Universidad Zaragoza

Trabajo Fin de Grado

RAG para facilitar el uso de DATUZ y el aprendizaje de almacenes de datos

RAG to facilitate the use of DATUZ and learning about data warehouses

Autor

de Vera Sanz, Carlos

Directores

Ilarri Artigas, Sergio

Gracia Latorre, María Belén

Escuela de Ingeniería y Arquitectura

2024-2025

Agradecimientos

La culminación de este Trabajo de Fin de Grado ha sido posible gracias al apoyo y la colaboración de numerosas personas y entidades, a quienes deseo expresar mi más sincero agradecimiento.

En primer lugar, agradezco a mis profesores por su dedicación y por haberme transmitido los conocimientos y valores fundamentales a lo largo de la carrera.

Mi gratitud especial a mis directores, Sergio Ilarri y María Belén Gracia, por su orientación, paciencia y confianza durante todo el proceso. Su guía ha sido esencial para la realización de este trabajo.

A mis compañeros de clase, por el compañerismo y el apoyo mutuo, que han hecho de esta etapa una experiencia enriquecedora.

A mi familia y amigos, por su apoyo incondicional y por estar siempre a mi lado, brindándome ánimo y comprensión en los momentos más exigentes.

Este proyecto se enmarca en el PID2020-113037RB-I00, financiado por el MICIU/AEI/10.13039/501100011033 (proyecto NEAT-AMBIENCE) y liderado por la Universidad de Zaragoza. Agradezco la oportunidad de haber formado parte de este equipo y de contribuir a sus objetivos de investigación.

Resumen

La gestión eficiente de los sistemas de inteligencia de negocio es un reto clave para el aprovechamiento de los datos institucionales. En la Universidad de Zaragoza se identifican dos necesidades principales: facilitar el acceso intuitivo a la información estratégica para el personal no técnico y proporcionar entornos prácticos que favorezcan la formación en análisis de datos para el alumnado.

Este trabajo aborda ambos desafíos mediante una doble solución. Se desarrolla RAGDWAREuz, un asistente conversacional basado en la arquitectura de Generación Aumentada por Recuperación (*RAG*), capaz de interpretar preguntas en lenguaje natural, consultar una base de conocimiento construida a partir de la documentación oficial de los cubos de datos de SEGEDA y generar respuestas precisas y fundamentadas. La implementación se apoya en el *framework* LangGraph y su eficacia se valida mediante herramientas automáticas como DeepEval y evaluación humana.

Se diseña también MDX LEARN, un entorno educativo autocontenido que integra una infraestructura en memoria basada en Atoti, un conjunto de datos sintéticos que replica la estructura del *Data Warehouse* de DATUZ (garantizando la privacidad), la interfaz de Saiku y una serie de *notebooks* de Jupyter que guían al estudiantado en el aprendizaje progresivo de MDX.

Los resultados obtenidos avalan la efectividad de ambas soluciones. RAGDWAREuz se consolida como una interfaz intuitiva que facilita el acceso a la información institucional, demostrando la viabilidad de implementar soluciones de IA avanzadas con un coste razonable. MDX LEARN se presenta como una herramienta eficaz para la formación técnica en análisis de datos multidimensionales.

Este trabajo aporta dos herramientas funcionales que resuelven necesidades reales en el ámbito universitario, facilitando tanto el acceso a la información como la formación en tecnologías clave.

El desarrollo realizado, enmarcado en la colaboración entre los proyectos NEAT-AMBIENCE (PID2020-113037RBI00) y DATUZ (Datos Abiertos y Transparencia de la Universidad de Zaragoza), sienta las bases para futuras investigaciones y desarrollos en el equipo, así como para posibles adaptaciones e implantaciones de la propuesta. MDX LEARN puede resultar de gran utilidad en asignaturas del Grado en Ingeniería Informática como “Almacenes y Minería de Datos”.

Índice general

1. Introducción **1**

1.1. Contextualización del problema 1

1.2. Descripción del problema 2

1.3. Objetivos 2

1.4. Metodología y organización del proyecto 2

2. Fundamentos Teóricos **5**

2.1. *Large Language Models* (LLMs) 5

2.2. *Retrieval Augmented Generation* (RAG) 5

2.3. Técnicas de evaluación en sistemas **RAG** 9

2.4. Análisis multidimensional y OLAP: cubos, dimensiones, medidas 11

2.5. Tecnologías de interés para el proyecto 12

3. Análisis y Diseño del Sistema **15**

3.1. Arquitectura general y decisiones tecnológicas 15

3.2. Diseño del sistema **RAG** (**RAGDWAREuz**) 17

3.3. Diseño del entorno educativo (**MDX LEARN**) 21

4. Implementación **25**

4.1. Desarrollo del sistema RAGDWAREuz 25

4.2. Desarrollo del entorno MDX LEARN 29

4.3. Mantenibilidad y escalabilidad de RAGDWAREuz 30

5. Evaluación y validación de RAGDWAREuz **33**

5.1. Datos de evaluación para DeepEval 33

5.2. Evaluación granular y el enfoque *LLM as a Judge* 33

5.3. Evaluación con expertos 35

5.4. Conclusiones 36

6. Conclusiones y trabajo futuro **37**

6.1. Viabilidad de RAGDWAREuz 37

6.2. Cronograma del proyecto 38

6.3. Conclusiones principales 39

6.4. Limitaciones del trabajo 39

6.5. Líneas de trabajo futuro 39

Índice de figuras **41**

Índice de cuadros **42**

Bibliografía **43**

Anexos **45**

A. Glosario de términos **46**

B. Herramientas y Tecnologías **49**

B.1. Ollama 49

B.2. HuggingFace 49

B.3. LLMs utilizados 49

B.4. Componentes de Haystack 50

B.5. Ecosistema LangChain y LangSmith 51

B.6. Interfaz de Usuario con Chainlit 51

C. Implementación de RAGDWAREuz **52**

C.1. Diagramas de componentes y secuencia de RAGDWAREuz 52

C.2. Diagrama de secuencia del *workflow* principal completo 55

C.3. Repositorio del Proyecto 57

C.4. Parser de documentos DATUZ 57

C.5. Documentos procesados de DATUZ en formato Markdown 57

C.6. Carga de documentos en Milvus 57

C.7. Análisis del corpus con NLTK 57

C.8. Chain of Thought (CoT) en RAGDWAREuz 58

C.9. Prompt Templates 60
C.10. Ejemplos de ejecución en LangSmith 60
C.11. Capturas de Pantalla de la Interfaz 61

D. Implementación de MDX LEARN **62**

D.1. Repositorio del Proyecto 62
D.2. Esquemas de Oracle 62
D.3. Generación de datos sintéticos con Faker 62
D.4. Notebooks de Jupyter 62
D.5. Saiku 63

E. Evaluación del RAGDWAREuz **66**

E.1. Creación de tests con DeepEval 66

F. Conceptos Teóricos Adicionales **69**

F.1. Diagramas de secuencia de arquitecturas RAG avanzadas 69
F.2. El proceso de embeddings y búsqueda semántica 70

Capítulo 1

Introducción

Este trabajo aborda dos retos fundamentales en el ámbito universitario: democratizar el acceso a los datos analíticos para el personal no técnico y promover el aprendizaje de herramientas de análisis de datos. La solución propuesta es dual: un asistente conversacional basado en arquitecturas RAG (*Retrieval Augmented Generation*), que combina la capacidad generativa de los LLMs con la recuperación de información, y un entorno educativo dedicado. El proyecto se enmarca en la iniciativa de investigación NEAT-AMBIENCE[1], orientada a aplicar avances en la gestión de datos para mejorar la toma de decisiones en organizaciones como la Universidad de Zaragoza.

1.1. Contextualización del problema

Las instituciones educativas generan grandes volúmenes de datos. La Universidad de Zaragoza, a través de DATUZ (*Datos Abiertos y Transparencia de la Universidad de Zaragoza*) y el Servicio de Gestión de Datos (SEGEDA), busca mejorar la accesibilidad de sus datos para optimizar la gestión.

SEGEDA ofrece un entorno analítico basado en Pentaho Server con Saiku que permite consultas sobre cubos OLAP, pero presenta limitaciones significativas. Su aprovechamiento se ve restringido por la complejidad técnica de la herramienta y del dominio de los datos. La información se estructura en ocho ámbitos temáticos (Académico, I+D+i, Recursos Humanos, etc.), lo que añade complejidad para los usuarios no familiarizados.

Los usuarios, como el personal docente o el Personal Técnico, de Gestión y de Administración y Servicios (PTGAS), interactúan con los datos a través de la interfaz de Saiku. Sin embargo, a menudo desconocen la terminología específica o la estructura de los cubos de datos (p. ej., qué significa exactamente el término .[eg]resado.[en] este contexto o cómo se relacionan los distintos indicadores), lo que dificulta la formulación de consultas efectivas.

Esta barrera se extiende al ámbito formativo. El estudiantado de informática, que actualmente no tiene acceso a DATUZ, encuentra dificultades para aprender MDX (*MultiDimensional eXpressions*) de manera práctica debido a la falta de infraestructuras de BI (*Business Intelligence*) accesibles. La inteligencia artificial conversacional surge como solución para simplificar el acceso a la información analítica, mientras que un entorno de aprendizaje autocontenido puede superar los obstáculos para la formación en MDX.

[1] `http://webdiis.unizar.es/~silarri/NEAT-AMBIENCE/`

1.2. Descripción del problema

La colaboración entre el proyecto de investigación NEAT-AMBIENCE y el equipo de gestión de datos de la Universidad de Zaragoza identificó la necesidad de mejorar la usabilidad de SEGEDA. Los usuarios autorizados manifiestan dificultades al realizar consultas. Para solucionarlo, se propuso desarrollar un sistema conversacional capaz de interpretar preguntas en lenguaje natural y generar respuestas.

El sistema debe responder a una variedad de cuestiones, desde consultas sobre la documentación de los datos hasta la recuperación de información específica. Por ejemplo:

- *"¿Qué es un egresado según la documentación de DATUZ?"* (pregunta sobre el dominio).
- *"¿Cuántos estudiantes de nuevo ingreso hubo en el grado de Física en el curso 2022-2023?"* (pregunta que requiere recuperación de datos).
- *"Muéstrame una comparativa de la tasa de éxito por centro en el último año"* (pregunta que podría implicar la generación de una visualización o tabla).

Este enfoque busca clarificar la semántica de los datos y facilitar su extracción, sin abordar la generación automática de gráficas complejas, que queda fuera del alcance de este proyecto. Para fomentar el aprendizaje de MDX entre el estudiantado y otros interesados, se planteó crear un entorno educativo autocontenido que permita practicar consultas OLAP sin depender de infraestructuras complejas, superando la falta de entornos prácticos accesibles.

1.3. Objetivos

Este trabajo persigue dos metas principales: facilitar el acceso a la inteligencia de negocio para usuarios autorizados de SEGEDA y promover la capacitación en análisis de datos del estudiantado. Los objetivos específicos son:

- Investigar arquitecturas RAG avanzadas (modular, correctivo, adaptativo) y *frameworks* como LangChain para seleccionar la tecnología más adecuada.
- Diseñar y desarrollar RAGDWAREuz, un sistema conversacional que interprete preguntas en lenguaje natural, consulte la base de conocimiento y genere respuestas precisas.
- Crear MDX LEARN, un entorno educativo complementario con un entorno OLAP autocontenido que permita aprender y practicar consultas MDX, así como experimentar con la interfaz de Saiku.
- Implementar un sistema de evaluación robusto con el *framework* DeepEval y métricas de calidad para sistemas RAG, como la fidelidad (*faithfulness*), la relevancia de la respuesta (*answer relevancy*) y la precisión del contexto (*contextual precision*).
- Validar el prototipo RAG mediante casos de prueba y evaluación humana, asegurando su efectividad y usabilidad.

El cumplimiento de estos objetivos garantiza un desarrollo y validación sistemáticos de ambos componentes: RAGDWAREuz y MDX LEARN.

1.4. Metodología y organización del proyecto

La metodología se estructura en varias etapas: investigación de tecnologías RAG y marcos de evaluación para LLMs, desarrollo de un prototipo conversacional que interactúa con la documentación y la base de datos de SEGEDA, y diseño de herramientas OLAP para el entorno educativo.

La memoria se organiza en seis capítulos y anexos. El Capítulo 1 introduce el contexto, el problema y los objetivos del trabajo. El Capítulo 2 presenta los fundamentos teóricos sobre LLMs, RAG y análisis multidimensional OLAP/MDX, así como el contexto tecnológico. El Capítulo 3 detalla el análisis y diseño del sistema, justifica las decisiones tecnológicas y describe la arquitectura de RAGDWAREuz y MDX LEARN.

