del sistema. Para ello, se describen los objetivos de la evaluación, el corpus documental, el conjunto de preguntas y las configuraciones comparadas. A continuación, se detallan las métricas aplicadas a la recuperación, la generación, la citación, la abstención y el comportamiento operativo. Finalmente, se expone la estrategia de validación mediante métricas deterministas, jueces LLM locales y revisión del autor, así como el protocolo seguido para garantizar la consistencia y la reproducibilidad de los experimentos.

6.1. Objetivos de la evaluación

La evaluación experimental persigue tres objetivos complementarios. En primer lugar, verificar empíricamente que la arquitectura descrita en el Capítulo 4 y el prototipo implementado en el Capítulo 5 cumplen los criterios de validación definidos en el Capítulo 3 (§3.6). En segundo lugar, comparar de forma sistemática las decisiones de diseño clave, recuperación densa, léxica o híbrida, activación o no del *re-ranking*, ajuste del umbral de abstención y política de *prompt*, sobre un conjunto controlado de preguntas. En tercer lugar, contrastar la evaluación asistida por jueces LLM locales con las métricas deterministas y con la revisión del autor, en línea con la pregunta de investigación RQ4. Esta triangulación sostiene las conclusiones interpretativas del Capítulo 8 y determina la configuración final que se consolida en el Capítulo 7.

Este capítulo describe únicamente el marco metodológico y el protocolo experimental. Las cifras de rendimiento, la comparación cuantitativa entre configuraciones y la justificación empírica de la configuración final se desarrollan en el Capítulo 7.

6.2. Corpus documental

El corpus utilizado en la evaluación se compone de materiales docentes reales de un curso de posgrado sobre MLOps (`course_id`=2 de la instancia institucional de `Moodle`). Sobre este corpus se ha construido un *snapshot* autorizado, disponible como almacén vectorial de solo lectura para permitir la reproducibilidad entre configuraciones sin necesidad de reejecutar la ingesta contra el LMS. El *snapshot* contiene 313 fragmentos indexados procedentes de 17 fuentes documentales de `Moodle`, en los formatos soportados por el *pipeline* de ingesta descrito en el Capítulo 5: `Markdown`, `TXT`, `PDF`, `DOCX`, `PPTX`, `HTML` y `Jupyter Notebooks`.

El corpus es privado y sensible; en esta memoria no se reproducen títulos de documentos ni fragmentos de contenido, en coherencia con el compromiso *privacy-first* declarado en §4.9. Como limitación honesta del corpus evaluado conviene señalar dos aspectos: (a) el *pipeline* no integra `OCR` local y, por tanto, el contenido puramente visual, como imágenes, capturas o esquemas escaneados, no se aprovecha; y (b) los archivos `PDF` y `PPTX` con estructura irregular generan fragmentos con contexto empobrecido. Estas limitaciones se retoman en el Capítulo 8 como amenazas a la validez interna del prototipo.

6.3. Conjunto de preguntas de evaluación

El conjunto de preguntas de evaluación toma la forma de un *benchmark* ampliado de 106 preguntas construido sobre el mismo corpus. La distribución por ámbito (*scope*) es la siguiente:

- 51 preguntas `in_scope`, dentro del temario del curso.
- 10 preguntas `in_scope_rephrased`, reformulaciones de preguntas dentro del temario para evaluar robustez léxica.
- 16 preguntas `mixed`, que combinan una parte respondible desde el corpus con una parte no cubierta.
- 12 preguntas `vague`, cuya formulación admite varias interpretaciones legítimas.
- 17 preguntas `out_of_scope`, deliberadamente fuera del temario, orientadas a evaluar la abstención.

Sobre este *benchmark* se dispone además de una anotación *chunk-level* que declara los fragmentos relevantes para cada pregunta, con un total de 193 *chunks* relevantes. Las

métricas de recuperación se calculan sobre el subconjunto de 58 preguntas con juicios de relevancia *chunk-level* utilizables para la evaluación del *ranking*. Las preguntas restantes no se eliminan del *benchmark*, sino que se emplean en las dimensiones de abstención, robustez o análisis de limitaciones, según su tipo.

La anotación *chunk-level* siguió un criterio conservador que descarta como relevantes los fragmentos puramente visuales, las portadas de material docente y los fragmentos con contexto insuficiente para sostener una respuesta. La anotación fue realizada por el autor, por lo que no se estima acuerdo interanotador; el detalle completo del proceso de anotación y de los criterios de aceptación se recoge en el Anexo E.

6.4. Configuraciones comparadas

Las configuraciones comparadas en la evaluación se agrupan en cuatro familias funcionales, sin resultados numéricos en este capítulo:

- ***Baselines* de recuperación**: recuperación densa pura por similitud coseno y recuperación léxica pura mediante `BM25` sobre el mismo corpus.
- **Variante híbrida inicial**: recuperación híbrida mediante RRF con el valor de referencia del parámetro de fusión y el umbral de abstención empleados antes de la calibración final, tomada como referencia previa frente al valor operativo posteriormente adoptado.
- **Configuración final del prototipo**: recuperación híbrida con parámetro de fusión $k = 20$, umbral de abstención candidato 0,52, *reranking* desactivado, política de *answerability* desactivada y política de *prompt* activa.

La justificación empírica de la configuración final y el análisis comparativo detallado entre familias se desarrollan en el Capítulo 7.

6.5. Métricas de recuperación

La evaluación de sistemas RAG combina métricas de recuperación y de generación, un ámbito metodológico en rápida evolución con la irrupción de los grandes modelos de lenguaje [37]. La calidad de la recuperación se mide mediante métricas estándar de recuperación de información aplicadas a nivel de fragmento sobre el subconjunto de 58

preguntas puntuables descrito en §6.3: *Precision@5*, *Recall@5*, MRR y nDCG@5. La elección de $k = 5$ es consistente con el tamaño del resultado final del *pipeline* (§5.5) y, por tanto, refleja la evidencia que efectivamente llega al modelo generador. Se registran asimismo las latencias de la fase de recuperación, la media y los percentiles cuando estén disponibles, que permiten discutir el compromiso entre calidad y coste operativo en línea con RQ2.

Las definiciones formales de estas métricas siguen la convención estándar del ámbito de recuperación de información y no se reproducen aquí; el lector puede consultar las referencias canónicas ya introducidas en el Capítulo 2.

6.6. Métricas de generación y citación

Las métricas de generación combinan dos familias complementarias. La primera es determinista: la verificación estructural de citas comprueba la correspondencia entre los marcadores citados en la respuesta y los fragmentos efectivamente incorporados al contexto, y desglosa los fallos observados en cuatro categorías narrativas ya descritas en el Capítulo 5, citas ausentes, citas no reconocidas, citas malformadas y citas duplicadas. Sobre este desglose se calcula una tasa agregada de citas verificadas por configuración.

La segunda familia es exploratoria y utiliza el marco `RAGAS` [20], ejecutado localmente sobre modelos servidos por `Ollama`, sin llamadas a `API` comerciales. Las dimensiones consideradas son la fidelidad al contexto (*faithfulness*), la relevancia de la respuesta y la precisión del contexto. La cobertura del contexto no se calcula dentro de `RAGAS` porque el conjunto de referencia del *benchmark* almacena los fragmentos relevantes como identificadores de fragmento, y rellenar el campo de referencia con el texto completo de esos fragmentos aumentaría la exposición del contenido del corpus, en tensión con el enfoque *privacy-first* del prototipo.

Ambas familias se interpretan de forma diferenciada. La verificación estructural aporta una comprobación reproducible y determinista, aunque limitada al soporte posicional de los marcadores; `RAGAS` aporta una triangulación exploratoria adicional, pero no se emplea como criterio decisor de la configuración final.

6.7. Evaluación de abstención

La evaluación de la política de abstención se articula en torno a cinco magnitudes que separan claramente las situaciones deseables de las indeseables por tipo de pregunta:

- Tasa de abstención correcta en preguntas fuera del temario: proporción de preguntas fuera del temario en las que el sistema se abstiene correctamente.
- Tasa de respuesta indebida en preguntas fuera del temario: proporción de preguntas fuera del temario en las que el sistema responde indebidamente.
- Tasa de abstención indebida en preguntas dentro del temario: proporción de preguntas dentro del temario en las que el sistema se abstiene de forma incorrecta.
- Tasa de abstención indebida en preguntas mixtas: proporción de preguntas mixtas en las que el sistema se abstiene sin cubrir la parte respondible.
- Tasa de abstención indebida en preguntas ambiguas: proporción de preguntas ambiguas en las que el sistema se abstiene sin intentar la respuesta.

La calibración del umbral se plantea como una **comparación metodológica de umbrales candidatos**: el valor previo del *baseline* (0,50) frente al valor 0,52 adoptado en la configuración final propuesta. La justificación empírica de este valor y su comportamiento frente a las cinco métricas anteriores se desarrollan en el Capítulo 7. El objetivo del diseño experimental es identificar un compromiso que minimice las falsas respuestas ante preguntas fuera del temario sin sobredimensionar la abstención sobre las preguntas legítimas.

6.8. Métricas operativas

Las métricas operativas cubren la latencia observada del flujo de consulta, incluyendo la media y los percentiles cuando estén disponibles, así como el coste computacional asociado a la ejecución local del prototipo, considerando las restricciones de un despliegue *self-hosted*. Estas métricas permiten discutir la viabilidad práctica del sistema en el entorno docente considerado y responden a la pregunta de investigación RQ2.

Los tiempos y el coste dependen del equipo concreto sobre el que se ejecuta el prototipo. En esta memoria se opta por una descripción sintética del entorno de ejecución en el Anexo A y por una interpretación prudente de las cifras: los resultados sostenidos por estas métricas quedan ligados al entorno local evaluado y no se presentan como garantías generales de rendimiento en otros despliegues.

6.9. Validación humana y jueces LLM

La estrategia de validación combina tres fuentes de evidencia complementarias: (i) métricas deterministas de recuperación, generación y abstención; (ii) jueces LLM ejecutados localmente sobre `Ollama` [22], en el prototipo se emplean dos modelos distintos como jueces, `mistral` y `qwen2.5:7b-instruct`, para reducir el sesgo derivado del uso de un único modelo, sin llamadas a `API` comerciales; y (iii) la revisión del autor sobre una muestra estratificada del *benchmark* canónico previo, formada por 49 preguntas, destinada a estimar el acuerdo entre los jueces LLM y el autor.

Sobre esa muestra se calculan estadísticos de acuerdo estándar, κ de Cohen y sus variantes ponderadas, error absoluto medio y sus intervalos de confianza, entre los dos jueces LLM y entre cada juez LLM y el autor. En el caso de κ de Cohen [38], el coeficiente expresa como:

$$\kappa = \frac{p_o - p_e}{1 - p_e},$$

donde p_o representa la proporción de acuerdos observados entre evaluadores y p_e la proporción de acuerdos que cabría esperar por azar dada la distribución de sus decisiones. Este segundo término se estima a partir de las distribuciones marginales de las categorías asignadas por cada evaluador:

$$p_e = \sum_{c \in C} p_{A,c} \cdot p_{B,c},$$

En esta memoria, κ se utiliza como indicador de acuerdo entre juez local y revisión del autor, no como sustituto de la evaluación determinista. La suficiencia de los jueces LLM locales como fuente auxiliar de evidencia queda acotada por un marco documentado que restringe su uso: se emplean como apoyo exploratorio, no como criterio único, y no sustituyen la verificación estructural determinista ni la revisión del autor. La discusión detallada del alcance real de los jueces LLM y de sus límites, particularmente en dimensiones ligadas a la citación material, se retoma en el Capítulo 8.

6.10. Protocolo experimental

El protocolo experimental fija el orden de ejecución y las condiciones de reproducibilidad de cada evaluación. Se articula en las siguientes etapas:

1. Fijar el *snapshot* del corpus como almacén vectorial de solo lectura, para garantizar que todas las configuraciones se evalúan sobre el mismo estado indexado.

2. Cargar el *benchmark* ampliado y su versión *chunk*-anotada, aplicando la política de subconjuntos descrita en §6.3.

3. Ejecutar la configuración que se va a evaluar contra la `API` local, con los parámetros de generación registrados y una gestión explícita de la caché de respuestas, sin asumir determinismo completo del modelo generador.

4. Guardar los *outputs* agregados por configuración, sin exponer contenido sensible del corpus.

5. Calcular las métricas deterministas de recuperación, generación estructural y abstención sobre los agregados.

6. Ejecutar la evaluación exploratoria local mediante `RAGAS` cuando la configuración lo requiera.

7. Ejecutar la evaluación de jueces LLM locales y consolidar los estadísticos de acuerdo con la revisión del autor sobre la muestra estratificada.