El Capítulo 4 documenta la implementación práctica de ambos sistemas, incluyendo la estructura de la base de conocimiento, el flujo conversacional y el entorno educativo. El Capítulo 5 presenta la evaluación y validación del sistema RAG con métricas y resultados. El Capítulo 6 recoge las conclusiones principales, limitaciones identificadas y líneas de trabajo futuro. Los Anexos complementan la memoria con información técnica adicional y recursos de apoyo.

Capítulo 2

Fundamentos Teóricos

Este capítulo establece los fundamentos conceptuales y el contexto tecnológico del trabajo. Se introducen los *Large Language Models* (LLMs), sus capacidades y limitaciones, como las alucinaciones (Sección 2.1). También se presenta el paradigma *Retrieval Augmented Generation* (RAG) como solución a dichas limitaciones (Sección 2.2). Se describen arquitecturas y técnicas avanzadas de RAG, incluyendo los enfoques correctivo y adaptativo. A continuación, se abordan las técnicas de evaluación en sistemas RAG (Sección 2.3) y los conceptos clave de análisis multidimensional y OLAP (Sección 2.4). Finalmente, se presenta el contexto tecnológico, detallando las herramientas y *frameworks* clave utilizados en el proyecto.

2.1. *Large Language Models* (LLMs)

Los *Large Language Models* (LLMs) son modelos de aprendizaje automático que procesan grandes volúmenes de texto para realizar tareas de procesamiento de lenguaje natural (PLN), principalmente la generación de lenguaje. Los más avanzados están basados en arquitecturas *transformers* generativas preentrenadas (GPTs) [27].

Su evolución ha sido muy rápida, aumentando exponencialmente su tamaño y datos de entrenamiento. Los LLMs actuales no solo generan texto coherente, sino que también comprenden el lenguaje, realizan razonamientos simples, mantienen conversaciones fluidas y aprenden de nuevos ejemplos en contexto (*In-context Learning*). Además, los modelos más recientes integran múltiples tipos de datos, como texto, imágenes y audio (multimodalidad) [24].

Sin embargo, los LLMs presentan limitaciones importantes. Las alucinaciones ocurren cuando el modelo genera información plausible pero incorrecta, ya que su diseño se basa en la predicción de patrones y no en la verificación de hechos [23]. Otra limitación es el conocimiento desactualizado, pues su información está limitada a la fecha de corte de sus datos de entrenamiento. El paradigma RAG aborda estas carencias al conectar los LLMs con bases de conocimiento externas y actualizadas [1].

2.2. *Retrieval Augmented Generation* (RAG)

El paradigma *Retrieval Augmented Generation* (RAG) surge para mitigar las limitaciones de los LLMs, como las alucinaciones y el conocimiento desactualizado. RAG combina la capacidad generativa de los LLMs con la recuperación de información [2] desde una base de conocimiento externa. Esto permite generar respuestas más precisas y fundamentadas en datos actualizados [1, 9].

Un componente clave en este proceso son los *embeddings* vectoriales, que representan numéricamente el texto en un espacio de alta dimensión, capturando sus relaciones semánticas. Los *embeddings*

permiten comparar la similitud entre fragmentos de texto, facilitando la recuperación de información relevante en respuesta a consultas del usuario. La Figura 2.1 ilustra este proceso de creación y uso (para más detalles, véase el Anexo F.2).

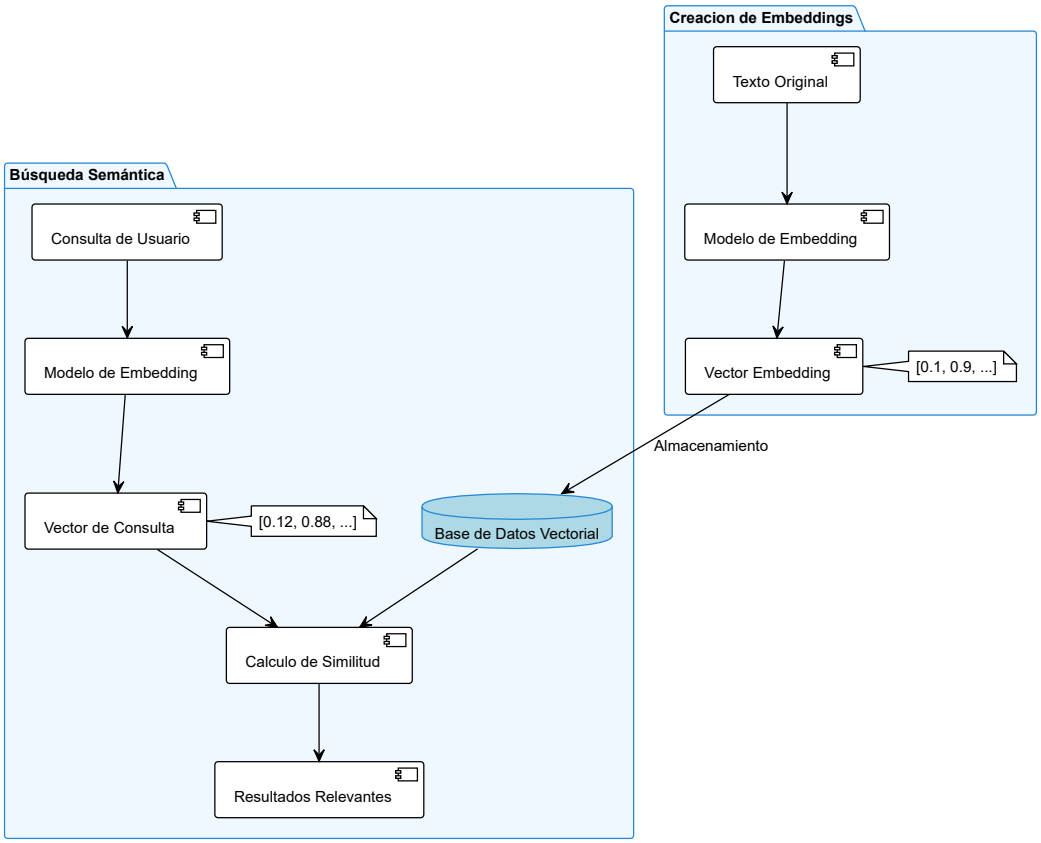

Figura 2.1: Proceso de creación y uso de *embeddings* vectoriales para búsqueda semántica

El flujo de trabajo de RAG se divide en dos fases. Primero, en la fase de recuperación, el sistema busca y extrae la información más relevante de la base de conocimiento en respuesta a una consulta del usuario. La Figura 2.2 ilustra este proceso. A continuación, en la fase de generación, los fragmentos recuperados se entregan al LLM junto con la consulta original. El modelo utiliza este contexto para formular una respuesta coherente y basada en hechos, como se muestra en la Figura 2.3. Esta arquitectura modular es fundamental para aplicaciones que exigen alta precisión factual [8].

2.2.1. Arquitecturas RAG clásicas versus avanzadas

Inicialmente, las implementaciones de RAG, conocidas como *RAG clásico* o *naive RAG*, adoptaban un enfoque simple. En este modelo, el sistema recupera fragmentos de información basados en similitudes básicas y los agrega a la consulta para que el LLM genere una respuesta. Aunque funcional en ciertos casos, este enfoque presenta limitaciones notables, como una recuperación potencialmente ineficiente para consultas complejas y una dependencia de un conocimiento estático. La Figura F.1 (incluida en el Anexo F.1) muestra este flujo de trabajo básico.

Para superar estas deficiencias han surgido enfoques RAG avanzados, que utilizan métodos más sofisticados para mejorar la precisión y relevancia de la recuperación. Entre estos enfoques destacan el *RAG avanzado* y el *RAG modular* [16]. El *RAG avanzado* aplica estrategias como la expansión de consultas y la recuperación en varias etapas [16]. Su flujo de trabajo se observa en la Figura F.3 (incluida en el Anexo F.1). Por su parte, el *RAG modular* permite dividir el proceso en módulos independientes que se pueden ajustar según las necesidades de cada aplicación, tal como se muestra en su arquitectura en la Figura F.2 (incluida en el Anexo F.1).

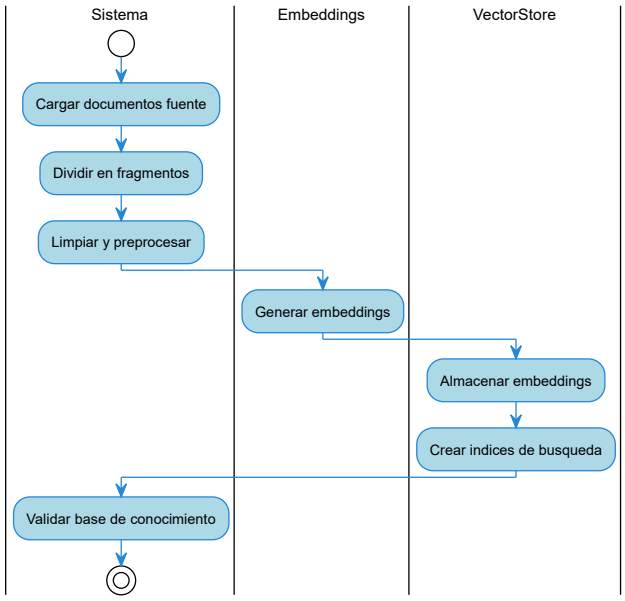

Figura 2.2: Proceso de la fase de recuperación en sistemas RAG

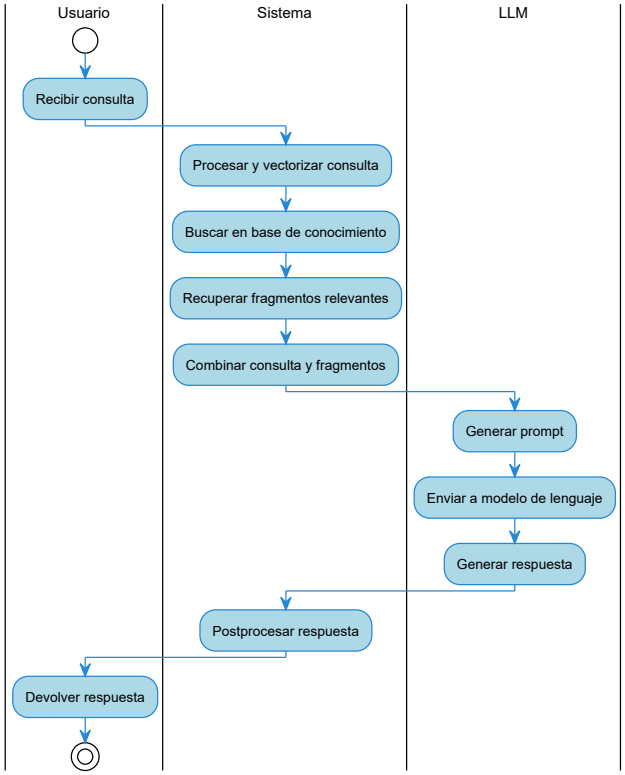

Figura 2.3: Proceso de la fase de generación en sistemas RAG

2.2.2. Técnicas avanzadas de RAG

Más allá de las arquitecturas básicas, diversas técnicas especializadas optimizan el rendimiento de los sistemas RAG. Esta sección examina tres de estas técnicas fundamentales: la reescritura de consultas, la recuperación de contexto y el *mix* de granularidad.

La reescritura de consultas (*Query Rewriting*) tiene como objetivo optimizar la consulta inicial del usuario para maximizar la relevancia de la información recuperada. Este proceso puede implicar la

adición de términos relacionados o la descomposición de consultas complejas en partes más simples, alineando así la consulta procesada con la intención real del usuario.

Ejemplo práctico de reescritura de consultas:

- *Consulta original*: "¿Cómo funciona la IA?"
- *Consulta reescrita*: "¿Cómo funcionan los algoritmos de inteligencia artificial, *machine learning* y redes neuronales? Explicar arquitecturas, entrenamiento y aplicaciones."
- *Resultado*: La consulta expandida incorpora términos técnicos específicos que mejoran la precisión de la recuperación semántica.

La recuperación de contexto (*Context Retrieval*) se refiere a las estrategias empleadas para extraer los fragmentos de información más pertinentes del corpus documental. Más allá de la búsqueda por similitud semántica básica, estas técnicas pueden incluir la recuperación jerárquica, la recuperación recursiva, la búsqueda híbrida [16] y el *chunking*.

El *chunking* es el proceso de dividir grandes documentos en fragmentos más pequeños y manejables, denominados *chunks*. Esta técnica es fundamental en RAG porque los LLMs tienen una ventana de contexto limitada, es decir, solo pueden procesar una cantidad finita de texto a la vez.

Al dividir la información en *chunks*, se facilita que el sistema de recuperación identifique y proporcione al LLM las porciones de texto más relevantes para una consulta específica. La efectividad del *chunking* depende de la estrategia utilizada para dividir el texto (por ejemplo, por tamaño fijo, por frases, o mediante técnicas más sofisticadas que consideran la estructura semántica del documento).

La recuperación jerárquica indexa los documentos en diferentes granularidades y recupera los niveles apropiados. La recuperación recursiva refina las recuperaciones en múltiples etapas. La búsqueda híbrida combina búsqueda vectorial con búsqueda de palabras clave para mayor cobertura. La compresión de contexto post-recuperación reduce el ruido y la redundancia en los fragmentos recuperados antes de pasarlos al LLM [7].

Ejemplo práctico - Recuperación jerárquica en RAGDWAREuz:

1. *Nivel Documento*: "Cubo MATRÍCULA - Información completa sobre medidas y dimensiones académicas".
2. *Nivel Sección*: "Dimensiones - Centro: clasificación por código, denominación, campus y tipo de centro".
3. *Nivel Párrafo*: "Centro - Campus: permite clasificar los datos en función del campus en que se ubica el centro de matrícula de los estudiantes".
4. *Resultado*: El sistema recupera contexto específico de SEGEDA manteniendo la estructura jerárquica del modelo de datos universitario.

La técnica de *mix de granularidad* (*Mix of Granularity* o MoG) ajusta la granularidad de los fragmentos de texto recuperados según la naturaleza de la consulta [28]. En lugar de recuperar fragmentos de tamaño fijo, MoG intenta determinar el nivel de detalle más adecuado para cada pregunta, que puede variar desde párrafos completos hasta oraciones específicas. MoG puede utilizar un componente de enrutamiento (*router*) entrenado para seleccionar la granularidad óptima basada en la consulta.

Ejemplo práctico - MoG en el cubo de Matrícula:

1. *Consulta*: "¿Qué significa la medida Alumnos de Nuevo Ingreso en el cubo de Matrícula?".
2. *Granularidad seleccionada*: Fragmentos pequeños (definición específica de la medida).

3. *Resultado*: Recupera la definición precisa "Número de estudiantes matriculados en un plan de estudios que consumen alguna de las plazas ofertadas para un año y plan (preinscripción)" en lugar de descripciones generales sobre matriculación.

Estas técnicas avanzadas permiten a los sistemas **RAG** adaptarse dinámicamente a diferentes tipos de consultas, mejorando significativamente la precisión y relevancia de las respuestas generadas.

2.2.3. RAG correctivo y adaptativo

Las arquitecturas RAG continúan evolucionando para enfrentar desafíos específicos en el rendimiento de los LLMs. Esta sección examina dos enfoques recientes que buscan mejorar la robustez y adaptabilidad del sistema: el ***RAG correctivo*** (CRAG) y el ***RAG adaptativo*** (Adaptive-RAG).

El ***Corrective Retrieval Augmented Generation*** (CRAG) tiene como objetivo mejorar la calidad de la recuperación y la robustez del sistema frente a recuperaciones irrelevantes o insuficientes [26]. CRAG introduce un mecanismo de "evaluación" o "juicio" que valora la calidad del texto recuperado.

Si el texto se considera de baja calidad o insuficiente, el sistema puede optar por estrategias de recuperación alternativas, como la **reescritura de la consulta** o la consulta a fuentes de datos adicionales. Esto permite que el sistema realice autocorrecciones, minimizando el impacto de recuperaciones erróneas en la calidad de la respuesta generada [26]. La Figura F.4 ilustra este flujo de trabajo (incluida en el Anexo F.1).

El ***Adaptive Retrieval Augmented Generation*** (Adaptive-RAG) adapta la estrategia de recuperación según la complejidad de la pregunta del usuario [12]. Este enfoque reconoce que no todas las preguntas requieren la misma profundidad o amplitud de recuperación. Adaptive-RAG aprende a decidir dinámicamente si es necesaria la recuperación, y en qué medida, para una consulta específica. Su flujo de trabajo se detalla en la Figura F.5 (incluida en el Anexo F.1).

Self-RAG propone un modelo que aprende a recuperar y generar información de manera reflexiva, evaluando la calidad de los documentos recuperados e identificando errores para ajustar el proceso cuando sea necesario [3]. Este flujo de trabajo se presenta en la Figura F.6 (incluida en el Anexo F.1).

2.3. Técnicas de evaluación en sistemas RAG

Evaluar la calidad en sistemas RAG resulta crucial para asegurar su fiabilidad y eficacia. Las métricas utilizadas varían según se evalúe la **fase de recuperación** o la de **generación**, y también valoran propiedades como la **precisión** y la **relevancia**.

Para la **fase de recuperación**, las métricas comunes son:

- **Relevancia contextual *(Contextual Relevancy)*:** relevancia de los fragmentos recuperados para la pregunta.
- **Precisión contextual *(Contextual Recall)*:** completitud de los hechos necesarios para responder la pregunta.

Para la **fase de generación**, y la calidad general de la respuesta, se utilizan métricas como:

- **Fidelidad *(Faithfulness):*** determina si la respuesta generada es consistente con la información proporcionada por el contexto recuperado [23]. Una baja fidelidad indica **alucinaciones**.
- **Relevancia de la respuesta (*Answer Relevancy*):** evalúa si la respuesta generada es directamente pertinente a la pregunta del usuario.

- **Exhaustividad *(Answer Recall)*:** mide si la respuesta aborda todos los aspectos de la pregunta.

Una combinación de estas métricas, junto con la evaluación humana, puede ofrecer una visión integral del rendimiento [24].

2.3.1. *Human in the loop* versus *LLM as judge*

La evaluación de sistemas RAG es esencial para asegurar la calidad de las respuestas. En esta sección se comparan dos técnicas destacadas para esta tarea: la evaluación con intervención humana (***Human in the Loop*** o HITL) y el uso de un LLM como evaluador (***LLM as Judge***), analizando sus ventajas e inconvenientes respectivos.

Human in the Loop (HITL)

La técnica ***Human in the Loop*** (HITL) integra evaluadores humanos en el proceso de revisión de las respuestas del sistema RAG. Este enfoque es particularmente valioso en contextos críticos donde la **precisión** es fundamental [22]. Los evaluadores humanos pueden identificar sesgos, falta de contexto o errores sutiles que las métricas automáticas podrían no detectar.

Ejemplo práctico - HITL:

1. *Respuesta generada:* "La aspirina es segura para todos los pacientes".
2. *Evaluación humana:* el evaluador señala que esto es incorrecto para pacientes con úlceras gástricas, ajustando la respuesta y refinando el sistema para futuras consultas.
3. *Resultado:* mejora de la precisión y seguridad del sistema mediante supervisión experta.

HITL, aunque costoso en tiempo y recursos, puede ser indispensable para capturar matices y garantizar la calidad en aplicaciones de alto riesgo [14].

LLM as Judge

El enfoque ***LLM as Judge*** utiliza un modelo de lenguaje (LLM) para evaluar automáticamente las respuestas, basándose en criterios predefinidos como **relevancia** (*Answer Relevancy*), **exhaustividad** (*Answer Recall*) y **fidelidad** (*Faithfulness*) [22]. Este método es eficiente en términos de escalabilidad, aunque su fiabilidad puede ser menor que la evaluación humana si no se calibra y supervisa adecuadamente.

Ejemplo práctico - *LLM as Judge*:

1. *Respuesta generada:* "La aspirina es un analgésico común utilizado para reducir el dolor y la fiebre".
2. *Evaluación por LLM as Judge:* el LLM la califica como relevante, pero no detecta información faltante sobre efectos secundarios específicos.
3. *Resultado:* evaluación rápida pero potencialmente incompleta sin supervisión humana.

Se recomienda utilizar *LLM as Judge* como un complemento a HITL, especialmente en entornos donde la supervisión humana es limitada pero se necesita una evaluación rápida y a gran escala [22]. En resumen, HITL ofrece una evaluación detallada y contextualizada, pero es costosa y lenta. Por otro

lado, *LLM as Judge* es eficiente pero puede ser menos fiable sin una adecuada supervisión humana. En aplicaciones críticas, un enfoque híbrido que combine ambos métodos suele ser ideal para equilibrar precisión y eficiencia [14].

2.4. Análisis multidimensional y OLAP: cubos, dimensiones, medidas

El **Análisis multidimensional** es una metodología para organizar y examinar datos empresariales desde múltiples perspectivas. En el centro de este enfoque se encuentra el ***Online Analytical Processing* (OLAP)**, un tipo de *software* que facilita a los usuarios el acceso y análisis de datos de manera rápida e interactiva [6]. Los conceptos fundamentales en OLAP son:

- **Cubo OLAP:** estructura central que organiza los datos en múltiples dimensiones. No es un cubo físico, sino una forma de representar datos a través de varias dimensiones. Almacena datos pre-agregados, permitiendo consultas rápidas.
- **Dimensiones:** categorías o perspectivas que se usan para analizar los datos. Son atributos descriptivos como Tiempo (año, trimestre, mes), Producto (categoría, tipo), Geografía (país, ciudad) y Cliente (edad, demografía) [6].
- **Medidas:** valores numéricos o métricas que se analizan, como Ventas, Cantidad, Beneficio o Costo. Se agrupan según las dimensiones para facilitar la comprensión de los datos [13].

Interactuar con un cubo OLAP implica realizar operaciones como *drill-down* (descender a un nivel de mayor detalle), *roll-up* (obtener un resumen), *slice* (examinar una sección de un cubo) y *dice* (seleccionar un subconjunto específico de un cubo) [11].

2.4.1. Lenguaje *Multidimensional Expressions* (MDX)

Para consultar y manipular datos en cubos OLAP, se utiliza el lenguaje ***Multidimensional Expressions*** (MDX). MDX es el estándar de la industria para consultar bases de datos multidimensionales, desempeñando un papel análogo al de SQL (*Structured Query Language*) para las bases de datos relacionales [4]. **MDX** permite a los usuarios:

- **Definir consultas:** escribir sentencias para seleccionar datos de un cubo, especificar qué dimensiones y medidas incluir, y filtrar o ordenar los resultados.
- **Realizar cálculos complejos:** crear miembros calculados y conjuntos que, aunque no están explícitamente en el cubo, son necesarios para el análisis.
- **Navegar el cubo:** llevar a cabo las operaciones OLAP como *drill-down*, *roll-up*, *slice* y *dice* dentro de las consultas.

La sintaxis de **MDX** es diferente de **SQL**, ya que está diseñada para trabajar con la estructura jerárquica de los cubos. En **MDX** se usan *tuples* para indicar intersecciones de miembros a través de diferentes dimensiones, y conjuntos para definir grupos de miembros. Para ilustrar el uso práctico de MDX en el ámbito universitario, considérese la siguiente consulta que analiza el rendimiento académico por centro y titulación:

Ejemplo de consulta MDX en contexto académico:

```
SELECT
```

```
    [Measures].[Estudiantes Matriculados] ON COLUMNS,
    NON EMPTY [Centro].[Centro].MEMBERS *
              [Titulacion].[Titulacion].MEMBERS ON ROWS
FROM [Cubo_Academico]
WHERE ([Periodo].[Año].[2023], [Tipo_Estudiante].[Grado])
```

Esta consulta recupera el número de estudiantes matriculados (medida) para cada combinación de centro y titulación (dimensiones), filtrada para el año 2023 y estudiantes de grado. El operador `NON EMPTY` elimina filas sin datos, mientras que el símbolo `*` realiza el producto cartesiano entre las dimensiones Centro y Titulación, permitiendo un análisis cruzado del rendimiento académico por unidad organizativa.

2.5. Tecnologías de interés para el proyecto

El avance de las tecnologías en inteligencia artificial y análisis de datos está transformando la consulta de información académica. Esta sección examina las herramientas y ***frameworks*** clave del enfoque propuesto, desde bases de datos vectoriales como **Milvus** y *frameworks* para sistemas conversacionales como **LangChain** y **LlamaIndex**, hasta sistemas de evaluación como **DeepEval** y herramientas de análisis OLAP en memoria como **Atoti**. Su integración en este proyecto busca democratizar el acceso a la información académica mediante sistemas inteligentes.