8. Revisar las advertencias y anomalías detectadas durante la ejecución y consolidar la configuración final que se comparará en el Capítulo 7.

Todas las etapas se ejecutan mediante *scripts* versionados en el repositorio, con *outputs* agregados que preservan la posibilidad de reproducción sin exponer contenido sensible del corpus. Debe hacerse explícita una acotación metodológica: el parámetro de fusión $k = 20$ y el umbral de abstención 0,52 se seleccionaron y caracterizaron sobre el mismo *benchmark* de desarrollo, sin una partición *holdout* independiente, por lo que los resultados describen ese conjunto y pueden contener optimismo de selección; esta evaluación no se presenta como validación externa.

La Figura 6.1 sintetiza el protocolo completo, desde las entradas congeladas hasta la triangulación de la evidencia y la selección de la configuración final.

El protocolo no introduce ninguna dependencia con `API` comerciales ni servicios en la nube externa: la totalidad de la inferencia, tanto la generación productiva como los jueces LLM, se ejecuta sobre modelos locales servidos por `Ollama`.

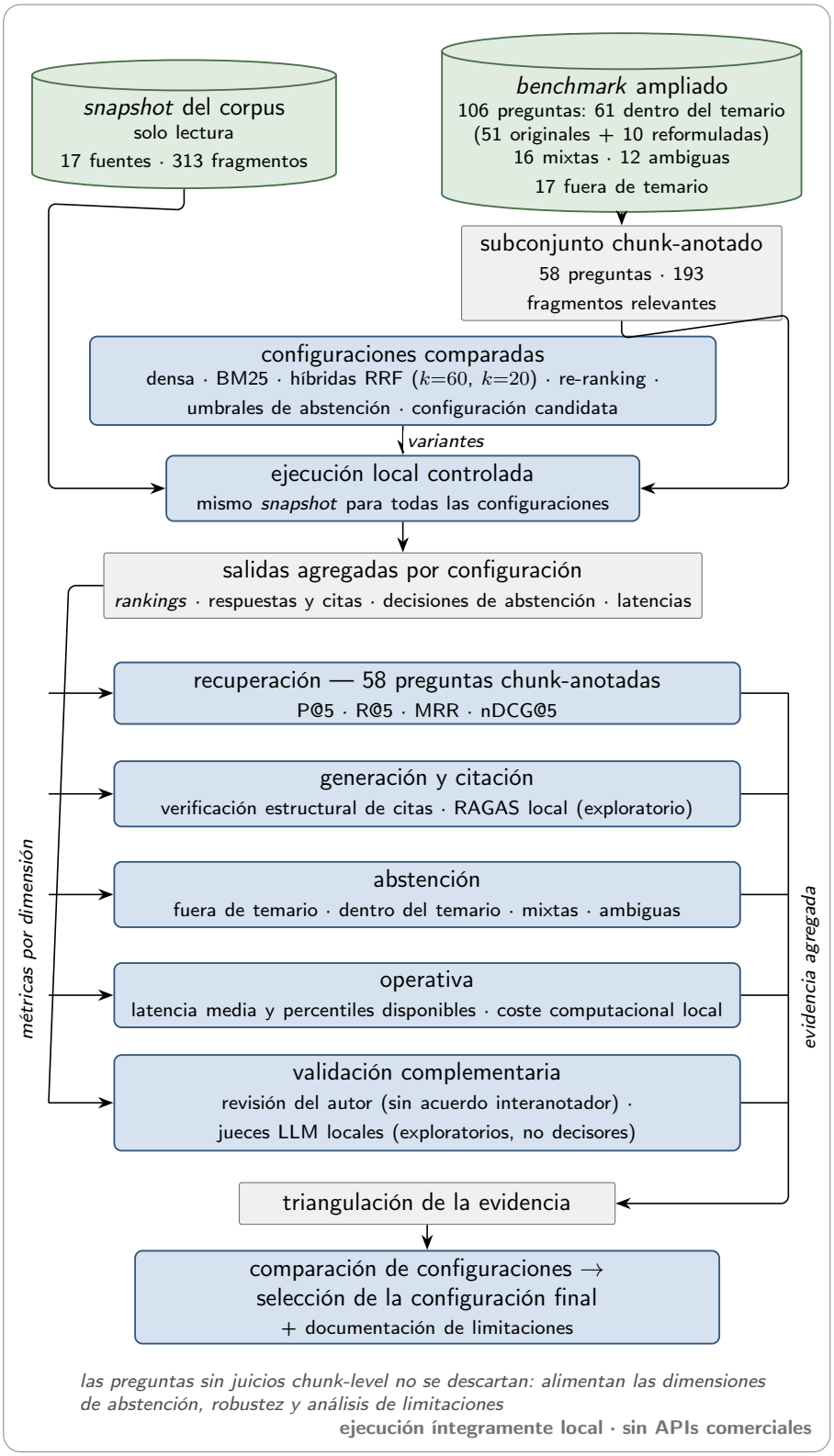

Figura 6.1: Flujo de datos del protocolo de evaluación experimental: entradas congeladas, ejecución local controlada, métricas por dimensión y triangulación de la evidencia para la selección de la configuración final. Fuente: elaboración propia.

7 Resultados experimentales

Este capítulo presenta los resultados obtenidos mediante el protocolo experimental definido en el capítulo anterior. En primer lugar, se comparan las configuraciones de recuperación y se analiza el efecto de la recuperación híbrida y del *re-ranking*. A continuación, se examinan los resultados de generación, citación y abstención, junto con la evaluación exploratoria mediante `RAGAS` y las métricas operativas de latencia y coste computacional. Finalmente, se establece la configuración definitiva del prototipo y se sintetizan los principales hallazgos que se interpretan posteriormente en el capítulo de discusión.

7.1. Resultados de recuperación

Las métricas de recuperación se calculan a nivel de fragmento sobre el subconjunto de 58 preguntas puntuables del *benchmark* ampliado descrito en §6.3. La Tabla 7.1 compara las cuatro configuraciones consideradas: recuperación densa pura, recuperación léxica mediante `BM25`, la variante híbrida inicial y la configuración híbrida final.

Tabla 7.1: Métricas de recuperación a nivel de fragmento sobre el subconjunto de 58 preguntas puntuables.

Configuración	P@5	R@5	MRR	nDCG@5	Lat. (ms)
Recuperación densa pura	0.328	0.434	0.478	0.417	20.1
Recuperación léxica BM25	0.390	0.621	**0.723**	**0.630**	14.3
Variante híbrida inicial	0.410	0.661	0.687	0.615	**12.5**
Configuración híbrida final	**0.421**	**0.681**	0.694	0.626	12.6

Las cifras están redondeadas para facilitar la lectura; los resultados extendidos con precisión ampliada se recogen en el Anexo F.

7.2. Impacto de la recuperación híbrida

En el corpus evaluado, la configuración híbrida final obtiene los mejores valores en *Precision@5* y *Recall@5* entre las cuatro configuraciones consideradas. Frente a la

recuperación densa pura, la mejora relativa es notable en las cuatro métricas principales: P@5 mejora un 28,3 %, R@5 un 56,9 %, MRR un 45,3 % y nDCG@5 un 50,1 %. Frente a la variante híbrida inicial, la mejora es más contenida y consistente: P@5 mejora un 2,7 %, R@5 un 3,1 %, MRR un 1,0 % y nDCG@5 un 1,8 %, sin coste apreciable en latencia media.

Como matiz honesto, la recuperación léxica mediante `BM25` sigue liderando el MRR y el nDCG@5 con un margen moderado sobre la configuración híbrida final, un 4,1 % en MRR y un 0,7 % en nDCG@5. Esta tensión es interpretable: `BM25` acierta pronto cuando la coincidencia léxica es exacta, mientras que la configuración híbrida final recupera más evidencia útil dentro del *top-5* completo. Dado que el modelo generador recibe el *top-k* completo y no únicamente la primera posición, la memoria prioriza P@5 y R@5 como criterios para elegir la configuración final. Los resultados observados sugieren, por tanto, una preferencia por la recuperación híbrida final en el marco de uso previsto, sin que ello invalide a `BM25` como referencia sólida en escenarios de coincidencia léxica predominante.

7.3. Impacto del *re-ranking*

El *re-ranking* se conserva en el prototipo como etapa opcional. La implementación integra un *cross-encoder* local basado en `Qwen3-Reranker-0.6B`, cuya selección se realizó durante una fase de ablación técnica previa a la evaluación sobre el *benchmark* ampliado con anotación a nivel de fragmento. El contraste cerrado sobre este *benchmark*, con el mismo conjunto de candidatos para todas las configuraciones, no muestra una mejora concluyente: la diferencia pareada en nDCG@5 es de $-0{,}027$, con un intervalo de confianza del 95 % de $[-0{,}135; +0{,}081]$, y la etapa multiplica la latencia de recuperación por aproximadamente 371. Este factor temporal se calcula dentro del contraste confirmatorio, comparando los 8,657 s del *reranker* con los 23,3 ms de su control homogéneo, en el que ambas variantes parten de veinte candidatos; la diferencia respecto a la latencia de la Tabla 7.1 no indica que el recuperador final se volviera peor o más lento, sino que se volvió a medir bajo un protocolo distinto, diseñado específicamente para aislar el coste del *reranking*.

En el corpus evaluado, esta evidencia no justifica la activación por defecto del *re-ranking* en términos de coste y beneficio, y la configuración final del prototipo lo mantiene desactivado. Esta conclusión no equivale a afirmar que el *re-ranking* no aporte valor: sugiere que, sobre un corpus reducido y temáticamente coherente como el de un curso individual, la ganancia esperable no compensa el coste computacional adicional. Corpus mayores o con mayor densidad temática podrían inclinar el balance; esa exploración queda

como línea futura y se retoma en el Capítulo 9.

7.4. Resultados de generación y citación

La calidad de la generación se analiza en dos planos complementarios, consistentes con la separación establecida en §6.6. En el plano determinista, la verificación estructural de las citas comprueba que los marcadores utilizados por la respuesta se correspondan de forma posicional con los fragmentos efectivamente incorporados al contexto. Esta comprobación es reproducible y detecta consistentemente las cuatro categorías descritas en el Capítulo 5: citas ausentes, citas no reconocidas, citas malformadas y citas duplicadas. En el corpus evaluado, la política de *prompt* y el *guard* estructural frente a fugas de plantilla contribuyen a mantener nula la tasa de fugas observadas. En la caracterización final se procesaron 106 consultas: 80 generaron respuesta y 26 activaron la abstención, sin errores; las 80 respuestas incluyeron fuentes y 79 incorporaron al menos una cita, aunque solo 18 superaron la verificación estructural estricta, 61 presentaron marcadores duplicados y 78 quedaron libres de incidencias graves al ignorar exclusivamente esas duplicaciones; no se observaron fugas de *prompt* (0 de 80).

Conviene subrayar dos límites reales. Primero, la verificación estructural no equivale a una validación semántica completa: no asegura que cada afirmación de la respuesta esté materialmente soportada por el pasaje citado. Segundo, los resultados detallados de los jueces LLM locales y de la revisión del autor sobre la muestra estratificada del *benchmark* canónico se interpretan bajo el marco de suficiencia introducido en el Capítulo 6 y se discuten en el Capítulo 8, evitando que este capítulo se convierta en un metaanálisis del juez.

7.5. Resultados de abstención

La calibración del umbral de abstención se aborda como un problema empírico. Sobre el *benchmark* ampliado de 106 preguntas se comparan varios umbrales candidatos de forma determinista, sin invocar al modelo generador, aprovechando que la decisión de abstención depende únicamente de la puntuación de recuperación. La Tabla 7.2 resume el comportamiento para cinco valores representativos.

En el *benchmark* evaluado, el paso del umbral previo, 0,50, al umbral final, 0,52, reduce a cero la tasa de respuesta indebida en preguntas fuera del temario y eleva al 100 % la tasa de abstención correcta sobre ese subconjunto. Las tasas de abstención indebida

Tabla 7.2: Métricas de abstención sobre el *benchmark* ampliado de 106 preguntas para cinco umbrales representativos. Abreviaturas: OOS = fuera del temario; IS = dentro del temario; Mix = mixtas; Amb = ambiguas.