2.5.1. Bases de datos vectoriales: Milvus

El procesamiento eficiente de *embeddings* a gran escala requiere bases de datos especializadas. Milvus es una solución de código abierto diseñada para la gestión de datos vectoriales en entornos dinámicos, con aplicaciones en recomendación y búsqueda semántica [19]. A diferencia de las bases de datos tradicionales, está optimizada para operaciones vectoriales, lo que la convierte en una herramienta fundamental para aplicaciones que dependen de búsquedas de similitud rápidas y precisas. Para este trabajo, se ha utilizado una instancia alojada en Docker que permite funcionalidades avanzadas como la búsqueda híbrida, que combina *embeddings* con búsqueda textual [18].

2.5.2. *Frameworks* de desarrollo y evaluación de sistemas conversacionales

El desarrollo de sistemas RAG complejos requiere *frameworks* que orquesten sus componentes. A continuación se presentan los más influyentes.

LangChain es una biblioteca de código abierto que facilita la creación de aplicaciones con LLMs al conectarlos con fuentes de datos externas. Ofrece una interfaz común para múltiples LLMs (***Model I/O***), componentes para la ingesta y recuperación de documentos (***Retrieval***), secuencias de llamadas a LLMs (***Chains***), sistemas que permiten a los LLMs usar herramientas (***Agents***) y mecanismos para mantener el estado de la conversación (***Memory***) [15].

Es importante aclarar que en este trabajo, el término **"agente"** se refiere específicamente a los *agents* del *framework* LangChain, que son componentes *software* que permiten a los LLMs acceder y utilizar herramientas externas de forma autónoma para resolver tareas complejas. Esta acepción difiere de la definición clásica de agentes *software* autónomos en sistemas multiagente [10], donde los agentes son entidades independientes que pueden comunicarse entre sí mediante protocolos específicos.

En nuestro contexto, los "agentes" funcionan como orquestadores que coordinan la interacción entre LLMs y herramientas específicas dentro de un flujo de trabajo RAG. Estas características convierten a LangChain en una solución integral para orquestar flujos de trabajo complejos en aplicaciones conversacionales.

Mientras LangChain ofrece una flexibilidad general, LlamaIndex se especializa en optimizar la recuperación e indexación de información [17]. Su arquitectura se centra en la ingesta, almacenamiento y consulta eficiente de datos para su uso en LLMs. LangChain ofrece un control más granular sobre la funcionalidad en cada paso del proceso, una gestión de la memoria más sofisticada para conversaciones largas y una mayor capacidad para manejar consultas complejas y estructuras de datos diversas mediante la combinación de diferentes técnicas de búsqueda.

Haystack es otro *framework* de Python de código abierto relevante, diseñado para construir aplicaciones personalizadas con LLMs, especialmente sistemas RAG robustos y listos para producción [20]. Su filosofía se basa en una arquitectura de **tuberías (pipelines)** modular, que encadena componentes para crear flujos de trabajo flexibles. Cada componente realiza una tarea específica, como la recuperación de documentos, la generación de texto o la conversión de datos. En el Anexo B.4 se detallan los componentes clave de Haystack.

La validación de la calidad en sistemas RAG requiere herramientas de evaluación especializadas. **DeepEval** es un *framework* de código abierto diseñado para esta tarea, que proporciona métricas para medir la precisión de la recuperación y la fidelidad de la generación. Ofrece métricas predefinidas como **Fidelidad** (***Faithfulness***), **Relevancia de la respuesta** (***Answer Relevancy***) y **Precisión contextual** (***Contextual Recall***) [5], tal y como se definen en la Sección 2.3, permitiendo también la personalización de métricas y casos de prueba. Su capacidad de integración con flujos de CI/CD y de rastreo de evaluaciones lo convierte en una herramienta esencial para garantizar la fiabilidad de los sistemas RAG.

2.5.3. Atoti para el análisis OLAP en memoria

El rendimiento es un factor crítico en el análisis OLAP. Las herramientas de **Análisis en Memoria** como Atoti [4] abordan este desafío cargando los datos en la memoria RAM para un acceso ultrarrápido. Atoti es un *framework* de código abierto que destaca por su capacidad para construir y consultar cubos OLAP en tiempo real. Sus características incluyen procesamiento en memoria, actualizaciones en tiempo real, APIs en Python para la creación de cubos y una fácil integración con diversas fuentes de datos y herramientas de visualización.

Esta combinación de características hace de Atoti una solución particularmente atractiva para entornos que requieren análisis rápido y flexible de grandes volúmenes de datos. En el contexto educativo, Atoti representa una herramienta valiosa para el aprendizaje práctico de MDX y conceptos OLAP. Su interfaz intuitiva y APIs en Python facilitan que los estudiantes experimenten con la construcción de cubos multidimensionales sin la complejidad asociada a la configuración de servidores OLAP tradicionales. Los estudiantes pueden crear rápidamente cubos de prueba, ejecutar consultas MDX y visualizar los resultados de forma inmediata, lo que acelera el proceso de comprensión de conceptos como jerarquías, dimensiones y operaciones de navegación.

Esta aproximación práctica resulta especialmente efectiva para consolidar el aprendizaje teórico de MDX con experiencia práctica (***hands-on***) en un entorno controlado y accesible. Por lo tanto, Atoti no solo ofrece ventajas técnicas en términos de rendimiento, sino que también proporciona un entorno educativo ideal para el aprendizaje de tecnologías OLAP modernas.

Capítulo 3

Análisis y Diseño del Sistema

Este capítulo detalla el análisis y diseño de RAGDWAREuz, un asistente conversacional, y MDX LEARN, un entorno para el aprendizaje de MDX. Se presenta la arquitectura general y las tecnologías adoptadas, se profundiza en el diseño de RAGDWAREuz y se describe la estructura de MDX LEARN.

3.1. Arquitectura general y decisiones tecnológicas

Construir un sistema que combine la recuperación de información con la generación de respuestas y facilite el aprendizaje de herramientas de análisis requiere una arquitectura híbrida. La combinación de LangChain, Milvus y Atoti constituye una base tecnológica óptima para democratizar el acceso a la inteligencia de negocio universitaria y la formación en análisis de datos. Este capítulo justifica cada decisión tecnológica con criterios de eficacia, escalabilidad y usabilidad educativa.

3.1.1. Selección de tecnologías y justificación

El proyecto utiliza Python como lenguaje principal, decisión fundamentada en la madurez de su ecosistema de inteligencia artificial, que incluye bibliotecas como LangChain, PyTorch, *Transformers* y DeepEval. Python ofrece una capa de abstracción que simplifica el desarrollo, mientras que las operaciones intensivas se ejecutan en bibliotecas optimizadas, garantizando un rendimiento adecuado.

3.1.2. Selección del motor de recuperación: Milvus

Para la recuperación de información, se eligió Milvus tras evaluar las principales alternativas del mercado: Chroma y Qdrant. Aunque Qdrant ofrece alto rendimiento con 150+ QPS y tres tipos de índices especializados, y Chroma destaca por su simplicidad e integración fluida para proyectos educativos, Milvus se impuso por su superioridad técnica en tres aspectos críticos para RAGDWAREuz.

Primero, su búsqueda híbrida avanzada combina recuperación semántica vectorial con búsqueda textual tradicional (BM25[1]), capacidad que ni Qdrant ni Chroma ofrecen con la misma profundidad. Esta funcionalidad es esencial para maximizar la cobertura de consultas en el dominio académico de SEGEDA.

Segundo, el soporte para 11 tipos de índices diferentes, incluyendo vectores binarios e indexación flexible en memoria o disco, supera las opciones más limitadas de Qdrant (3 índices) y Chroma (principalmente HNSW), permitiendo optimizar el rendimiento según las características específicas de cada tipo de consulta.

[1] `https://en.wikipedia.org/wiki/Okapi_BM25`

Tercero, la flexibilidad de esquema de Milvus facilita la incorporación de metadatos complejos (ámbito temático, cubo OLAP, versiones de documentos) sin comprometer el rendimiento, lo que resulta fundamental para la arquitectura jerárquica de documentos del sistema.

En la Tabla 3.1 se resumen las características clave de Milvus en comparación con Chroma y Qdrant.

Tabla 3.1: Comparativa de motores de recuperación vectorial: Milvus, Chroma y Qdrant

Característica	**Milvus**	**Chroma**	**Qdrant**
Escalabilidad	Diseñada para vectores a escala de billones, escalabilidad elástica y horizontal.	Escalable, pero para conjuntos de datos más pequeños o medianos.	Maneja grandes conjuntos de datos, menos probada a escala de billones.
Rendimiento	Soporta 11 índices, incluyendo vectores binarios, alto rendimiento para grandes datos.	Optimizada para búsquedas en tiempo real, ideal para audio, menos índices.	Alta QPS, baja latencia, pero podría carecer de características avanzadas.
Facilidad de Uso	SDKs en múltiples lenguajes, bien documentada, pero más configuración para principiantes.	Muy intuitiva, ideal para principiantes, flujo de trabajo sencillo.	Instalación fluida, documentación completa, intuitiva.
Características Clave	Búsqueda híbrida, consistencia fuerte, indexación flexible, 11 índices, vectores binarios.	Múltiples tipos de datos, HNSW, búsquedas en tiempo real, enfoque en audio.	Tres índices, búsqueda híbrida, alta QPS, filtrado de atributos.
Soporte Comunitario	Comunidad grande, establecida, más de 1000 usuarios empresariales, recursos extensos.	Comunidad pequeña, en crecimiento.	Comunidad en crecimiento, buena documentación, activa en GitHub.

Selección del *framework* RAG

Para construir un flujo conversacional avanzado con lógicas de autocorrección, se requiere un *framework* flexible. Tras analizar las alternativas, se concluyó que LangChain, y en particular su extensión **LangGraph**, ofrece el mejor equilibrio entre control, modularidad y capacidad de experimentación. A diferencia de LlamaIndex o Haystack, LangGraph facilita la creación de grafos cíclicos, capacidad fundamental para implementar arquitecturas complejas como Self-RAG.

En este trabajo, los **componentes especializados encapsulan herramientas y lógica de decisión**, permitiendo a los LLMs ejecutar tareas complejas. Si bien todas las herramientas son robustas, la flexibilidad de LangChain para definir flujos de trabajo no secuenciales fue el factor determinante. La Tabla 3.2 resume los criterios de decisión clave.

Tabla 3.2: Comparativa de *frameworks* RAG: LangChain, LlamaIndex y Haystack

Característica	**LangChain**	**LlamaIndex**	**Haystack**
Enfoque Central	Flujos LLM personalizados, orquestación modular.	Orquestación de datos, recuperación optimizada.	Tuberías NLP/LLM modulares, RAG productivo.
Implementación RAG	Control granular completo de fragmentación e incrustación.	Especializado en ingestión e indexación eficiente.	Componentes modulares para *pipelines* completos.
Capacidades Avanzadas	Muy Alta: LangGraph permite grafos cíclicos y control granular.	Alta: Ejecución introspectiva paso a paso.	Moderada-Alta: Decisiones sobre fuentes de conocimiento.
Self-RAG/Correctivo	Fuerte: LangGraph ideal para bucles de retroalimentación.	Moderado: Soporte para reflexión y autocorrección.	Moderado: Flexibilidad para enfoques correctivos.
Extensibilidad	Muy Extensa: +600 integraciones.	Muy Extensa: +160 conectores.	Buena: Integraciones LLM y documentos.
Madurez	Consolidado (2022), evolución rápida y activa.	Establecido (2022), enfoque especializado.	Maduro (2019), estable y orientado a producción.
Curva Aprendizaje	Moderada-Alta: Abstracciones complejas.	Moderada: Conceptos de indexación.	Moderada: Familiaridad con *pipelines*.

Selección del motor OLAP: Atoti

Para el entorno MDX LEARN, se eligió Atoti por su simplicidad y enfoque en la experiencia de aprendizaje. Atoti destaca por su facilidad de configuración sin servidores complejos, integración nativa con Python y Jupyter *Notebooks*, análisis en memoria para un rendimiento superior y una *API* intuitiva que reduce la curva de aprendizaje.

Estas características lo convierten en la solución idónea para un entorno educativo que busca eliminar barreras tecnológicas y centrarse en la enseñanza práctica de MDX. Aunque presenta limitaciones cuando los datos superan la RAM disponible, se pueden utilizar muestras representativas que mantengan las características principales para el aprendizaje de MDX y conceptos OLAP sin comprometer la experiencia educativa.

3.2. Diseño del sistema RAG (RAGDWAREuz)

El diseño de RAGDWAREuz se basa en una arquitectura modular con LangGraph que integra recuperación vectorial y generación aumentada. El sistema combina una organización jerárquica de documentos, un *pipeline* de procesamiento multi-etapa y un *workflow* conversacional adaptativo para responder a las necesidades del dominio académico. La Figura 3.1 muestra una vision general de los componentes que conforman el sistema, tanto los elementos opcionales como los obligatorios.