Umbral	Abst. OOS	Resp. OOS	Abst. IS	Abst. Mix	Abst. Amb
0.42	70.6 %	29.4 %	0.0 %	0.0 %	8.3 %
0.45	94.1 %	5.9 %	0.0 %	0.0 %	16.7 %
0.50 (previo)	94.1 %	5.9 %	1.6 %	12.5 %	50.0 %
0.52 (final)	**100.0 %**	**0.0 %**	**1.6 %**	**12.5 %**	**50.0 %**
0.55	100.0 %	0.0 %	1.6 %	31.2 %	58.3 %

sobre preguntas dentro del temario, 1,6 %, mixtas, 12,5 %, y ambiguas, 50,0 %, se mantienen respecto al umbral previo; umbrales superiores como 0,55 empeoran las tasas de preguntas mixtas, 31,2 %, y ambiguas, 58,3 %, sin ganancia en las restantes columnas. Las preguntas ambiguas siguen siendo el caso más difícil del prototipo y se identifican como el foco natural de las políticas de clarificación. Este resultado se presenta como observación empírica dentro del corpus y del *benchmark* evaluados, no como garantía general.

Complementariamente, la política de *answerability* permanece implementada pero desactivada por defecto en la configuración final; no se ha reevaluado bajo esta configuración y se conserva como opción *opt-in* y como línea futura de evaluación controlada.

7.6. Evaluación local exploratoria mediante `RAGAS`

Como triangulación adicional, el trabajo incorpora una evaluación exploratoria con el marco `RAGAS` ejecutado localmente sobre modelos servidos por `Ollama`, sin llamadas a `API` comerciales. La Tabla 7.3 resume los valores observados para las tres dimensiones consideradas.

Tabla 7.3: Evaluación exploratoria RAGAS local. Dimensiones: fidelidad al contexto, relevancia de la respuesta y precisión del contexto. N común por métrica: 44 en fidelidad y 58 en relevancia y precisión. La media es descriptiva y no estandarizada.

Configuración	n	Fidel.	Relev.	Prec.	Media
Recuperación densa pura	44–58	0.905	0.399	0.983	0.762
Recuperación léxica BM25	44–58	0.890	0.323	0.962	0.725
Configuración híbrida final	44–58	0.840	0.328	0.953	0.707

Bajo cobertura homogénea entre configuraciones, ninguna comparación pareada muestra diferencias estadísticamente concluyentes, y la ventaja previa de la variante híbrida, obtenida en una ejecución con cobertura parcial ($n = 19$), no se reproduce. Conviene, sin embargo, subrayar dos matices. En primer lugar, estos valores se reportan únicamente como triangulación exploratoria: no fundamentan por sí solos la decisión de configuración

final. En segundo lugar, la cobertura del contexto no se calcula dentro de `RAGAS` en este trabajo por la razón ya expuesta en el Capítulo 6: los fragmentos relevantes se almacenan como identificadores y rellenar el campo de referencia con el texto completo aumentaría la exposición del contenido del corpus, en tensión con el enfoque *privacy-first* del prototipo.

7.7. Latencia y coste computacional

Las latencias medias de la fase de recuperación por configuración son las ya recogidas en la Tabla 7.1: 20,1 ms para la recuperación densa pura, 14,3 ms para `BM25`, 12,5 ms para la variante híbrida inicial y 12,6 ms para la configuración híbrida final. Estos valores se refieren exclusivamente a la fase de recuperación y al entorno de ejecución evaluado, e incluyen la media y los percentiles cuando el instrumento los registra. No deben interpretarse como garantías generales de rendimiento en otros despliegues.

El coste computacional del ciclo completo de consulta depende del equipo local sobre el que se ejecuta el prototipo, del modelo servido por `Ollama` y de los parámetros de generación. Su descripción detallada se remite al Anexo A para preservar la compacidad del cuerpo y evitar comprometer más de lo que la instrumentación efectivamente emite.

7.8. Configuración final del prototipo

La configuración final adoptada como resultado de la evaluación combina las siguientes decisiones, cada una justificada en las secciones anteriores del capítulo:

- `RAG_RETRIEVAL_MODE=hybrid`: recuperación híbrida como modo por defecto (§7.2).
- `RAG_HYBRID_RRF_K=20`: parámetro RRF fijado en 20 tras el ajuste sobre el *benchmark* ampliado (§7.2).
- `RAG_ABSTENTION_MIN_SCORE=0.52`: umbral de abstención tras la comparación con el umbral previo (§7.5).
- `RAG_RERANK_ENABLED=false`: *re-ranking* desactivado por defecto en el corpus evaluado, disponible como opción *opt-in* (§7.3).
- `RAG_ANSWERABILITY_POLICY_ENABLED=false`: política de *answerability* desactivada por defecto, conservada como línea futura (§7.5).

- `PromptPolicy` activa, con perfiles de alumnado y profesorado y con el rol de administración tratado como profesorado a efectos de generación (§7.4).

Las implicaciones de esta configuración para las preguntas de investigación se retoman en el Capítulo 8.

7.9. Resumen de resultados

Tres observaciones sintetizan los resultados del capítulo. En primer lugar, la configuración híbrida final mejora *Precision@5* y *Recall@5* sobre los *baselines* densos y léxicos en el corpus evaluado, mientras que `BM25` mantiene una ventaja moderada en MRR y nDCG@5 asociada al patrón de coincidencia léxica exacta. En segundo lugar, el umbral de abstención 0,52 reduce a cero la tasa de respuesta indebida en preguntas fuera del temario en el *benchmark* evaluado y mantiene las tasas de abstención indebida sobre preguntas dentro del temario y sobre preguntas mixtas, con el coste conocido sobre preguntas ambiguas. En tercer lugar, la evaluación local mediante `RAGAS` y los jueces LLM locales se utilizan como triangulación y apoyo exploratorio, no como sustituto de las métricas deterministas ni de la revisión del autor. La discusión interpretativa de estas observaciones frente a las preguntas de investigación se aborda en el Capítulo 8.

8 Discusión

Este capítulo interpreta los resultados experimentales y los relaciona con las preguntas de investigación formuladas al inicio del trabajo. En primer lugar, se analiza la contribución de los distintos componentes del sistema y el compromiso existente entre calidad, privacidad y latencia. A continuación, se examinan los riesgos educativos, los límites de uso y las principales amenazas a la validez del estudio. Finalmente, se delimita qué elementos de la arquitectura pueden transferirse a otros cursos y qué parámetros y resultados requieren una nueva calibración en cada contexto de aplicación.

8.1. Respuesta interpretativa a las preguntas de investigación

Los resultados presentados en el Capítulo 7 permiten interpretar las cuatro preguntas de investigación formuladas en el Capítulo 3.

La primera pregunta (RQ1) encuentra una respuesta matizada en los resultados de recuperación. En el corpus evaluado, la configuración híbrida final mejora la *Precision@5* y la *Recall@5* sobre los *baselines* densos y léxicos, pero esa mejora no es universal ni monótona: `BM25` puro conserva una ventaja moderada en MRR y nDCG@5. La tensión admite una lectura coherente: `BM25` acierta pronto cuando la coincidencia léxica es exacta, mientras que la recuperación híbrida aporta mejor cobertura de evidencia útil dentro del *top-5*. Como el modelo generador recibe el *top-k* completo y no únicamente la primera posición, la elección de la configuración por defecto prioriza esa cobertura, sin afirmar superioridad universal. Debe añadirse que la evidencia principal es de recuperación: el trabajo no permite concluir que la arquitectura completa mejore la calidad de las respuestas frente a todos los *baselines*.

La respuesta a RQ2 es, de forma deliberada, parcial. El prototipo funciona íntegramente en local, sin llamadas a `API` comerciales, y sus tiempos quedaron caracterizados en el equipo evaluado: la recuperación se resuelve en decenas de milisegundos, pero la generación domina el ciclo completo, con una media de 13,29 s por respuesta generada y

un percentil 95 de generación de 29,25 s. La privacidad se apoya en propiedades arquitectónicas, ejecución local y ausencia de servicios externos, no en una métrica experimental. Quedan fuera de la evidencia disponible la utilidad percibida por el alumnado, que no se evaluó con usuarios reales, y el comportamiento bajo concurrencia institucional; la viabilidad observada describe el entorno evaluado y solo podría trasladarse a otros despliegues previa comprobación en su configuración concreta.

Para RQ3, la evidencia disponible no sostiene una jerarquía absoluta de valor incremental, porque no todos los componentes cuentan con una ablación individual. Sí permite distinguir grados. La recuperación híbrida dispone de evidencia comparativa directa frente a las alternativas densa y léxica. El umbral de abstención cuenta con una calibración observada sobre el *benchmark* de desarrollo. El *re-ranking* fue objeto de una ablación confirmatoria que no mostró mejora concluyente y multiplicó el coste de la fase de recuperación, lo que justifica mantenerlo desactivado. La política de *prompt* está activa como decisión de diseño de adecuación por rol, sin una ablación cuantitativa que permita atribuirle una mejora medida. Y la política de *answerability* está implementada, desactivada por defecto y pendiente de una ablación controlada bajo la configuración final.

RQ4 interroga los límites de la evaluación asistida. La triangulación entre métricas deterministas, `RAGAS` local y jueces LLM locales muestra que estas fuentes pueden aportar una señal complementaria, aunque con restricciones importantes: el acuerdo de los jueces con el criterio humano fue limitado en las dimensiones ligadas a la citación material, la cobertura de `RAGAS` es homogénea entre configuraciones pero con un N común reducido en fidelidad (44 de 58), y la verificación estructural de citas no equivale a un soporte semántico completo. La convergencia observada entre los dos jueces tampoco implica corrección, porque ambos pueden compartir un mismo error sistemático. Ninguna de estas fuentes sustituye a las métricas deterministas ni a la revisión humana.

8.2. Componentes del *pipeline* que aportan más valor

Más que una jerarquía cerrada de componentes, los resultados permiten ordenar la discusión por la fuerza de la evidencia disponible. Con evidencia comparativa directa se sitúa la recuperación híbrida, que mejora *Precision@5* y *Recall@5* frente a los *baselines* densos y léxicos en el corpus evaluado. Con caracterización operativa, la abstención por umbral: el valor 0,52 elimina las respuestas indebidas fuera del temario en el *benchmark* de desarrollo con un coste contenido dentro del temario, si bien esa protección es observacional y queda ligada al conjunto sobre el que se calibró.

Un tercer grupo reúne los componentes implementados pero no validados de forma

aislada. La política de *prompt*, activa por defecto, condiciona el estilo y la adecuación por rol de las respuestas; es una decisión de diseño razonada, no una mejora cuantificada, porque carece de ablación propia. La política de *answerability* permanece desactivada y pendiente de reevaluación bajo la configuración final. El *re-ranking*, por último, ilustra el valor de un resultado negativo bien medido: una técnica más compleja, con un *cross-encoder* local, no mejoró de forma concluyente ninguna métrica de recuperación y multiplicó la latencia de esa fase por aproximadamente 371, lo que llevó a descartarla como opción por defecto sin negar su posible utilidad en otros contextos.

8.3. Compromiso entre calidad, privacidad y latencia

La arquitectura reduce la exposición de datos al mantener la recuperación, la generación y la evaluación dentro del entorno local evaluado, aunque su robustez operativa dependería de la configuración institucional concreta. La decisión de no utilizar `API` comerciales limita el espacio de modelos accesibles al prototipo y, con ello, impone un techo a algunas dimensiones de calidad, particularmente a la calidad del modelo generador y de los jueces LLM disponibles, a cambio de mantener bajo control la circulación del corpus, de las consultas y de las respuestas.

Sobre la latencia conviene separar tres planos que responden a protocolos de medición distintos. La fase de recuperación se resuelve en decenas de milisegundos en el equipo evaluado. El *re-ranking*, medido en su contraste confirmatorio, eleva esa fase hasta 8,657 s de media, unas 371 veces su control homogéneo. Y el ciclo completo de una respuesta generada está dominado por la generación: 13,29 s de media, con un percentil 95 de generación de 29,25 s en el *hardware* evaluado. Estas cifras describen el entorno de evaluación y no una garantía general, porque dependen del equipo, del modelo servido por `Ollama`, de la concurrencia y del tamaño del corpus; el tiempo total observado puede resultar, además, una limitación para un uso interactivo fluido. La reducción de dependencias externas que aporta el enfoque *privacy-first* acota la superficie de riesgo operativo, pero no la suprime, y no constituye por sí sola una prueba de cumplimiento institucional completo.