3.2.1. Organización jerárquica de los documentos

La información del sistema se estructura según el modelo de SEGEDA, reflejando la organización de la Universidad de Zaragoza. Cada documento se asocia a un cubo OLAP (con sus dimensiones y medidas, véase el Anexo C.5.1 para un ejemplo) y se clasifica en ámbitos temáticos (académico, I+D+i, RRHH, movilidad, etc.). Esta estructura permite al sistema filtrar las búsquedas por ámbito antes de aplicar técnicas semánticas, aumentando la precisión. Los documentos originales, en un formato tabular de difícil procesamiento, se convierten a **Markdown** con un *parser* (véase el Anexo C.4), organizando el contenido jerárquicamente para su correcta indexación en **Milvus**.

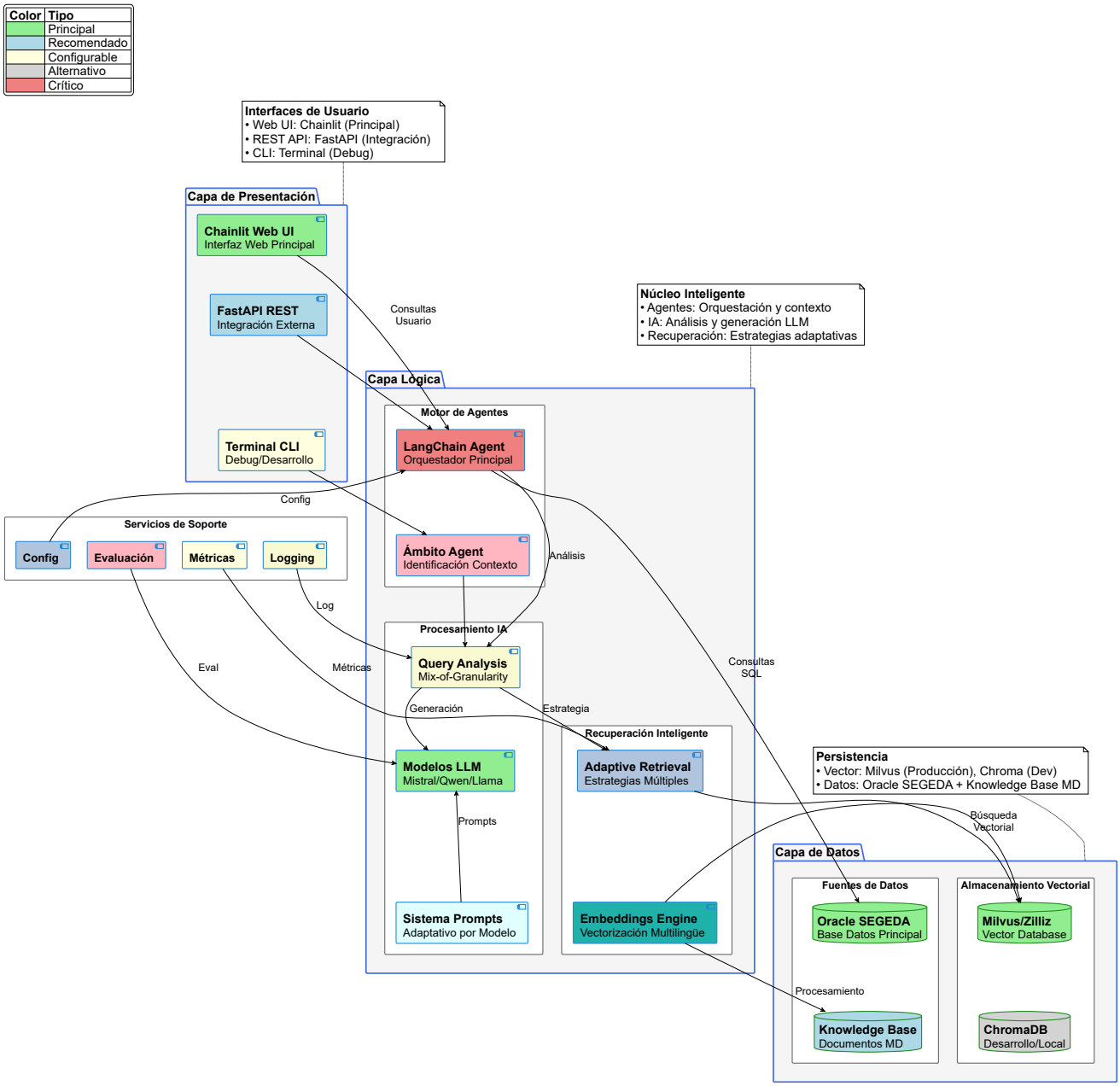

Figura 3.1: Diagrama de componentes de RAGDWAREuz.

3.2.2. *Pipeline* de procesamiento

El ***pipeline*** de procesamiento transforma las consultas en respuestas precisas mediante un flujo secuencial de cuatro etapas, cuyo diseño modular facilita la optimización independiente de cada componente.

Fase de preparación y procesamiento de los documentos

La inicialización del sistema prepara la base de conocimiento mediante tres procesos secuenciales: procesamiento de documentos, generación de contexto (opcional) e indexación. La Figura 3.2 ilustra esta fase. Finalmente, los documentos se dividen en fragmentos de texto (*chunks*) superpuestos para no perder contexto, y posteriormente se generan sus *embeddings* con un modelo de lenguaje especializado.

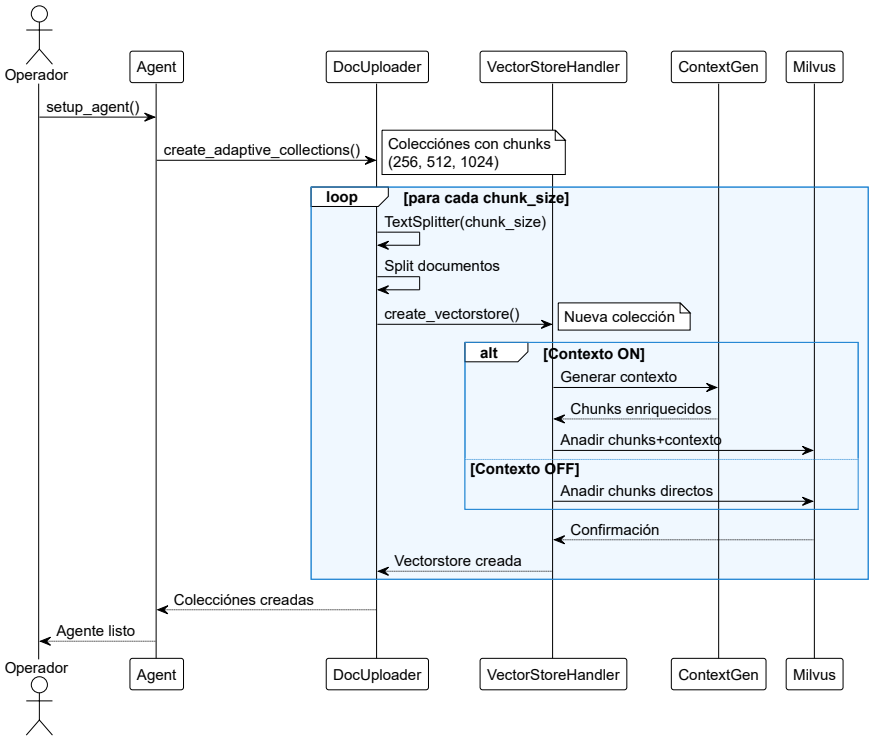

Figura 3.2: Diagrama del secuencia de preparación de la base de conocimiento.

Generación de contexto

Si esta opción se encuentra activada, el sistema ejecuta una fase de enriquecimiento semántico. El propósito de este paso es añadir una capa de inteligencia y comprensión contextual a cada fragmento de documento (*chunk*) antes de su indexación. Para ello, se utiliza una cadena específica de LangChain denominada `context_generator`, a la que se le proporciona tanto el documento completo como el *chunk* correspondiente.

Esto es posible gracias a la estructuración jerárquica de los documentos Markdown, cuya sintaxis refleja la lógica del modelo de datos de SEGEDA: una oración suele definir una medida o un atributo; un párrafo completo (delimitado por marcadores) describe una dimensión o medida en su totalidad; y una sección agrupa conjuntos de dimensiones o medidas (véase el Anexo C.5).

Para lograr este enriquecimiento, se invoca al LLM principal con un *prompt* de razonamiento estructurado. Este proceso implementa una estrategia que se adapta dinámicamente al tamaño y, por tanto, al nivel jerárquico del fragmento.

- Para ***chunks* pequeños** (a nivel de oración, como la descripción de "Grupo - Tipo"), el análisis se centra en la precisión semántica de ese atributo o métrica específica.
- Para ***chunk* de tamaño medio** (a nivel de párrafo, ej: la dimensión "Grupo" con su descripción y todos sus atributos), busca establecer el rol y las características de esa dimensión o medida completa.
- Para los ***chunks* más grandes** (a nivel de sección, como la sección completa de "Dimensiones"), el objetivo es comprender el marco general y la relación entre los distintos componentes del cubo.

Siguiendo una cadena de pensamiento guiada, el LLM identifica sistemáticamente el cubo de datos pertinente (p. ej., `GRUPOS DE INVESTIGACIÓN`), distingue entre `MEDIDAS` y `DIMENSIONES`, y extrae las métricas y la terminología institucional clave.

El resultado de esta etapa no es una lista de metadatos, sino un párrafo de texto conciso que encapsula el contexto del fragmento según su nivel jerárquico. En la Figura 3.3 se muestra un ejemplo de contexto enriquecido generado por el LLM.

```
[
    ‘‘context_generation’’: ‘‘The chunk discusses the metric ’Plazas asignadas a los alumnos OUT,’
    which represents the total number of mobility placements assigned to Unizar students for stays
     at other
    universities under bilateral agreements registered in SIGMA. This measure is specific to
    programs like ERASMUS,
    SICUE, Movilidad UE-Suiza, UZ/NOA, and Movilidad Iberoamérica, with data available from the
    2012/13 academic year
    onwards for some programs. The metric focuses on placements where all relevant student
    application details,
    placement assignments, and study contracts are recorded in SIGMA. This hierarchical context is
     at the sentence
    level due to the chunk size being less than 600 characters, emphasizing precise semantic
    meaning and
    immediate conceptual relationships within the SEGEDA framework.’’
]
```

Figura 3.3: Ejemplo de contexto enriquecido generado por el LLM.

Indexación en la base de conocimiento

Una vez procesados todos los documentos y generado el contexto enriquecido, se crean los *embeddings* de cada fragmento. Posteriormente, se añaden metadatos que incluyen el ámbito temático, el cubo de datos asociado, la ruta del documento original y un campo con el texto original. Este último campo permite utilizar la búsqueda híbrida de Milvus, que combina la búsqueda semántica con la de palabras clave para mejorar la precisión de los resultados.

Si este proceso se ejecuta por primera vez, se crea una colección nueva en la base de datos vectorial. Si la colección ya existe, se actualiza añadiendo únicamente los documentos nuevos o modificados. Un documento se actualiza si cumple una de estas dos condiciones: no existía previamente en la base de conocimiento o su versión ha cambiado (los documentos se versionan con el sufijo _vX).

Flujo principal de consulta

El proceso de respuesta a las consultas del usuario sigue una secuencia optimizada:

1. **Pre-recuperación de ámbito:** identificación del ámbito más relevante mediante análisis de palabras clave y recuperación ponderada.
2. **Recuperación vectorial:** búsqueda semántica en la base de conocimiento filtrada por ámbito.
3. **Evaluación de documentos:** filtrado de documentos irrelevantes mediante un ***document grader***.
4. **Generación de respuesta:** síntesis de información utilizando técnicas de RAG avanzadas.
5. **Evaluación correctiva:** se valida la calidad de la respuesta mediante un LLM independiente que intenta detectar alucinaciones y evalúa la relevancia de la respuesta.

Este flujo optimizado combina eficiencia y precisión para asegurar respuestas relevantes y basadas en la documentación oficial de SEGEDA. La evaluación correctiva actúa como una salvaguarda final contra posibles alucinaciones del modelo.En el Anexo C se detalla la arquitectura completa del sistema.