8.4. Riesgos educativos y límites de uso

El uso del sistema en aulas reales conlleva riesgos educativos que la memoria debe declarar de forma explícita. El asistente se concibe como apoyo al estudio, no como sustituto de la interacción docente ni de la lectura crítica del alumnado, y no debe em-

plearse como sistema de calificación automática. El sistema puede fallar si el corpus está incompleto o mal estructurado, o si el material docente contiene errores o ambigüedades. En tal caso, la respuesta del asistente puede propagar esos errores dentro del contexto que recupera. La existencia de citas verificadas estructuralmente no equivale al soporte semántico de cada afirmación, por lo que el alumnado debe leer las citas críticamente y contrastar la respuesta con los materiales originales.

Del lado del sistema, la abstención ante evidencia insuficiente y la verificación estructural de citas actúan como mecanismos de mitigación de estos riesgos, no como garantías absolutas: reducen la probabilidad de respuesta indebida, no la eliminan. Debe subrayarse, además, que el trabajo no incluye un estudio con alumnado ni profesorado; no se midieron aprendizaje, satisfacción, utilidad percibida ni efectos de dependencia. Las recomendaciones que siguen son, por tanto, condiciones prudentes de uso y no resultados experimentales: acompañar el despliegue con supervisión del profesorado, ofrecer al alumnado una formación mínima sobre la lectura crítica de respuestas generadas y comunicar de forma explícita el alcance y los límites del sistema, entendido como apoyo documental supervisado y no como evaluador ni decisor académico.

8.5. Amenazas a la validez

Las amenazas a la validez declaradas en el Capítulo 3 y en el Capítulo 6 adquieren aquí un significado concreto a la luz de la evidencia observada. La validez externa está limitada por el propio escenario: un único curso del dominio MLOps, 313 fragmentos indexados y ninguna evaluación sobre otros cursos. Los valores numéricos obtenidos describen ese corpus concreto y no se transfieren automáticamente a otros dominios ni escalas.

La validez de constructo descansa sobre 58 preguntas puntuables con anotación *chunk-level* realizada por el autor, sin estimación de acuerdo interanotador: las comparaciones se apoyan en una base reducida y en un único criterio de anotación, de modo que cualquier sesgo de ese criterio afecta por igual a todas las configuraciones comparadas, pero no puede detectarse desde dentro del propio estudio. En el plano interno, los jueces LLM locales presentan un acuerdo limitado con el criterio humano en las dimensiones ligadas a la citación material: ninguna dimensión juez–humano alcanzó el umbral de acuerdo moderado adoptado en este trabajo y la convergencia entre jueces no se trasladó al acuerdo con el autor. La cobertura de `RAGAS` es homogénea entre configuraciones, aunque con un N común reducido en fidelidad (44 de 58), y la verificación de citas es estructural, no semántica. La evidencia sobre citación resulta, en consecuencia, más débil que la relativa a recuperación y abstención.

Respecto a la validez de conclusión, las latencias observadas dependen del entorno concreto de ejecución y no constituyen una garantía general de rendimiento; $k = 20$ y el umbral 0,52 se seleccionaron y caracterizaron sobre el mismo *benchmark* de desarrollo, sin partición *holdout* independiente, por lo que las cifras asociadas pueden contener optimismo de selección; y las preguntas fueron construidas por el autor con conocimiento del corpus, un sesgo inherente al proceso de construcción del *benchmark*. A ello se suman las limitaciones de ingesta, como la ausencia de `OCR` local y el aprovechamiento limitado del contenido visual y de los formatos irregulares, que acotan la calidad de la evidencia sobre la que se calculan las métricas: el sistema evalúa lo que efectivamente puede recuperar.

8.6. Generalización a otros cursos

La generalización del trabajo a otros cursos exige distinguir con claridad qué es transferible y qué requiere recalibración. Resulta plausiblemente transferible la arquitectura general, con la `API REST` de `Moodle` para la ingesta documental, `FastAPI` y su ruta `/ask` como contrato `HTTP` de consulta, el bloque de `Moodle` desarrollado en `PHP` como interfaz de usuario integrada y el motor RAG desarrollado en `Python` como núcleo, con `Ollama` como servidor de inferencia y `ChromaDB` como persistencia vectorial. También son transferibles la separación de responsabilidades entre el LMS y el motor RAG, el *pipeline* multiformato, el modelo de metadatos educativos, el marco de evaluación por recuperación, abstención y triangulación, y el enfoque *privacy-first* que orienta las decisiones de despliegue.

En cambio, requieren recalibración o nueva validación en cada contexto el parámetro de fusión $k = 20$ y el umbral de abstención 0,52; el modelo de *embeddings* y la estrategia de fragmentación; el modelo generador servido por `Ollama`; los valores concretos de *Precision@5*, *Recall@5*, MRR y nDCG@5, junto con el equilibrio observado entre `BM25` y la recuperación híbrida; la utilidad relativa del *re-ranking*; el comportamiento ante preguntas ambiguas; las latencias; el tratamiento de permisos y metadatos en otros cursos; y, en último término, la utilidad educativa, que no se midió con usuarios. La arquitectura ofrece una transferibilidad plausible; los resultados numéricos no viajan con ella. El capítulo siguiente recoge las conclusiones que se derivan de este balance y las líneas de trabajo que deja abiertas.

IV

Parte

Cierre

9 Conclusiones y trabajo futuro

Este capítulo sintetiza los resultados del trabajo y valora el grado de cumplimiento de los objetivos y las preguntas de investigación planteadas. En primer lugar, se resumen las principales contribuciones y se delimitan las conclusiones que pueden sostenerse dentro del alcance experimental. A continuación, se presentan las limitaciones del prototipo y de su evaluación. Finalmente, se proponen las líneas de trabajo futuro necesarias para ampliar la integración con `Moodle`, validar el sistema con usuarios reales, mejorar el tratamiento documental y estudiar su posible escalado institucional.

9.1. Cumplimiento de objetivos

Los objetivos específicos formulados en el Capítulo 3 se consideran cubiertos dentro del alcance experimental definido, aunque con distintos grados de validación. La revisión y sistematización del estado del arte (OE1) se ha cerrado con la síntesis del Capítulo 2 y sostiene el resto del trabajo. La definición de una arquitectura modular *privacy-first* (OE2) se ha materializado en el diseño documentado en el Capítulo 4, con separación explícita de responsabilidades entre el LMS y el motor RAG. La implementación (OE3) se ha alcanzado como **prototipo funcional**, no como producto institucional completo: el Capítulo 5 describe su realización efectiva sobre el corpus del curso considerado.

La construcción del conjunto experimental (OE4) se ha alcanzado mediante el *benchmark* ampliado y la anotación *chunk-level* descritos en el Capítulo 6, con la limitación honesta de que la anotación fue realizada por el autor y, por tanto, no se estima acuerdo interanotador. El protocolo experimental (OE5) queda alcanzado en las dimensiones de recuperación, abstención y triangulación exploratoria, y con cautela explícita en las dimensiones ligadas a la citación material y a los jueces LLM locales, tal como se discute en el Capítulo 8. Por último, la documentación de limitaciones y líneas futuras (OE6) queda cerrada en este mismo capítulo, en el que se enuncian los límites reales del prototipo y las evoluciones razonables.

9.2. Respuesta final a las preguntas de investigación

En síntesis, la respuesta final a las cuatro preguntas de investigación es la siguiente. Respecto a **RQ1**, en el corpus evaluado la configuración híbrida final mejora la *Precision@5* y la *Recall@5* sobre las configuraciones densas y léxicas aisladas; `BM25` conserva una ventaja moderada en MRR y nDCG@5, por lo que la elección de la configuración híbrida se justifica por su cobertura de evidencia útil en el *top-5* y no por una superioridad universal.

Respecto a **RQ2**, la evidencia experimental respalda la viabilidad de la ejecución local *privacy-first* en el entorno experimental evaluado, con la cautela de que la latencia observada corresponde a la fase de recuperación y no equivale a la latencia completa del ciclo con generación. La generalización a otros despliegues dependería del *hardware* disponible, del modelo servido por `Ollama`, del nivel de concurrencia y del tamaño del corpus indexado. Respecto a **RQ3**, los componentes con mayor valor incremental observado son la recuperación híbrida y la calibración del umbral de abstención; la política de *prompt* aporta valor pedagógico y de adecuación de la respuesta; la política de *answerability* y el *reranking* quedan implementados, pero desactivados por defecto en la configuración final, este último por no justificarse en términos de coste y beneficio en el corpus considerado. Respecto a **RQ4**, la evaluación exploratoria con `RAGAS` y con jueces LLM locales aporta triangulación adicional, pero no sustituye a las métricas deterministas ni a la revisión del autor, especialmente en las dimensiones ligadas al soporte semántico de las respuestas.

9.3. Contribuciones principales

Las contribuciones principales del trabajo, sostenidas en los capítulos correspondientes de la memoria, son las siguientes: (i) una arquitectura RAG modular y *privacy-first* integrada con `Moodle` mediante interfaces oficiales, con separación clara entre la `API REST` de `Moodle` para la ingesta documental, `FastAPI` y su ruta `/ask` para la consulta, un bloque de `Moodle` desarrollado en `PHP` como interfaz de usuario y el motor RAG desarrollado en `Python` como núcleo; (ii) un prototipo funcional multiformato con instrumentación reproducible para la evaluación experimental; (iii) la implementación de dos bloques funcionales de `Moodle` desarrollados en `PHP` dentro del alcance del prototipo, un bloque de *chat* RAG orientado a la consulta integrada desde el curso y un bloque de indexación orientado a lanzar la ingesta documental desde `Moodle`, que actúan como interfaces finas sobre `FastAPI` y mantienen la lógica de recuperación, generación, citación, abstención y evaluación en el motor RAG implementado en `Python`; (iv) un *benchmark* ampliado

de 106 preguntas sobre material docente real, con anotación a nivel de fragmento y 58 preguntas puntuables para las métricas de recuperación; (v) un marco de evaluación multifase que contrasta métricas deterministas, evaluación exploratoria local mediante `RAGAS`, jueces LLM ejecutados localmente y revisión del autor; y (vi) una discusión honesta de los límites de la evaluación asistida por LLM locales bajo restricciones *privacy-first*, con la documentación reproducible que sostiene la trazabilidad de las decisiones y de sus límites.

9.4. Limitaciones

Las limitaciones del trabajo se declaran de forma sintética; su interpretación se desarrolla en el Capítulo 8. La evaluación se ejecuta sobre un único curso del dominio `MLOps`. El *benchmark* contiene 106 preguntas totales, de las cuales 58 son puntuables para las métricas de recuperación. La anotación *chunk-level* fue realizada por el autor, por lo que no se estima acuerdo interanotador. No se ha realizado una evaluación con usuarios reales ni una evaluación sobre varios cursos. El *pipeline* no integra `OCR` local, por lo que el contenido puramente visual queda aprovechado de forma limitada. La evaluación local mediante `RAGAS` no distingue estadísticamente entre configuraciones bajo cobertura homogénea y actúa como triangulación exploratoria. Los jueces LLM locales se emplean como apoyo exploratorio y no como criterio decisor. La verificación estructural de citas no equivale al soporte semántico completo de cada afirmación. Las latencias observadas dependen del entorno concreto de ejecución y no constituyen garantías generales de rendimiento.

9.5. Trabajo futuro

Las líneas de trabajo futuro derivadas del análisis anterior se organizan en cinco bloques.

9.5.1. *Plugin* institucional completo para `Moodle`

El siguiente paso natural del trabajo consiste en madurar la integración con `Moodle` desde los bloques actuales hacia un *plugin* institucional completo, con soporte de instalación por administradores, gestión de permisos alineada con las capacidades de `Moodle`, superficies de configuración accesibles desde la interfaz de administración del LMS y mantenimiento sostenido. Esta evolución debe preservar la línea metodológica del prototipo: el motor RAG no debe implementarse en `PHP`, sino que debe permanecer como núcleo externo al LMS y consumirse mediante contratos `HTTP` estables.

9.5.2. Evaluación continua con usuarios reales

La validación con usuarios reales en un curso operativo constituye una extensión imprescindible antes de una implantación institucional. Se propone un diseño de estudio dirigido al alumnado y al profesorado que combine métricas objetivas de uso con dimensiones cualitativas de utilidad percibida, carga docente, confianza en las respuestas y trazabilidad de las citas. El estudio debe respetar las restricciones éticas y las garantías de privacidad que impone el entorno universitario.

9.5.3. *Chunking* estructural y `OCR` local

Las limitaciones observadas en materiales visualmente ricos, en presentaciones y en archivos `PDF` con estructura irregular sugieren dos líneas técnicas concretas: un *chunking* estructural más fino, sensible al tipo de documento, y la incorporación de `OCR` ejecutado localmente para textos embebidos en imágenes o diagramas. Ambas líneas se pueden abordar sin comprometer las restricciones *privacy-first* que orientan el trabajo.

9.5.4. `GraphRAG`, *Agentic RAG* y *query rewriting*

En el plano de investigación, el trabajo abre la puerta a explorar variantes avanzadas de RAG como línea futura, sin presentarlas como una necesidad inmediata: `GraphRAG` con grafos de conocimiento extraídos del corpus, arquitecturas de *Agentic RAG* para consultas de varios pasos y estrategias de *query rewriting* y clarificación controlada para preguntas ambiguas o mixtas. Estas extensiones deben integrarse manteniendo la trazabilidad documental como invariante y evitando sobredimensionar la arquitectura.

9.5.5. Escalado institucional con `Qdrant` y monitorización

La evolución del prototipo hacia un servicio institucional requiere sustituir la base vectorial actual, `ChromaDB`, adecuada para el prototipo, por una solución escalable, como `Qdrant` *self-hosted*, entre otras, e incorporar monitorización operativa, con métricas, trazas y alertas, que permita detectar degradaciones tempranas y recalibrar la política de abstención al comportamiento real del sistema en un despliegue institucional. En escenarios institucionales, el escalado debería dimensionarse a partir de la concurrencia efectiva, el tamaño del modelo, la longitud media de las respuestas, la política de *batching*, la latencia objetivo y el tamaño del corpus, no únicamente a partir del número total de usuarios registrados.

La opción más alineada con el enfoque *privacy-first* sería una infraestructura local con unidades de procesamiento gráfico (*Graphics Processing Units*, GPU) gestionada por la institución. Como alternativa condicionada, podría estudiarse el uso de GPU en una infraestructura *cloud* privada o dedicada, siempre que se validen previamente la región de tratamiento, los acuerdos de protección de datos, el aislamiento de red, el cifrado, la política de *logs*, la gestión de secretos, las copias de seguridad (*backups*) y el impacto sobre la privacidad. Este escalado debe preservar el enfoque *privacy-first* que orienta el trabajo.

Bibliografía

[1] E. Kasneci et al., "ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education", *Learning and Individual Differences*, vol. 103, pág. 102 274, 2023. DOI: 10.1016/j.lindif.2023.102274.

[2] F. Miao y W. Holmes, "Guidance for Generative AI in Education and Research", UNESCO, inf. téc., 2023.

[3] Y. Shi, K. Yu, Y. Dong y F. Chen, "Large language models in education: a systematic review of empirical applications, benefits, and challenges", *Computers and Education: Artificial Intelligence*, vol. 10, pág. 100 529, 2026, Open access (CC BY 4.0). Available online 13 December 2025., ISSN: 2666-920X. DOI: 10.1016/j.caeai.2025.100529.

[4] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", en *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv: 2005.11401 [cs.CL].

[5] Y. Gao et al., *Retrieval-Augmented Generation for Large Language Models: A Survey*, 2023. arXiv: 2312.10997 [cs.CL].

[6] European Union, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data (General Data Protection Regulation)*, Official Journal of the European Union, L 119, 2016.

[7] K. Bourne, *Unlocking Data with Generative AI and RAG: Enhance generative AI systems by integrating internal data with large language models*, First. Birmingham, UK: Packt Publishing, sep. de 2024, ISBN: 978-1-83588-790-5.

[8] D. Dhyani, *RAG with Python Cookbook: Learn principles of RAG with LLM and agentic AI, with 120+ recipes*, First. India: BPB Publications, 2026, ISBN: 978-93-65895-735.

[9] B. Chen y T. Taipalus, "Metadata-aware RAG for Enforcing Access Control and Metadata-based Filtering: Proof of Concept and Evaluation", en *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, ép. RACS '25, Association for Computing Machinery, 2025, págs. 1-7. DOI: 10.1145/3769002.3769952.

[10] V. Karpukhin et al., "Dense Passage Retrieval for Open-Domain Question Answering", en *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020. arXiv: 2004.04906 [cs.CL].

[11] J. Devlin, M.-W. Chang, K. Lee y K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", en *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, 2019, págs. 4171-4186. DOI: 10.18653/v1/N19-1423.

[12] C. D. Manning, P. Raghavan y H. Schütze, *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2008.

[13] Y. Gao, Y. Xiong, M. Wang y H. Wang, *Modular RAG: Transforming RAG Systems into LEGO-like Reconfigurable Frameworks*, 2024. arXiv: 2407.21059 [cs.CL].

[14] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu y M. Gatford, "Okapi at TREC-3", en *Overview of the Third Text REtrieval Conference (TREC-3)*, D. K. Harman, ed., ép. NIST Special Publication 500-225, Gaithersburg, MD, USA: National Institute of Standards and Technology (NIST), 1995.

[15] L. Gao, X. Ma, J. Lin y J. Callan, *Precise Zero-Shot Dense Retrieval without Relevance Labels*, 2022. arXiv: 2212.10496 [cs.IR].

[16] O. Khattab y M. Zaharia, "ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT", en *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2020. arXiv: 2004.12832 [cs.IR].

[17] D. Edge et al., *From Local to Global: A GraphRAG Approach to Query-Focused Summarization*, Preprint. Under review., 2024. arXiv: 2404.16130 [cs.CL].

[18] E. M. Voorhees y D. M. Tice, "The TREC-8 Question Answering Track Evaluation", en *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, MD, USA: National Institute of Standards and Technology, 1999.

[19] K. Järvelin y J. Kekäläinen, "Cumulated Gain-Based Evaluation of IR Techniques", *ACM Transactions on Information Systems*, vol. 20, n.º 4, págs. 422-446, 2002. DOI: 10.1145/582415.582418.

[20] S. Es, J. James, L. Espinosa-Anke y S. Schockaert, *RAGAS: Automated Evaluation of Retrieval Augmented Generation*, 2023. arXiv: 2309.15217 [cs.CL].

[21] J. Saad-Falcon, O. Khattab, C. Potts y M. Zaharia, *ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems*, 2023. arXiv: 2311.09476 [cs.CL].

[22] L. Zheng et al., *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*, 2023. arXiv: 2306.05685 [cs.CL].

[23] European Commission, "Ethical Guidelines on the Use of Artificial Intelligence and Data in Teaching and Learning for Educators", European Commission, Directorate-General for Education, Youth, Sport and Culture, inf. téc., 2022.

[24] European Union, *Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act)*, Official Journal of the European Union, L series, 2024.

[25] A. Vangelova y A. Aleksieva-Petrova, "An Integrated RAG and Agent-Based Architecture for Automated Assessment in Moodle", *Machine Learning and Knowledge Extraction*, vol. 8, n.º 127, 2026.

[26] A. Ostrowska et al., "From Surface Learning to Deep Understanding: A Grounded AI Tutoring System for Moodle", *arXiv preprint arXiv:2605.06963*, 2026.

[27] Khairu Aqsara. "Terus RAG Block for Moodle". GitHub repository of the Moodle plugin block_terusrag, 19 de jun. de 2026. dirección: https://github.com/khairu-aqsara/terusrag.

[28] Microsoft. "moodle-ai-assistant". GitHub repository; version 0.1.0 Alpha; MIT license, 19 de jun. de 2026. dirección: https://github.com/microsoft/moodle-ai-assistant.

[29] A. R. Hevner, S. T. March, J. Park y S. Ram, "Design Science in Information Systems Research", *MIS Quarterly*, vol. 28, n.º 1, págs. 75-105, 2004.

[30] K. Peffers, T. Tuunanen, M. A. Rothenberger y S. Chatterjee, "A Design Science Research Methodology for Information Systems Research", *Journal of Management Information Systems*, vol. 24, n.º 3, págs. 45-77, 2007. DOI: 10.2753/MIS0742-1222240302.

[31] FastAPI. "FastAPI Documentation", 6 de jul. de 2026. dirección: https://fastapi.tiangolo.com/.

[32] Chroma. "Chroma Documentation: Introduction", 6 de jul. de 2026. dirección: https://docs.trychroma.com/docs/overview/introduction.

[33] Ollama. "Ollama API Documentation", 6 de jul. de 2026. dirección: https://docs.ollama.com/api/introduction.

[34] Moodle. "External Services", 6 de jul. de 2026. dirección: https://moodledev.io/docs/5.0/apis/subsystems/external.

[35] N. Reimers e I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", en *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, 2019, págs. 3982-3992. DOI: 10.18653/v1/D19-1410.

[36] G. V. Cormack, C. L. A. Clarke y S. Büttcher, "Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods", en *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 2009, págs. 758-759. DOI: 10.1145/1571941.1572114.

[37] A. Gan et al., "Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey", *Frontiers of Computer Science*, 2025, Review article. Versión preprint en arXiv (21 Apr 2025). arXiv: 2504.14891 [cs.CL].

[38] J. Cohen, "A Coefficient of Agreement for Nominal Scales", *Educational and Psychological Measurement*, vol. 20, n.º 1, págs. 37-46, 1960. DOI: 10.1177/001316446002000104.

V

Parte

Apéndices

A Manual de despliegue reproducible

Este anexo describe el despliegue del prototipo en un equipo local. No constituye una guía de despliegue institucional completo. Los ejemplos utilizan marcadores de posición entre ángulos en lugar de valores reales; ningún secreto debe almacenarse en el repositorio.

A.1. Requisitos previos

El despliegue asume un sistema anfitrión con `Docker` y `Docker Compose` instalados. Las dependencias de `Python` del motor RAG se gestionan dentro del contenedor y no requieren instalación manual por parte del operador. Se asume, además, la disponibilidad de:

- una instancia de `Moodle` ya operativa;
- un *token* de la `API REST` de `Moodle` configurado en `Moodle` con los permisos mínimos necesarios para enumerar y descargar los materiales del curso;
- un servidor `Ollama` accesible en local o *self-hosted*;
- el modelo generador y el modelo de *embeddings* descargados previamente en `Ollama` y en el contenedor, respectivamente, si el entorno no dispone de acceso de red durante el arranque;
- acceso administrativo a la instancia de `Moodle` para instalar y configurar los dos bloques de integración desarrollados en `PHP`.

Una `GPU` local puede reducir la latencia de inferencia, pero no es un requisito conceptual del despliegue reproducible: el sistema funciona sobre una unidad central de procesamiento (*Central Processing Unit*, `CPU`) con modelos locales de tamaño moderado. Ningún secreto, como *tokens* o claves, debe incluirse en el repositorio; se gestionan mediante un fichero de entorno no versionado.

A.2. Servicios obligatorios y opcionales

La Tabla A.1 resume los componentes del despliegue del prototipo.

Tabla A.1: Servicios del despliegue del prototipo.

Componente	**Estado**	**Función**
Moodle	Obligatorio	LMS y fuente documental.
FastAPI + RAG Python	Obligatorio	API de consulta e indexación; motor RAG.
ChromaDB	Obligatorio	Persistencia vectorial local del prototipo.
Ollama	Obligatorio	Inferencia local/*self-hosted*.
Bloque Moodle PHP de chat RAG	Obligatorio	Interfaz de consulta integrada en Moodle.
Bloque Moodle PHP de indexación	Obligatorio	Interfaz para lanzar la indexación desde Moodle.
Reranker local	Opcional	Desactivado en la configuración final.
Monitorización	Opcional	Apoyo operativo; línea futura.
Qdrant	No incluido	Evolución futura *self-hosted*; no forma parte del prototipo.

El almacén vectorial del prototipo es `ChromaDB`, persistido en un volumen local del contenedor. `Qdrant` no forma parte de este despliegue y se documenta únicamente como línea futura en el Capítulo 9. Los dos bloques de `Moodle` desarrollados en `PHP` actúan como interfaces finas sobre `FastAPI` y no implementan recuperación, generación, *embeddings*, citación, abstención, *reranking* ni evaluación: toda la lógica RAG reside en el motor desarrollado en `Python`.

A.3. Variables de entorno

Las variables de entorno se definen en un fichero no versionado a partir de una plantilla de ejemplo. La Tabla A.2 recoge las variables relevantes con ejemplos seguros; los nombres exactos pueden variar según la configuración concreta del despliegue, pero en ningún caso deben almacenarse valores reales en el repositorio.

A.4. Instalación paso a paso

El procedimiento reproducible en un equipo local desde cero es el siguiente:

1. Preparar el fichero de entorno a partir de la plantilla y completar los valores locales sin versionarlos.

Tabla A.2: Variables de entorno del despliegue (ejemplos seguros).