Las fases de recuperación y generación, núcleo del sistema, se han diseñado de forma modular y eficiente. La Figura 3.4 ilustra el proceso de recuperación, desde la consulta inicial hasta la selección de

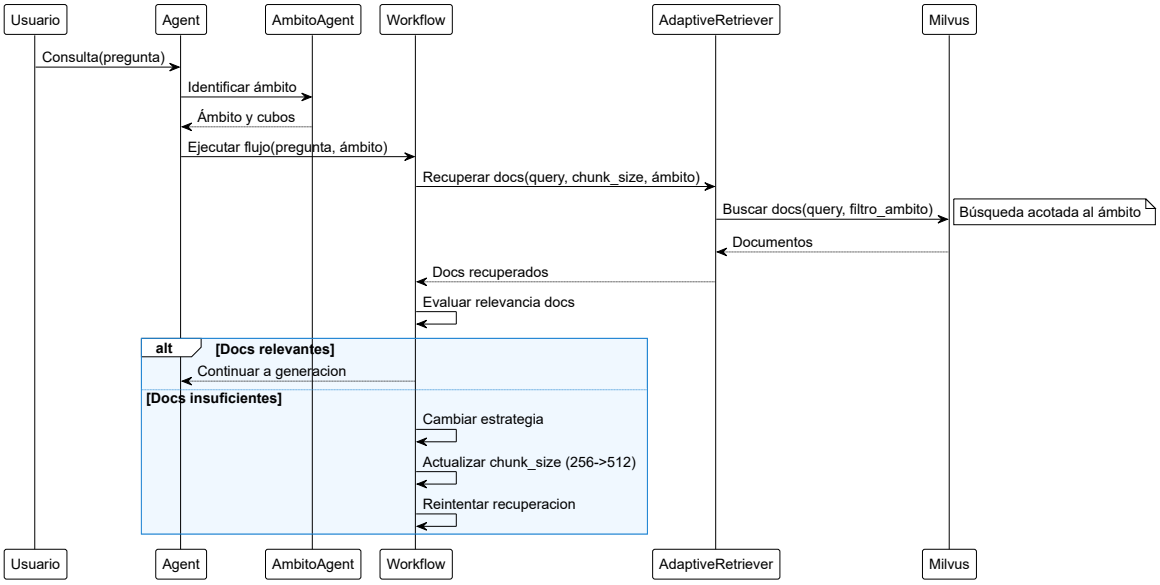

Figura 3.4: Diagrama del flujo de recuperación de información.

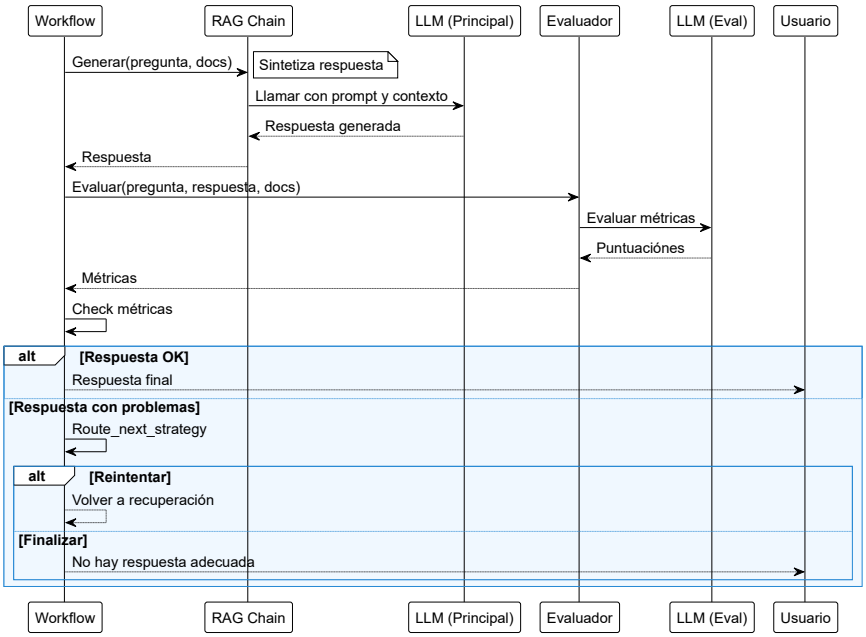

Figura 3.5: Diagrama del flujo de generación de respuesta.

documentos. Por su parte, la Figura 3.5 detalla cómo el sistema sintetiza la información para formular una respuesta final coherente y precisa.

En iteraciones anteriores del sistema, existía un componente de enrutamiento de preguntas (***question router***) embebido, que fue posteriormente extraído y evolucionado hacia el actual sistema de detección de ámbito. Esta modificación mejoró la modularidad y la precisión del proceso.

3.3. Diseño del entorno educativo (MDX LEARN)

La creación de un entorno educativo que facilite el aprendizaje práctico de MDX sin comprometer la privacidad de los datos institucionales se logra mediante una arquitectura modular que combina

datos académicos sintéticos con una infraestructura OLAP containerizada. Esta se estructura en tres componentes interconectados: un generador de datos sintéticos que preserva la complejidad estructural real, una infraestructura de análisis optimizada para la experimentación y *notebooks* pedagógicos que guían el aprendizaje progresivo.

El entorno reproduce el esquema del ***Data Warehouse*** de **DATUZ**, modelando la información académica mediante entidades como estudiantes, asignaturas o profesorado. Esta estructura preserva las dependencias y jerarquías del sistema real, lo que permite ofrecer una experiencia de aprendizaje auténtica.

3.3.1. Generación de datos sintéticos

La solución a la cuestión de cómo garantizar la privacidad de datos sensibles manteniendo la autenticidad del modelo para el aprendizaje, combina el análisis automatizado de esquemas reales con la generación sintética.

Se han evaluado también otras opciones para la generación sintética de datos, como la herramienta de código abierto AUTO-DataGenCARS+ (https://webdiis.unizar.es/ silarri/prot/AUTO-DataGenCARSPlus/) [21], que tiene gran potencia sobre todo para la generación de datos sintéticos en entornos de sistemas de recomendación, pero se optó finalmente por Faker por ser suficiente para el propósito planteado.

Este enfoque preserva la integridad estructural y las relaciones del ***Data Warehouse*** real, pero eliminando cualquier información personal identificable. La estrategia se fundamenta en tres pilares: análisis semántico automatizado, generación coherente multi-tabla y validación de integridad referencial.

Análisis del esquema Académico de DATUZ

El punto de partida es el análisis del esquema parcial real del ***Data Warehouse*** **académico** de **DATUZ** (existen otros cinco esquemas). Este modela la actividad universitaria mediante tablas que abarcan desde la admisión y matrícula hasta indicadores de investigación y recursos humanos.

El análisis se basa en la ingeniería inversa del esquema de la base de datos para extraer metadatos estructurales, como tipos de datos, restricciones de integridad y claves. Esta información se complementa con un análisis semántico de las nomenclaturas para inferir el propósito funcional de cada atributo.

Estrategia de inferencia semántica

La limitada documentación del esquema de datos obligó a desarrollar un *script* de inferencia semántica automatizada. Este *script* analiza las nomenclaturas de los campos para determinar los tipos de datos sintéticos más apropiados. Por ejemplo, `fecha_nac` se interpreta como una fecha de nacimiento, `cod_centro` como un código de centro académico y `num_creditos` como un valor numérico de créditos **ECTS**. Este proceso permitió identificar relaciones implícitas entre entidades que no estaban documentadas formalmente, lo que mejoró la coherencia del modelo sintético resultante.

Diseño de generación coherente

La generación de datos sintéticos se fundamenta en tres principios arquitectónicos: la preservación de la integridad referencial, el mantenimiento de distribuciones estadísticas realistas y la garantía de anonimización completa. La integridad referencial se preserva mediante una generación secuencial que respeta las dependencias entre tablas. Las distribuciones realistas se logran modelando patrones

temporales (matrículas por semestre) y demográficos (distribución por edades) basados en estadísticas universitarias públicas y agregadas.

La anonimización se garantiza mediante la generación completamente aleatoria de identificadores personales, utilizando la biblioteca **Faker** con semillas controladas para asegurar la reproducibilidad. Ningún dato generado corresponde a información real, cumpliendo así con las regulaciones de privacidad. El sistema resultante produce datos sintéticos que replican fielmente la complejidad estructural de **DATUZ**, permitiendo experiencias de aprendizaje auténticas sin comprometer la confidencialidad. Los esquemas detallados de las tablas se presentan en el Anexo D.2.

3.3.2. Infraestructura del entorno

Los componentes tecnológicos que garantizan la portabilidad y simplicidad del entorno educativo incluyen una arquitectura containerizada que elimina las complejidades de configuración de los entornos **OLAP** tradicionales. Esta decisión responde a las barreras de entrada que la configuración técnica supone para los estudiantes. La infraestructura incluye:

- ***Notebooks* educativos:** seis **Jupyter *Notebooks*** que abarcan desde la introducción a **Atoti** hasta consultas analíticas y Saiku (véase el Anexo D.4).
- **Base de datos Oracle:** en contenedor **Docker** con **Oracle Database 21c** y *scripts* de creación de usuarios y tablas, y *scripts* de carga de datos.
- **Generador sintético:** el módulo `dm_academico_faker` que produce datos coherentes mediante la biblioteca **Faker**.
- ***Scripts* de automatización:** Herramientas para la descarga de controladores **JDBC** y la carga de datos.
- **Pentaho Server y Saiku:** el entorno de análisis OLAP con **Pentaho Server** y el complemento **Saiku** para consultas MDX.

Esta arquitectura modular facilita el despliegue en diferentes entornos educativos, desde laboratorios universitarios hasta instalaciones personales. La containerización garantiza la reproducibilidad y elimina problemas de configuración, permitiendo que los estudiantes se enfoquen en el aprendizaje de MDX.

Capítulo 4

Implementación

Este capítulo documenta la implementación de RAGDWAREuz y MDX LEARN, detallando los componentes técnicos clave. Se describe la estructuración de la base de conocimiento en Milvus, la construcción del *workflow* con LangGraph, el *frontend* en Chainlit y el desarrollo de los *notebooks* educativos. Se demuestra la viabilidad de integrar tecnologías de IA emergentes para crear herramientas de consulta y formación académica.

4.1. Desarrollo del sistema RAGDWAREuz

El desarrollo de RAGDWAREuz abarca desde la estructuración de la base de conocimiento hasta la implementación del *workflow* conversacional, integrando técnicas avanzadas de recuperación y generación.

4.1.1. Evolución de los *prompts*

El diseño de los *prompts* constituye un factor crítico. En RAGDWAREuz, se ha evolucionado desde versiones iniciales (véase el Anexo C.9) hasta versiones finales que incorporan técnicas avanzadas como *Chain of Thought* (CoT) [25].

La técnica CoT instruye al LLM para descomponer problemas complejos en una secuencia de pasos de razonamiento intermedios. El sistema implementa cinco etapas guiadas por CoT: identificación del tema, determinación de información relevante, extracción de datos, análisis de relaciones y formulación de la respuesta. Este proceso estructurado mejora la precisión y la consistencia, aunque implica un mayor tiempo de respuesta. Para más detalles sobre CoT, véase el Anexo C.8.

4.1.2. Base de conocimiento

La base de conocimiento del sistema está compuesta por la documentación oficial de los cubos de datos de SEGEDA, proporcionada por el personal de la Universidad. Estos documentos, en formato PDF y Markdown, describen las métricas, dimensiones y filtros de cada uno de los ocho ámbitos de datos. Un ejemplo de estos documentos puede consultarse en el Anexo C.5.

Durante la ingesta, cada documento se procesa para extraer su contenido textual y metadatos clave, como el ámbito temático (por ejemplo, “Grupo - Tipo” se refiere a categorías como “Académico” o “Recursos Humanos”). Una vez procesados todos los documentos, los fragmentos de texto resultantes conforman la base sobre la que opera el sistema.

La base de conocimiento alojada en Milvus se estructura mediante fragmentos (*chunks*) de tamaño variable. Esta aproximación permite una adaptación dinámica al contexto. Cada fragmento conserva metadatos esenciales: documento de origen, cubo SEGEDA y ámbito temático. Esto facilita una recuperación selectiva y precisa. Durante la carga, un LLM enriquece semánticamente cada fragmento generando resúmenes contextuales. Las plantillas de *prompt* utilizadas se detallan en el Anexo C.9. La búsqueda combina recuperación vectorial con algoritmos léxicos como BM25, optimizando así la cobertura y la relevancia de los resultados.

4.1.3. Arquitectura del *workflow* con LangGraph

La orquestación de la interacción entre los componentes se implementa con LangGraph, que crea *workflows* como máquinas de estados finitos. El sistema integra un *workflow* conversacional y otro de detección de ámbito. Estos coordinan las operaciones relativas a los ocho ámbitos de SEGEDA.

La arquitectura se estructura en seis componentes especializados: identificación de ámbito, reescritura de consultas, evaluación documental, generación de respuestas, procesamiento SQL y evaluación granular de la respuesta. Esta organización modular facilita el mantenimiento y la optimización. La Figura C.3 muestra el diagrama de secuencia del *workflow* principal completo de RAGDWAREuz. Por otro lado, la Figura C.2 muestra el diagrama de secuencia del uso básico de RAGDWAREuz.

Componente de identificación de ámbito

La correcta identificación del ámbito es el primer paso crítico. El *workflow* de este componente (Figura 4.1) implementa un mecanismo de tres fases: primero, analiza la consulta con palabras clave de los ámbitos de SEGEDA; si la confianza es baja, ejecuta una recuperación vectorial y analiza los metadatos; si ambos fallan, genera preguntas de clarificación.

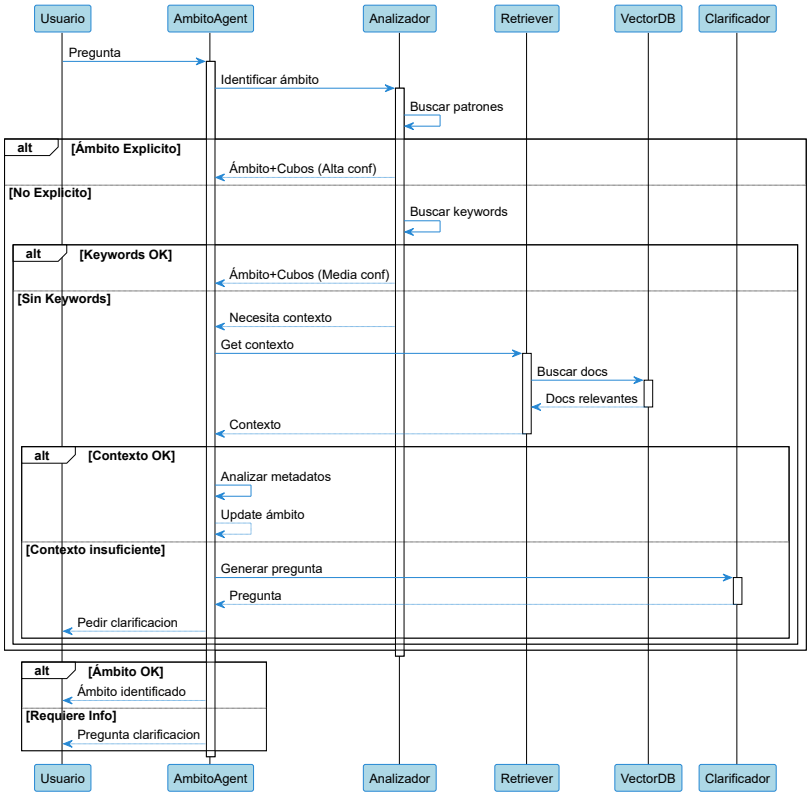

Figura 4.1: Diagrama de secuencia del componente de identificación de ámbito.

Reescritura adaptativa de consultas y evaluación de documentos

Para clarificar consultas ambiguas, el sistema implementa una reescritura condicional que se activa solo tras una clarificación exitosa. Este módulo transforma las consultas en formulaciones más precisas usando la terminología de SEGEDA, mejorando la efectividad de la recuperación. Los *prompts* para este proceso se detallan en el Anexo C.9.

La calidad de la respuesta depende de la relevancia de los documentos recuperados. Siguiendo el enfoque de CRAG (Subsección 2.2.3), un LLM especializado evalúa la relevancia de cada documento recuperado. Este *document grader* retorna una decisión binaria, y solo los documentos evaluados positivamente se incluyen en el contexto de generación. Las plantillas de evaluación se documentan en el Anexo C.9.

Generación de respuestas RAG y SQL

La fase final sintetiza la información recuperada en respuestas coherentes. La implementación utiliza técnicas RAG que garantizan la fidelidad a las fuentes documentales mientras producen respuestas naturales. Las plantillas de generación se presentan en el Anexo C.9.

Las consultas que requieren datos numéricos o agregaciones se traducen a SQL. El sistema incorpora un generador que transforma el lenguaje natural a sentencias SQL válidas para Oracle y un intérprete que explica los resultados a usuarios no técnicos. Las plantillas adaptadas se documentan en el Anexo C.9.

Evaluación granular y selección de estrategia

Utilizando un enfoque que combina la técnica *LLM as a Judge* (descrita en la Sección 2.3) con una extensión del concepto de *Mixture of Granularity* (MoG) presentado en la Sección 2.2.2. Este mecanismo analiza la respuesta generada a través de cinco métricas: fidelidad, precisión contextual, recuperación contextual, relevancia y un diagnóstico detallado. Las métricas se calculan automáticamente tras cada respuesta, lo que permite detectar degradaciones en el comportamiento del sistema y ajustar el tamaño de *chunk* según la consulta del usuario.

Esta funcionalidad introduce un coste adicional, ya que el *workflow* puede repetirse hasta dos veces para refinar la salida. Sin embargo, este proceso de autoevaluación permite obtener respuestas más precisas. Por ejemplo, mientras que un evaluador anterior solo analizaba métricas generales como la ausencia de alucinaciones o la relevancia, este enfoque granular ofrece una mejora sustancial. A continuación se muestra un ejemplo real del sistema para ilustrar la diferencia:

Pregunta: *¿Qué son los créditos matriculados y cómo se contabilizan?*

Respuesta sin evaluador granular:

> *Los créditos matriculados se refieren a los créditos que han sido oficialmente evaluados y han recibido una calificación, ya sea por presentación o no presentación del estudiante. Estos créditos son fundamentales para calcular métricas como la Tasa de Rendimiento y la Tasa de Evaluación dentro del ámbito académico del SEGEDA. Es importante destacar que los créditos obtenidos a través de programas de intercambio, movilidad, aquellos no incluidos en el plan oficial del estudiante o reconocidos no se contabilizan para estas métricas. Esto asegura que las evaluaciones reflejen con precisión el rendimiento académico dentro del marco institucional de la Universidad de Zaragoza.*

Respuesta con evaluador granular:

> *Los créditos matriculados son aquellos que los estudiantes han inscrito en sus planes de*

estudio y que no han sido cancelados, independientemente de si han sido calificados o no. Estos créditos se contabilizan dentro del ámbito académico del sistema SEGEDA, específicamente en el cubo MATRÍCULA. Es importante destacar que los créditos matriculados excluyen aquellos que son reconocidos, adaptados o convalidados a través de procesos de transferencia de créditos. Además, se deben considerar las exclusiones mencionadas para asegurar una contabilización precisa, especialmente en cursos académicos abiertos donde algunos créditos pueden estar pendientes de calificación. En el contexto del rendimiento académico, los créditos matriculados son fundamentales para evaluar la carga académica y el compromiso de los estudiantes dentro de un período determinado.

Esta evaluación multidimensional proporciona retroalimentación específica sobre cada aspecto de la generación. En la Figura 4.2 se puede observar el proceso de evaluación.

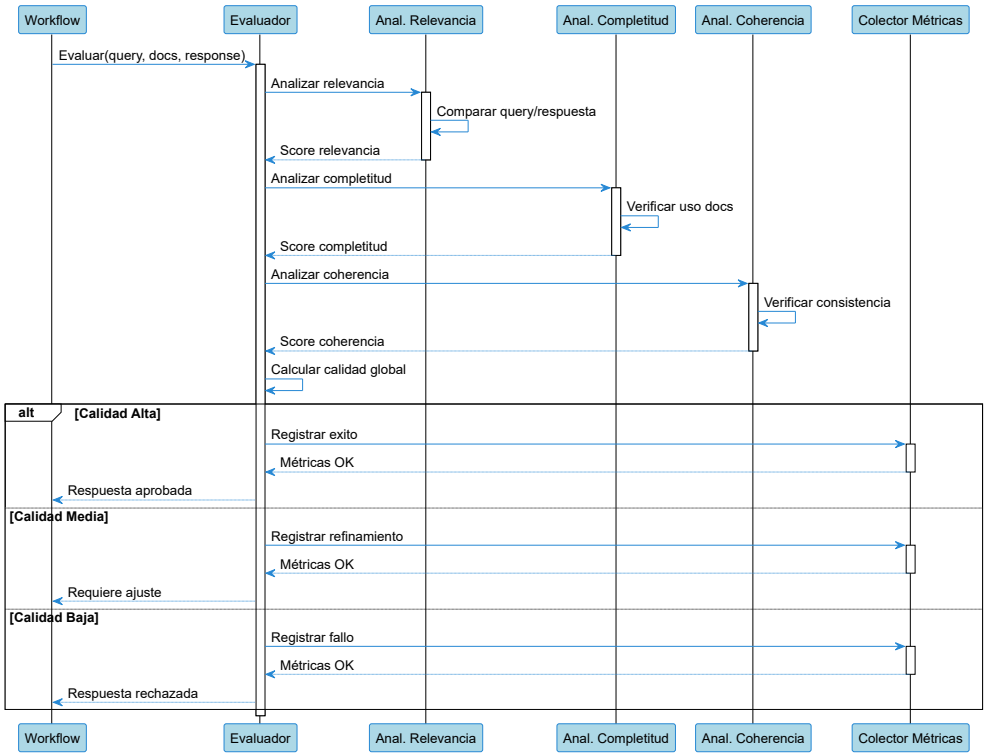

Figura 4.2: Diagrama de secuencia de la evaluación granular de documentos.

La selección de la granularidad óptima de los fragmentos de texto según la complejidad de la consulta se implementa mediante un mecanismo adaptativo que optimiza la estrategia de recuperación. Este proceso se ilustra en la Figura 4.3.

El sistema implementa una lógica inspirada en *Mix-of-Granularity* (MoG), que analiza patrones lingüísticos del dominio SEGEDA para determinar la estrategia de *chunking* más efectiva. El análisis clasifica las consultas en tres categorías (específicas, analíticas y amplias) para orientar la selección de la granularidad inicial.

La lógica adaptativa opera mediante retroalimentación continua. Cuando la métrica de *Context Recall* es insuficiente, el sistema tiende a incrementar la granularidad. Si fallan *Context Precision* o *Faithfulness*, tiende a reducirla. Para problemas de *Answer Relevance*, aplica estrategias diferenciadas. El sistema incorpora una memoria de intentos que previene bucles infinitos y evita la repetición de estrategias fallidas.

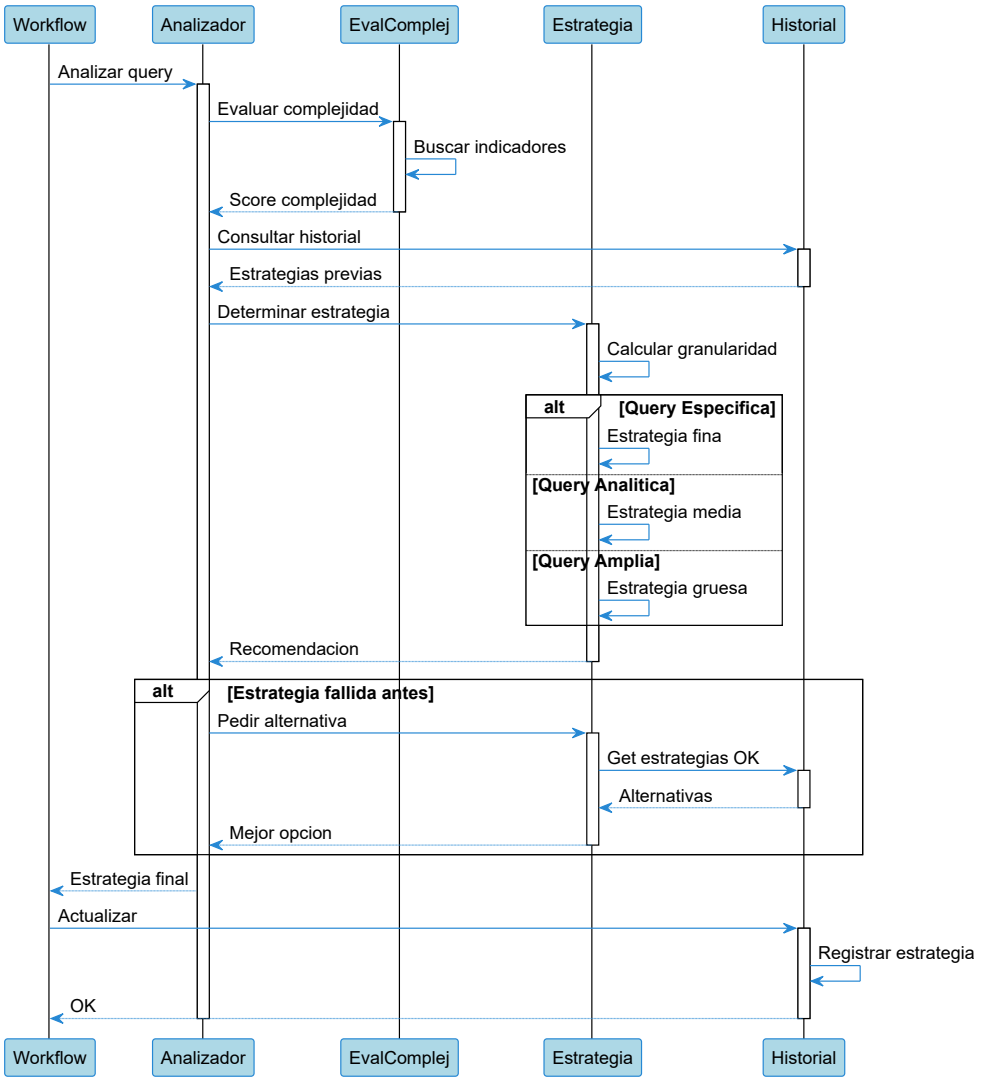

Figura 4.3: Diagrama de secuencia del análisis de la consulta para la recuperación adaptativa.