Variable	Ejemplo seguro	Finalidad
MOODLE_BASE_URL	<MOODLE_BASE_URL>	URL base de Moodle.
MOODLE_TOKEN	<TOKEN_PRIVADO>	Token Moodle REST para ingesta.
RAG_API_TOKEN	<RAG_SHARED_SECRET>	Secreto compartido entre la API y los bloques.
OLLAMA_BASE_URL	<OLLAMA_BASE_URL>	Endpoint del servidor Ollama.
RAG_RETRIEVAL_MODE	hybrid	Modo de recuperación.
RAG_HYBRID_RRF_K	20	Parámetro de fusión RRF.
RAG_ABSTENTION_MIN_SCORE	0.52	Umbral de abstención.
RAG_RERANK_ENABLED	false	Activación del *re-ranking*.
RAG_ANSWERABILITY_POLICY_ENABLED	false	Política de *answerability*.
CHROMA_PERSIST_DIR	./data/chroma	Volumen persistente de ChromaDB.
COURSE_ID	<COURSE_ID>	Curso a indexar.

2. Comprobar que Ollama está disponible y que los modelos locales están descargados.
3. Levantar el servicio de la API y el motor RAG con Docker Compose.
4. Comprobar el estado del servicio mediante el *endpoint* de salud.
5. Instalar o activar los dos bloques de Moodle desarrollados en PHP en la instancia de Moodle.
6. Configurar en Moodle, desde la administración de cada bloque, la URL de FastAPI y el secreto compartido.
7. Lanzar la indexación del curso desde el bloque de indexación o, de forma equivalente, mediante el *endpoint* /index-course.
8. Probar una consulta desde el bloque de *chat* RAG.

Los pasos con línea de comandos utilizan órdenes seguras. La preparación del entorno y el arranque de los servicios:

```
cp .env.example .env
docker compose up -d
docker compose logs -f rag-api
```

La comprobación de salud y el lanzamiento de la indexación, sin escribir el valor real del *token* en la línea de comandos, se realizan mediante:

```
curl <FASTAPI_BASE_URL>/health

curl -X POST <FASTAPI_BASE_URL>/index-course \
  -H "Authorization: Bearer <RAG_SHARED_SECRET>" \
  -H "Content-Type: application/json" \
  -d '{"course_id": "<COURSE_ID>", "force": false}'
```

Una vez indexado el curso, la consulta se realiza desde el bloque de *chat* RAG dentro de `Moodle`, que construye el contexto docente y lo envía al *endpoint* `/ask`.

A.5. Comandos de ejecución habitual

Los comandos de operación habitual del prototipo son:

```
docker compose up -d              # levantar servicios
docker compose logs -f rag-api    # ver logs del servicio
curl <FASTAPI_BASE_URL>/health    # comprobar salud
docker compose down               # detener sin borrar datos
```

La orden `docker compose down` detiene los servicios, pero conserva los datos persistentes mientras se mantengan los volúmenes. Por el contrario, `docker compose down -v` elimina los volúmenes asociados y, con ellos, el índice vectorial y el estado de indexación; debe evitarse salvo que se pretenda un reinicio deliberado del despliegue desde cero.

A.6. Notas sobre reproducibilidad

La reproducibilidad del despliegue se apoya en varias consideraciones. Conviene fijar las versiones de las dependencias y el *snapshot* del corpus indexado empleado en la evaluación. La zona sensible de datos privados no debe incluirse en el repositorio y el fichero de entorno con secretos no debe versionarse. La configuración experimental puede registrarse de forma trazable siempre que no exponga datos sensibles del corpus.

Las latencias observadas dependen del *hardware* y del modelo local empleados, por lo que no son directamente comparables entre entornos distintos. Además, la reproducibilidad funcional del despliegue no implica una igualdad exacta, de *token* a *token*, de las respuestas generadas: la generación con modelos de lenguaje introduce variabilidad que este manual no pretende suprimir. Por último, este anexo describe el despliegue del proto-

tipo evaluado en el trabajo y no un despliegue institucional completo, cuyas implicaciones se apuntan como línea futura en el Capítulo 9.

A efectos de contextualización, el desarrollo y las ejecuciones experimentales del prototipo se realizaron en un entorno local de prototipado sobre un portátil `OMEN Gaming Laptop 16-ap0xxx` con `Windows 11 Pro` y el Subsistema de Windows para Linux 2 (*Windows Subsystem for Linux 2*, `WSL2`). El equipo dispone de un procesador `AMD Ryzen AI 7 350` con `Radeon 860M`, 32 GB de memoria de acceso aleatorio (*Random Access Memory*, `RAM`) y una `GPU` dedicada `NVIDIA GeForce RTX 5070 Laptop GPU` con 8 GB de memoria, además de la `GPU` integrada `AMD Radeon 860M`. El entorno `Linux` utilizado fue `Ubuntu 24.04.4 LTS` sobre `WSL2`, donde LTS corresponde a soporte a largo plazo (*Long-Term Support*), con `WSL` en su versión `2.6.3.0` y el *kernel* `6.6.87.2`. La orquestación del prototipo se realizó con `Docker 29.1.3` y `Docker Compose v2.40.3`, mientras que la inferencia local se sirvió mediante `Ollama 0.14.2` ejecutado dentro del entorno `WSL`. La `GPU` fue visible desde `WSL` mediante `nvidia-smi`, con el *driver* `592.27`, `CUDA 13.1` y 8151 MiB de memoria disponible.

Estos datos se incluyen únicamente para contextualizar la reproducibilidad y la interpretación de las latencias observadas. No representan una recomendación de dimensionamiento institucional, ya que el rendimiento dependerá del modelo local empleado, la configuración de `WSL2`, la carga concurrente, el tamaño del corpus y los recursos disponibles en cada despliegue.

B Contratos de API

Este anexo documenta los contratos HTTP que expone la aplicación FastAPI del motor RAG. Los dos *endpoints* principales, /ask y /index-course, son los que consumen los bloques de Moodle desarrollados en PHP, que actúan como interfaces finas: no implementan recuperación, generación, *embeddings*, citación, abstención, *reranking* ni evaluación. Toda esa lógica reside en el motor RAG desarrollado en Python. Los ejemplos utilizan marcadores de posición entre ángulos en lugar de valores reales; ningún ejemplo reproduce contenido del corpus ni *tokens* reales.

B.1. *Endpoint* /ask

El *endpoint* /ask atiende las consultas al sistema RAG. Su consumidor principal es el bloque de Moodle desarrollado en PHP para el *chat* RAG, que construye el contexto docente a partir de la sesión de Moodle y envía la pregunta del usuario. Es un *endpoint* que utiliza el método POST y está protegido por autenticación de consulta (§B.3).

El esquema de petición está definido de forma estricta, con prohibición de campos adicionales, de modo que se rechazan campos personales no previstos. El cuerpo se representa mediante la notación de objetos de JavaScript (*JavaScript Object Notation*, JSON):

Listing B.1: Petición a `/ask`.

```
POST <FASTAPI_BASE_URL>/ask
Authorization: Bearer <RAG_SHARED_SECRET>
Content-Type: application/json

{
  "question": "<PREGUNTA_DEL_USUARIO>",
  "context": {
    "course_id": "<COURSE_ID>",
    "role": "student",
    "section_id": "<SECTION_ID>",
    "activity_id": "<ACTIVITY_ID>",
    "resource_id": "<RESOURCE_ID>",
    "language": "es",
    "visibility": "visible"
  }
}
```

El campo `role` se valida contra una lista cerrada, formada por `student`, `teacher`, `editingteacher`, `guest` y `admin`; el resto de los campos del contexto, salvo `course_id` y `role`, son opcionales. La respuesta incluye el estado de la consulta, la respuesta generada, las fuentes citadas y un bloque de metadatos operativos:

Listing B.2: Respuesta de `/ask` (metadatos resumidos).

```
{
  "status": "answered",
  "answer": "<RESPUESTA_GENERADA>",
  "sources": [
    {
      "chunk_id": "<CHUNK_ID>",
      "source_uri": "<SOURCE_URI>",
      "document_title": "<DOCUMENT_TITLE>",
      "section": "<SECTION>",
      "page": 1
    }
  ],
  "metadata": { ... },
  "warnings": [],
  "latency_ms": 0,
  "request_id": "<REQUEST_ID>"
}
```

El campo `status` toma uno de los valores `answered`, `abstained`, `error`, `invalid_request` o `degraded`. El objeto `metadata` agrupa los indicadores operativos de la respuesta: el estado y el motivo de abstención cuando el sistema se abstiene, la mejor puntuación de recuperación, el resultado de la verificación estructural de citas, el perfil pedagógico aplicado y una bandera defensiva de fuga de plantilla. Cuando la política de *answerability* está activa, el objeto incorpora campos adicionales opcionales, como el modo, las partes soportadas y no soportadas y la oferta de clarificación.

Nota de privacidad. El contrato está diseñado para minimizar la exposición de datos. El esquema del contexto rechaza campos personales no previstos; el objeto de cada fuente devuelve únicamente metadatos de procedencia y **no** incluye el contenido del documento. Los ejemplos de este anexo no reproducen preguntas ni respuestas reales.

Errores. Los errores observables de este *endpoint* siguen la política común descrita en la Sección B.4.

B.2. *Endpoint* `/index-course`

El *endpoint* `/index-course` lanza la indexación de los materiales de un curso. Su consumidor principal es el bloque de `Moodle` desarrollado en `PHP` para la indexación. Es un *endpoint* que utiliza el método `POST` y está protegido por autenticación de indexación (§B.3).

Listing B.3: Petición a `/index-course`.

```
POST <FASTAPI_BASE_URL>/index-course
Authorization: Bearer <RAG_SHARED_SECRET>
Content-Type: application/json

{
  "course_id": "<COURSE_ID>",
  "force": false,
  "sync_visibility": true
}
```

El campo `force` solicita una reindexación completa ignorando el estado previo. Su valor predeterminado es `false`, lo que permite una indexación incremental basada en el resumen criptográfico de cada documento. El campo `sync_visibility`, cuyo valor predeterminado es `true`, sincroniza la visibilidad de los recursos con la política vigente en `Moodle`. La respuesta es un resultado agregado que resume el efecto de la operación sin exponer el contenido del corpus:

Listing B.4: Respuesta agregada de `/index-course`.

```
{
  "status": "ok",
  "documents_new": 0,
  "documents_modified": 0,
  "documents_unchanged": 0,
  "documents_deleted": 0,
  "chunks_added": 0,
  "embedding_recomputed": 0,
  "visibility_updates": 0,
  "warnings": [],
  "errors": []
}
```

Nota de privacidad. La respuesta contiene únicamente contadores agregados y mensajes resumidos de advertencia o error; no incluye títulos reales de documentos ni fragmentos del corpus.

Errores. Los errores observables de este *endpoint* siguen la política común descrita en la Sección B.4.

***Endpoints* auxiliares.** La aplicación `FastAPI` expone, además, un conjunto de *endpoints* auxiliares que **no** son consumidos por los bloques de `Moodle` en la versión verificada del prototipo:

- `GET /index-status/course_id`: estado de indexación de un curso, protegido por autenticación de indexación.
- `POST /sync-visibility`: sincronización de visibilidad a partir de un contexto con `course_id`.
- `GET /health`: comprobación básica de salud, que incluye estado, servicio, versión y entorno.
- `GET /health/deep`: diagnóstico con comprobación de dependencias.

Ninguno de estos *endpoints* expone secretos ni contenido del corpus.

B.3. Autenticación y control de acceso

La API distingue dos dependencias de autenticación: una para la consulta y otra para las operaciones de indexación. La primera protege /ask; la segunda, /index-course y los *endpoints* auxiliares de indexación. Ambas emplean un esquema de *token* portador: la petición incluye una cabecera de autorización con el esquema Bearer seguido del secreto compartido, según se muestra en los ejemplos anteriores. El servidor compara el valor recibido con el configurado mediante una comparación en tiempo constante.

La autenticación es opcional según la configuración del entorno: si la comprobación está desactivada, no se exige *token*. Para despliegues reales del prototipo se recomienda mantenerla activa, con *tokens* distintos para consulta e indexación. En Moodle, el localizador uniforme de recursos (*Uniform Resource Locator*, URL) de la API y el *token* de cada bloque se configuran desde la administración del bloque correspondiente; no se codifican en el repositorio. El *token* no debe registrarse en *logs* ni aparecer en los detalles de los mensajes de error. Este anexo describe el contrato de autenticación del prototipo y no constituye una guía institucional de seguridad.

B.4. Códigos de error

Ambos *endpoints* principales comparten la misma política de errores, resumida en la Tabla B.1. Los códigos 403 y 404 no se documentan como manejadores específicos en la versión verificada, por lo que no se incluyen como parte del contrato operativo del prototipo. El comportamiento ante rutas o métodos no definidos corresponde al tratamiento predeterminado del *framework*.

Tabla B.1: Códigos de error de la API.

Código	**Causa típica**	**Tratamiento recomendado**
400	Petición de negocio inválida (pregunta vacía, `course_id` vacío).	Corregir el contenido de la petición.
401	Token ausente o inválido con autenticación activa.	Revisar la configuración del token en el bloque.
422	Fallo de validación del esquema de entrada.	Ajustar los campos al contrato documentado.
500	Error interno no recuperable.	Consultar los *logs* del servicio.
503	Servicio no configurado o dependencia no disponible.	Verificar configuración y dependencias.

C Estructura del repositorio

Este anexo describe la estructura limpia y publicable del motor RAG desarrollado en `Python`: el conjunto mínimo de elementos necesarios para ejecutar, probar y mantener el sistema. No reproduce el árbol completo del entorno de trabajo local, que incluye además utilidades de desarrollo, datos derivados del corpus real, resultados por pregunta e informes internos que no forman parte del artefacto publicable. Los bloques de `Moodle` desarrollados en `PHP`, que actúan como interfaz de usuario, residen en un repositorio complementario separado y se comentan al final.

Disponibilidad del código. La implementación asociada a este trabajo se publica en dos repositorios alojados en `GitHub`:

- Motor RAG desarrollado en `Python`: `https://github.com/AdolfoPM02/moodle-rag-engine`.
- Bloques de `Moodle` y entorno de integración: `https://github.com/AdolfoPM02/LoyolaTeams-IA`.

Este último repositorio se ha desarrollado junto con Andrés Jiménez Bonilla, estudiante de segundo curso del Grado en Ingeniería del Software en la Universidad Loyola y miembro de Loyola Teams, equipo de IA.

C.1. Árbol de carpetas

La estructura siguiente presenta el motor como un artefacto autocontenido, no como una copia del directorio de trabajo:

```
moodle-rag-engine/
|-- src/
|   `-- moodle_rag/
|       |-- api/
|       |-- ingestion/
|       |-- indexing/
|       |-- retrieval/
|       |-- generation/
|       |-- integrations/
|       |-- evaluation/
|       `-- schemas.py
|-- tests/
|-- pyproject.toml
|-- uv.lock
|-- Dockerfile
|-- docker-compose.yml
|-- .dockerignore
|-- .gitignore
|-- README.md
`-- .env.example
```

El árbol se organiza en torno al paquete `moodle_rag`, situado bajo `src/`, y a un directorio de pruebas paralelo; en la raíz se sitúan los ficheros de configuración del proyecto y del despliegue reproducible descrito en el Anexo A. El subpaquete `evaluation` aparece como módulo auxiliar de evaluación reproducible: sostiene el marco experimental de los capítulos de resultados, pero no forma parte del camino de ejecución del servicio ni es importado por la capa de servicio. El directorio `tests/` se incluye por su valor de verificación, ya que refleja la estructura de los paquetes sin contener material real del corpus. Los ficheros `.dockerignore`, `README.md` y `.env.example` forman parte de la versión publicable. El primero limita el contexto de construcción de la imagen de `Docker`, mientras que los dos restantes documentan la puesta en marcha y las variables de entorno mediante valores de ejemplo sin credenciales reales.

El árbol mínimo excluye deliberadamente los directorios de trabajo que no son necesarios para servir el motor: utilidades operativas, conjuntos de datos de evaluación, informes técnicos, datos derivados del corpus, documentación de la memoria, ficheros `PDF`, referencias y artefactos locales, como entornos virtuales o cachés. El repositorio de los bloques de `Moodle` desarrollados en `PHP` tampoco forma parte de este árbol.

C.2. Paquetes principales

El motor se estructura en subpaquetes con responsabilidades separadas. `api` define los contratos `HTTP`, la validación y el cableado del servicio `FastAPI` (Anexo B); `ingestion` realiza la carga y el *parsing* multiformato de los materiales docentes; `indexing` genera los *embeddings* y gestiona la persistencia vectorial y la indexación de fragmentos; `retrieval` implementa la recuperación densa, léxica mediante `BM25` e híbrida con fusión `RRF`, además de un *reranking* opcional; `generation` compone el *prompt* y cubre la generación local, la citación, la abstención, la política de *answerability* y la comprobación defensiva frente a fugas de plantilla; `integrations` contiene el cliente de la `API REST` de `Moodle`, que aísla el acceso al LMS; `evaluation` agrupa las métricas y los *runners* reproducibles como componente auxiliar no importado por la capa de servicio; y `schemas.py` reúne los modelos de datos compartidos.

La separación se mantiene en las dependencias: la capa de servicio no depende del módulo de evaluación, y la generación y la recuperación no acceden directamente a `Moodle`, cuyo acceso queda confinado en `integrations`.

C.3. *Scripts* operativos

Ningún *script* externo es necesario para atender la consulta principal: el servicio se despliega mediante `Docker Compose` y expone su lógica a través de `FastAPI` (Anexos A y B). El entorno de trabajo contiene además utilidades auxiliares de indexación, evaluación, construcción de conjuntos de datos e inspección, empleadas durante el desarrollo. No forman parte del árbol mínimo porque requieren una curación específica; las ligadas a materiales o resultados reales quedan excluidas por privacidad.

C.4. *Datasets* de evaluación

Los conjuntos de datos de evaluación no son necesarios para atender la consulta y quedan fuera del árbol mínimo. Podrían publicarse como paquete complementario separado siempre que estuvieran adecuadamente anonimizados. No se versionan corpus privados, enunciados sensibles, fuentes reales, fragmentos, identificadores concretos ni instantáneas del almacén vectorial. El Anexo E documenta su estructura, categorías y criterios de anotación sin reproducir su contenido.

C.5. Informes de evaluación

Los informes de evaluación justifican las decisiones técnicas del trabajo, pero no son necesarios para ejecutar el motor y quedan fuera del árbol mínimo. No se versionan los resultados por pregunta, los registros completos de ejecución, las respuestas generadas ni las trazas con material del corpus. El Anexo F recoge los resultados extendidos de forma agregada y anonimizada.

Conviene precisar la frontera entre el motor y su interfaz: los bloques de `Moodle` desarrollados en `PHP` pertenecen a un repositorio complementario separado y actúan como interfaces finas, que consumen el servicio `FastAPI` mediante los contratos del Anexo B y no implementan recuperación, generación, *embeddings*, citación, abstención ni evaluación. Toda la lógica del sistema RAG permanece en el motor desarrollado en `Python`.

D Plantillas de *prompt*

Este anexo documenta la política de *prompt* del prototipo con fines de reproducibilidad, sin reproducir literalmente las instrucciones internas del sistema. La lógica de composición del *prompt*, la generación, la citación y la abstención residen en el motor RAG desarrollado en `Python`; los bloques de `Moodle` desarrollados en `PHP` actúan como interfaz de usuario y no implementan estas funciones. La política de *prompt*, implementada mediante la clase `PromptPolicy`, está activa en la configuración final del prototipo.

D.1. *Prompt* base común

El *prompt* del sistema se compone de un bloque de instrucciones comunes, independiente del perfil del usuario, y de un bloque específico de perfil (§D.2 y §D.3). El bloque común recibe como entradas la pregunta del usuario y los fragmentos recuperados, y fija las reglas de anclaje documental que gobiernan la generación. De forma resumida, estas reglas establecen que el modelo debe:

- responder únicamente con la información presente en las fuentes recuperadas, sin recurrir a conocimiento externo;
- citar las afirmaciones relevantes mediante marcadores numéricos (§D.4);
- no inventar información no sostenida por el material;
- ante una pregunta con varias partes, responder únicamente las partes respaldadas por las fuentes y declarar de forma explícita cuáles no están cubiertas;
- abstenerse cuando las fuentes no contienen información suficiente (§D.5);
- no reproducir las instrucciones internas, los marcadores estructurales de la plantilla ni la cabecera del sistema, ni exponer rutas, *tokens* o credenciales que pudieran aparecer en el material;
- redactar la respuesta en castellano, con independencia del idioma de las fuentes.

El perfil pedagógico, derivado del rol de `Moodle`, determina qué bloque de perfil se añade al bloque común.

D.2. *Prompt* para alumnado

El perfil `student` se aplica a los roles de alumnado y, por degradación conservadora, también cuando el rol es desconocido, está vacío o no se reconoce. Su bloque de instrucciones orienta la respuesta hacia un tono didáctico y un lenguaje accesible: guía al estudiante paso a paso cuando la materia lo requiere, define de forma breve los términos técnicos imprescindibles y evita el tono de diagnóstico interno. En este perfil, la respuesta no menciona identificadores internos del sistema, identificadores de fragmento ni detalles de implementación del motor RAG. Ante una pregunta parcialmente cubierta por el material, el perfil indica primero qué partes pueden responderse y, a continuación, qué partes no están cubiertas por el material del curso.

D.3. *Prompt* para profesorado

El perfil `teacher` agrupa los roles docentes y de gestión. La normalización de roles asigna a este perfil los valores correspondientes a profesor, profesor editor, gestor y creador de cursos, entre otros; en particular, el rol de administración se trata como `teacher` a efectos de generación. Su bloque de instrucciones adopta un tono técnico y sintético, asume dominio del área y pone énfasis en la trazabilidad, dejando claro qué afirmación se apoya en cada fuente. Además, el perfil señala de forma explícita las lagunas del corpus respecto a la pregunta, de modo que el docente pueda valorar si el material del curso cubre un tema determinado. Este perfil constituye una ayuda para la revisión de la cobertura documental y no una herramienta de evaluación automática, ni sustituye el criterio docente.

D.4. Instrucciones de citación

Las instrucciones de citación obligan a que cada afirmación relevante de la respuesta quede respaldada por las fuentes recuperadas y marcada con un identificador numérico entre corchetes, como `[1]` o `[2]`. Los marcadores se asocian de forma posicional con los fragmentos incorporados al contexto de generación. Cada fuente devuelta al bloque de `Moodle` contiene únicamente metadatos de procedencia y no el contenido completo del documento.

Sobre la respuesta se ejecuta una verificación estructural de las citas, cuyo alcance se describe en el Capítulo 5 (§5.8). Esta verificación comprueba la correspondencia posicional entre los marcadores utilizados y los fragmentos recuperados, y detecta situaciones anómalas como citas ausentes, citas no reconocidas, citas malformadas o citas duplicadas. Se trata de una verificación estructural y no de una validación semántica: refuerza la trazabilidad de la respuesta, pero no comprueba materialmente que cada afirmación esté sostenida por el pasaje citado.

D.5. Instrucciones de abstención

La abstención opera en dos capas complementarias. La primera es una política determinista que actúa **antes** de invocar al modelo generador: compara la mejor puntuación de recuperación con un umbral mínimo configurable, definido mediante la variable `RAG_ABSTENTION_MIN_SCORE`, cuyo valor operativo es `0.52` en la configuración final, y decide la abstención por ausencia de resultados suficientes o por puntuación insuficiente. La segunda capa es la instrucción del *prompt* común que ordena al modelo abstenerse cuando las fuentes no permiten sostener una respuesta.

Cuando la política determinista decide abstenerse, el sistema no invoca al modelo y devuelve un mensaje uniforme que informa de la ausencia de información suficiente en el material, junto con el motivo operativo de la abstención. Esta abstención reduce el riesgo de producir respuestas no sostenidas por el corpus, pero no lo elimina: constituye un mecanismo de seguridad, no una garantía absoluta.

D.6. Restricciones frente a la invención

El conjunto de restricciones frente a la invención se apoya en el anclaje documental descrito en §D.1: el modelo debe responder solo con el material recuperado, sin recurrir a conocimiento externo ni fabricar datos ausentes del corpus. Adicionalmente, la política prohíbe reproducir las instrucciones internas de la plantilla, sus marcadores estructurales o las cabeceras del sistema, así como exponer rutas, *tokens* o credenciales presentes en el material. Como capa defensiva complementaria, el *pipeline* aplica una comprobación posterior a la generación que inspecciona la respuesta en busca de fugas de plantilla (*prompt leakage*). Esta comprobación reduce la probabilidad de que se filtren elementos internos, aunque no puede considerarse infranqueable. Por prudencia, este anexo no reproduce los patrones concretos que emplea dicha comprobación.

De forma independiente, el prototipo incorpora una política de *answerability*, implementada mediante la clase `AnswerabilityPolicy`, que contempla respuestas parciales y turnos de clarificación ante preguntas mixtas o ambiguas. Esta política permanece **desactivada por defecto** en la configuración final, mediante la variable `RAG_ANSWERABILITY_POLICY_ENABLED`, fijada a `false`. No se ha reevaluado bajo esa configuración.

En síntesis, quedan **implementados** la política de *prompt* activa, los perfiles de alumnado y profesorado, las instrucciones de citación y abstención, y la comprobación defensiva de fugas de plantilla; se ha **evaluado** su efecto dentro del protocolo experimental del Capítulo 7; las plantillas constituyen un **prototipo** adaptado al contexto de `Moodle`; y se mantienen como **líneas futuras** el refinamiento pedagógico con profesorado y la evaluación con usuarios reales.

E *Dataset* de evaluación

Este anexo documenta la estructura, las categorías y los criterios de anotación del conjunto de preguntas de evaluación, con fines de reproducibilidad y sin reproducir su contenido. El corpus asociado es privado; por ello, la memoria no incluye los enunciados de las preguntas, las respuestas, los fragmentos del corpus ni los títulos o identificadores de las fuentes.

E.1. Estructura del conjunto de preguntas

El conjunto de preguntas se almacena en formato de líneas JSON (*JSON Lines*, `JSONL`), con una pregunta por línea. Se manejan dos variantes principales del mismo conjunto de 106 preguntas: un *benchmark* ampliado con anotación a nivel de documento y una versión anotada a nivel de fragmento, orientada a la evaluación de la recuperación. El conjunto **no** incluye un campo de respuesta textual de referencia; en su lugar, cada pregunta se caracteriza mediante etiquetas de comportamiento esperado y relevancia, lo que reduce la exposición del contenido del corpus y resulta coherente con el enfoque *privacy-first* del trabajo. La Tabla E.1 resume los campos principales del esquema.

E.2. Tipos de pregunta

Las preguntas se clasifican por ámbito. La Tabla E.2 resume la distribución y la finalidad experimental de cada tipo; la suma de las cinco categorías es de 106 preguntas.

Esta clasificación se corresponde con el comportamiento esperado registrado en el campo `expected_behavior`, cuyos valores, respuesta, respuesta parcial, clarificación y abstención, se distribuyen en 61, 16, 12 y 17 preguntas, respectivamente.

Tabla E.1: Campos principales del esquema del conjunto de preguntas.

Campo	**Función**
`id`	Identificador único de la pregunta.
`question`	Enunciado de la pregunta (no reproducido en la memoria).
`scope`	Ámbito de la pregunta (véase §E.2).
`expected_behavior`	Comportamiento esperado del sistema.
`expected_answerability_mode`	Modo de respondibilidad esperado.
`relevant_sources`	Documentos o recursos relevantes (nivel de fuente).
`relevant_chunks`	Fragmentos relevantes (nivel de fragmento).
`must_cover`	Elementos que la respuesta debería cubrir.
`must_not_answer`	Elementos que la respuesta no debería incluir.
`notes`	Notas de anotación de apoyo.

E.3. Fuentes relevantes

La anotación a nivel de documento, registrada en el campo `relevant_sources`, identifica para cada pregunta los documentos o recursos del curso que sostienen la respuesta esperada a un nivel general. Esta anotación está presente en 89 de las 106 preguntas. La memoria no reproduce títulos, rutas ni referencias de las fuentes, con el fin de preservar la privacidad del corpus.

Esta anotación documental se distingue de la anotación a nivel de fragmento (§E.4): la primera opera sobre el documento o recurso completo, mientras que la segunda identifica los fragmentos concretos que sostienen la respuesta y se emplea en la evaluación del *ranking* de recuperación.

E.4. Fragmentos relevantes

La versión anotada a nivel de fragmento añade esta información en el campo `relevant_chunks`. Este subconjunto comprende 58 preguntas puntuables para las métricas de recuperación y un total de 193 fragmentos marcados como relevantes. Sobre estas 58 preguntas se calculan las métricas de recuperación a nivel de fragmento, *Precision@5*, *Recall@5*, MRR y nDCG@5, descritas en el Capítulo 6.

Tabla E.2: Tipos de pregunta, recuento y finalidad experimental.

Tipo	**N.º**	**Finalidad experimental**
`in_scope`	51	Preguntas respondibles desde el corpus.
`in_scope_rephrased`	10	Reformulaciones para evaluar robustez léxica.
`mixed`	16	Combinan una parte cubierta y otra no cubierta.
`vague`	12	Formulaciones ambiguas que pueden requerir clarificación.
`out_of_scope`	17	Fuera del temario, orientadas a evaluar la abstención.

No todas las preguntas del *benchmark* disponen de fragmentos relevantes. Las restantes se emplean en las dimensiones de abstención, robustez, ambigüedad o análisis de limitaciones, según su tipo. La memoria no reproduce identificadores concretos de fragmentos ni su contenido. La anotación fue realizada por el autor, por lo que no se estima acuerdo interanotador.

E.5. Evidencia disponible

El conjunto distingue, de forma implícita mediante sus etiquetas, entre preguntas con evidencia suficiente en el corpus, preguntas con evidencia parcial, preguntas ambiguas y preguntas sin evidencia suficiente. Esta distinción se corresponde con los valores de comportamiento esperado: respuesta completa para las preguntas con evidencia suficiente, respuesta parcial para las preguntas mixtas, clarificación para las preguntas ambiguas y abstención para las preguntas fuera del temario.

Las preguntas `out_of_scope` se emplean para medir la abstención del sistema, mientras que las preguntas `mixed` y `vague` permiten observar los límites del *pipeline* y de la política de *answerability*, aunque esta última permanezca desactivada por defecto en la configuración final. Este diseño permite analizar el comportamiento del sistema en distintos escenarios de evidencia, sin pretender cubrir de forma exhaustiva todas las situaciones posibles.

E.6. Criterio de anotación

La anotación a nivel de fragmento sigue un criterio conservador. Se consideran relevantes únicamente los fragmentos que sostienen realmente una respuesta a la pregunta. Se descartan las portadas de material docente, los fragmentos puramente visuales cuyo contenido no se recupera como texto y los fragmentos con contexto insuficiente. Se priorizan, por tanto, los fragmentos con contenido textual explícito y trazable.

La anotación fue realizada por el autor, por lo que no se estima acuerdo interanotador. Esta circunstancia se declara explícitamente como limitación metodológica y se retoma, junto con las restantes amenazas a la validez, en el Capítulo 8.

F Resultados extendidos

Este anexo complementa el Capítulo 7 con resultados agregados adicionales. No reproduce las tablas principales del capítulo ni los artefactos experimentales por pregunta, y presenta la información de forma anonimizada para no exponer contenido del corpus.

F.1. Resultados por pregunta

El protocolo experimental genera, para cada configuración, un conjunto de artefactos por pregunta que sostienen el cálculo de las métricas agregadas presentadas en el Capítulo 7. Estos artefactos registran, para cada pregunta, identificadores internos, el resultado de la recuperación, la decisión de abstención y las advertencias asociadas a la respuesta. Por motivos de privacidad y para no exponer la estructura del corpus, no se reproducen en la memoria. Permanecen como soporte técnico interno del protocolo, versionado junto con el resto de la instrumentación de evaluación. Las secciones siguientes documentan únicamente vistas agregadas complementarias.

F.2. Resultados por configuración

Las métricas principales de recuperación se calculan a nivel de fragmento sobre el subconjunto de preguntas puntuables (Capítulo 7, Tabla 7.1). La generación opera sobre fragmentos, por lo que las métricas a nivel de fragmento son las de referencia para las decisiones de configuración.

F.3. Errores observados

En lugar de reproducir ejemplos reales, se resume la taxonomía de los errores observados durante la evaluación:

- **Recuperación insuficiente ante reformulaciones**: preguntas que reformulan

conceptos del material sin coincidencia léxica directa, en las que la recuperación no sitúa la evidencia correcta en las primeras posiciones.

- **Coincidencia léxica demasiado específica**: casos en los que la recuperación léxica acierta por coincidencia exacta, pero no generaliza a formulaciones alternativas.
- **Abstención indebida ante preguntas ambiguas**: preguntas de formulación imprecisa en las que el sistema se abstiene pese a existir evidencia parcialmente relevante.
- **Problemas estructurales de cita**: situaciones detectadas por la verificación estructural en forma de citas ausentes, citas no reconocidas, citas malformadas o citas duplicadas.

Estas categorías se describen a un nivel agregado; no se incluyen enunciados, respuestas ni identificadores concretos.

F.4. Trazas resumidas de recuperación

A modo de síntesis interpretativa y sin reproducir registros literales de ejecución, se resumen los patrones de recuperación observados:

- cuando la pregunta contiene términos técnicos exactos del material, la recuperación léxica tiende a situar evidencia útil en posiciones altas;
- cuando la pregunta reformula conceptos del material, la recuperación híbrida tiende a mejorar la cobertura de evidencia útil dentro del *top-5*;
- cuando la pregunta queda fuera del corpus, las puntuaciones bajas de recuperación activan la abstención;
- en preguntas mixtas, el sistema puede recuperar evidencia para la parte cubierta, pero no para la parte ausente del corpus;
- en materiales visualmente ricos o en portadas, los fragmentos con contexto pobre limitan la calidad de la recuperación.

Estas trazas son interpretativas y agregadas, no registros literales de ejecución.

F.5. Tablas extendidas

Esta sección recoge, en forma de tablas compactas, los resultados agregados validados que complementan las tablas principales del Capítulo 7.

Tabla F.1: Caracterización final sobre la configuración híbrida $k = 20$, *top-k* 5, umbral 0,52 y *reranker* desactivado. La verificación de citas es estructural, no semántica.

Indicador	**Valor**
Consultas totales	106
Respuestas generadas	80
Abstenciones	26
Errores técnicos	0
Respuestas con fuentes	80/80
Respuestas con al menos una cita	79/80
Verificación estructural estricta	18/80
Respuestas con marcadores duplicados	61/80
Sin incidencias graves ignorando solo duplicados	78/80
Fugas de *prompt*	0/80

Tabla F.2: Comparación confirmatoria del *reranker* sobre las 58 preguntas puntuables, con el mismo conjunto de candidatos.

Configuración	**P@5**	**R@5**	**MRR**	**nDCG@5**	**Lat. media**
Híbrida RRF $k = 20$	0.4207	0.6813	0.6943	0.6261	23.3 ms
Híbrida + Qwen3-Reranker-0.6B	0.4000	0.6456	0.6957	0.5996	8.66 s

ΔnDCG@5 $= -0{,}0265$; IC *bootstrap* 95 % $[-0{,}1349; +0{,}0807]$. En este contraste, ambas variantes parten de `candidate_k=20` y devuelven `top_k=5`; por ello, la latencia del control no es directamente comparable con la obtenida en el barrido previo de la Tabla 7.1.

Tabla F.3: Evaluación RAGAS con cobertura homogénea y denominadores explícitos. La media es descriptiva y no estandarizada. Ninguna comparación pareada excluyó 0.

Configuración	**Fidelidad**	**Relevancia**	**Precisión**	**Media**
Densa	0.905 (n=44)	0.399 (n=58)	0.983 (n=58)	0.7625
BM25	0.890 (n=44)	0.323 (n=58)	0.962 (n=58)	0.7249
Híbrida $k = 20$	0.840 (n=44)	0.328 (n=58)	0.953 (n=58)	0.7068

El *reranking* queda implementado, pero desactivado por defecto: su contraste cerrado (Tabla F.2) no muestra una mejora concluyente y multiplica la latencia de recuperación

Tabla F.4: Fiabilidad de los jueces LLM locales (κ ponderada cuadrática). $N = 49$. Ninguna dimensión juez–humano alcanzó κ moderado.

Dimensión	Mistral–humano	Qwen–humano	Mistral–Qwen	N
Faithfulness	0.2868	0.4019	0.4917	49
Answer relevance	0.2647	0.0926	−0.0134	49
Citation accuracy	−0.2374	−0.0348	0.7019	49
Citation consistency	0.1873	0.2933	0.6920	49
Overall	0.0565	0.2331	0.7592	49

por aproximadamente 371, por lo que no se presenta como componente ganador. La política de *answerability* permanece implementada, desactivada por defecto y no reevaluada bajo la configuración final.