4.1.4. *Frontend*: Chainlit

La complejidad técnica del sistema RAG se traduce en una interfaz intuitiva mediante Chainlit, que ofrece una experiencia conversacional profesional. Chainlit proporciona transparencia al permitir visualizar los documentos que han contribuido a cada respuesta, lo que garantiza la credibilidad y la verificación de las fuentes. La gestión del contexto conversacional asegura la coherencia durante interacciones extendidas. Las capturas de pantalla del *frontend* se presentan en el Anexo C.11. Cuando el usuario accede por primera vez al sistema, se le presenta un mensaje de bienvenida que explica brevemente cómo interactuar con él y los ámbitos y cubos que se pueden consultar. En el Anexo B.6 se puede ver el mensaje de bienvenida que se muestra al usuario.

4.2. Desarrollo del entorno MDX LEARN

El entorno educativo para el aprendizaje de MDX se estructura en dos componentes: la generación de datos sintéticos que preservan la estructura del *Data Warehouse* de referencia y el desarrollo de *notebooks* educativos con una progresión didáctica.

4.2.1. Generación de datos sintéticos

Para crear un entorno educativo realista sin comprometer la privacidad, se generan datos sintéticos que reproducen la estructura y los patrones estadísticos del *Data Warehouse* de DATUZ. El módulo `dm_academico_faker`, que utiliza la biblioteca Faker para generar datos coherentes, preservando la integridad referencial y garantizando la anonimización. El sistema es compatible con Oracle Database y PostgreSQL mediante *scripts* automatizados. La documentación completa del proceso se presenta en el Anexo D.3.

4.2.2. *Notebooks* educativos

Para minimizar la curva de aprendizaje de MDX, se han desarrollado cinco Jupyter *Notebooks* con una progresión pedagógica desde conceptos fundamentales hasta técnicas avanzadas:

1. *01_introduccion_atoti.ipynb*: introduce conceptos básicos de análisis multidimensional y configuración de Atoti.
2. *02_consultas_mdx_basicas.ipynb*: desarrolla técnicas fundamentales de consulta MDX.
3. *03_cubos_multidimensionales.ipynb*: aborda operaciones de agregación, filtrado y construcción de consultas complejas.
4. *04_consultas_mdx_avanzadas.ipynb*: cubre técnicas especializadas como *drill-down* dinámico.
5. *05_consultas_sql_analiticas.ipynb*: integra consultas SQL analíticas con funciones de ventana.
6. *06_Saiku.ipynb*: explora Saiku como herramienta alternativa de análisis OLAP.

La integración nativa entre Jupyter y Atoti permite la visualización inmediata de resultados y la experimentación interactiva. Cada *notebook* incluye ejercicios graduales sobre datos sintéticos para consolidar los conceptos. Por ejemplo, un estudiante puede generar un conjunto de 10,000 estudiantes sintéticos, cargarlos en Oracle y después escribir consultas MDX para analizar la tasa de éxito por asignatura en el *notebook 02_consultas_mdx_basicas.ipynb*.

En etapas más avanzadas, puede utilizar el *notebook 04_consultas_mdx_avanzadas.ipynb* para dominar funciones complejas como *Year-to-Date* (YTD) y crear *rankings* de programas de movilidad más populares analizando flujos de estudiantes entrantes y salientes.

Por último, el *notebook 06_Saiku.ipynb* permite explorar Saiku como herramienta alternativa de análisis OLAP. Permite a los estudiantes experimentar con visualizaciones y consultas MDX similar a las que se hacen en SEGEDA. (Ver Anexo D.5 para ver fotos de ejemplos de análisis de datos). Aunque Saiku es un *plugin* obsoleto de Pentaho Server [1] la comunidad sigue manteniendo el proyecto (como la Ambiente Livre que cuenta con las versiones *legacy* de Pentaho Server [2] y un *fix* para el *plugin* de Saiku [3]). La documentación completa se presenta en el Anexo D.4.

4.3. Mantenibilidad y escalabilidad de RAGDWAREuz

La arquitectura modular de RAGDWAREuz facilita su mantenimiento y escalabilidad. La separación de componentes permite actualizaciones independientes. Por ejemplo, la configuración centralizada en un único archivo simplifica el despliegue y permite cambiar la estrategia adaptativa, el modelo de lenguaje o la base de conocimiento sin reescribir código.

[1] `https://www.meteorite.bi/`
[2] `https://github.com/ambiente-livre/pentaho-server`
[3] `https://github.com/ambiente-livre/saiku-fix`

La actualización de la base de conocimiento es incremental. El sistema detecta si un documento ha cambiado de versión (mediante el sufijo `_vX`) y actualiza únicamente ese documento, o añade los nuevos que no existían. En la Figura 4.4 se muestra el diagrama de secuencia de la actualización de la base de conocimiento.

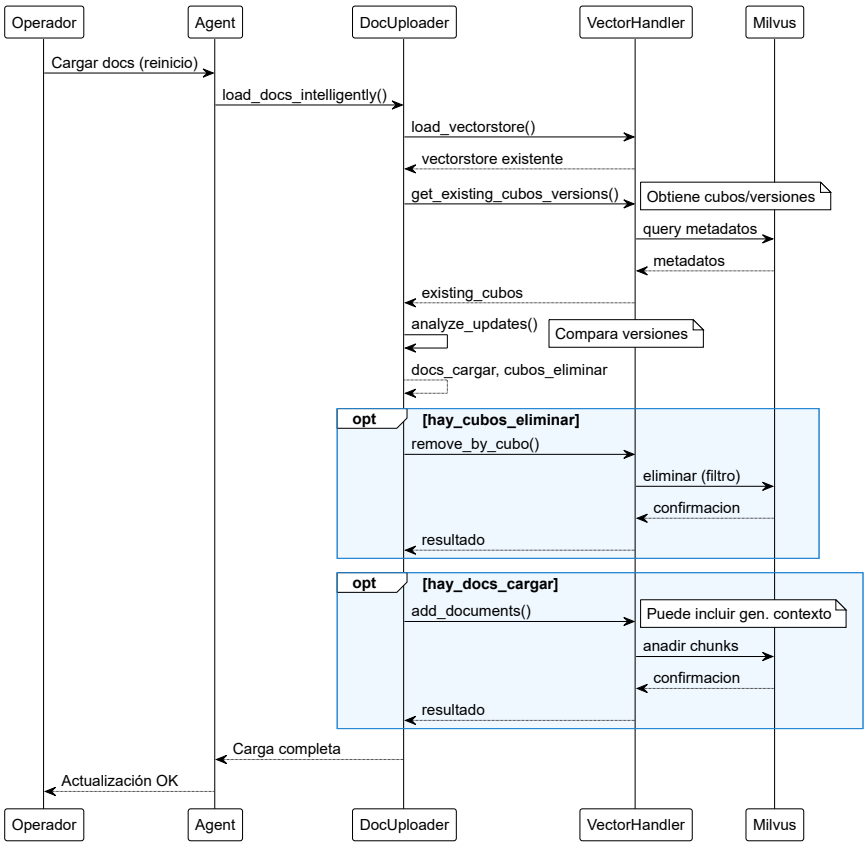

Figura 4.4: Diagrama de secuencia de la actualización de la base de conocimiento.

Además, el uso del patrón *Factory* en la implementación del *vectorstore* (con una clase base VectorStoreBase y una VectorStoreFactory) permite cambiar fácilmente de proveedor (de Milvus a Pinecone, por ejemplo) implementando una nueva clase y modificando la configuración.

También se ha implementado un recolector de métricas. La clase denominada MetricsCollector se encarga de recopilar métricas de rendimiento y calidad del sistema. Recoge métricas sobre las llamadas a LLM, el tiempo de ejecución de un nodo específico y del *workflow* completo. Estas métricas se almacenan en el directorio *metrics*, ordenadas por estrategia de *chunking* permitiendo un análisis del sistema sin dependencias externas.

Finalmente, la elección de LLMs es flexible. Solo es necesario añadir un nuevo fichero con los *prompts* y sus delimitadores específicos, ya que algunos modelos en Ollama no los especifican y deben buscarse en HuggingFace. (Ver Anexo B.1 y Anexo B.2 para más información sobre Ollama y HuggingFace).

Capítulo 5

Evaluación y validación de RAGDWAREuz

La evaluación de un sistema RAG es fundamental para determinar su efectividad y optimizar parámetros como la estrategia de *chunking*. En este capítulo se presenta la evaluación del sistema RAGDWAREuz, donde se examinan diferentes estrategias de segmentación y técnicas de *prompt engineering*.

Los LLM utilizados (llama3.2:3b, mistral-small-3.1:24b, qwen2.5:1.5b y BAII Multilingual-Large) se seleccionaron por su reducido tamaño, buscando una implementación eficiente en un entorno con recursos computacionales limitados (2 GPUs A10 con 46 GB de VRAM compartida). En el Anexo B.3 se detallan sus características técnicas específicas.

5.1. Datos de evaluación para DeepEval

Para la evaluación sistemática, se construyó un conjunto de datos de 90 preguntas sobre cinco ámbitos universitarios[1] [2], complementado con 20 preguntas frecuentes propuestas por expertos mediante la técnica HITL (*Human In The Loop*). Como LLM evaluador se ha utilizado Qwen2.5:32b.

La distribución de preguntas busca reflejar las prioridades institucionales para una evaluación equilibrada. El ámbito académico concentra la mayoría de las preguntas (41), seguido de I+D+i y Recursos Humanos (18 cada uno), Movilidad (11) y Estudios Propios (2). Esta estructura garantiza una cobertura integral de las actividades de la Universidad de Zaragoza y proporciona una base sólida para la evaluación del rendimiento del sistema.

5.2. Evaluación granular y el enfoque *LLM as a Judge*

Una de las características clave del sistema es la evaluación continua de la calidad de las respuestas, implementada mediante un enfoque conocido como *LLM as a Judge*. Tras cada respuesta generada, un LLM especializado la evalúa para detectar posibles degradaciones en el comportamiento del sistema. Esta funcionalidad, que puede desactivarse para reducir la latencia, implica un coste adicional, ya que el *workflow* puede repetirse hasta dos veces para refinar la respuesta. Sin embargo, este proceso de autoevaluación permite obtener respuestas más precisas y detalladas. Por ejemplo, ante la pregunta "¿Qué son los créditos matriculados y cómo se contabilizan?", una respuesta sin evaluación granular

[1] `https://github.com/carloss4dv/RAGDWAREuz/blob/main/preguntas_eval.json`

[2] `https://docs.google.com/spreadsheets/d/182SGJM5O5bK5zNL44m2Bg4oyxu3rIz_b/edit?usp=sharing&ouid=109629430432486032975&rtpof=true&sd=true`

podría ser superficial. En cambio, con el evaluador activado, el sistema ofrece una respuesta más completa, diferenciando entre créditos inscritos, cancelados, reconocidos y adaptados, y contextualizando la información en el cubo de datos correspondiente.

En esta sección se presentan los resultados de la evaluación del sistema **RAGDWAREuz**, analizando las distintas iteraciones del sistema con DeepEval y la técnica **HITL**, así como la evolución de los *prompts*.

5.2.1. Estrategia de *chunking* de tamaño fijo

Se evaluaron tres configuraciones de tamaño de fragmento fijo (256, 512 y 1024 *tokens*) para determinar su impacto en el rendimiento. Los resultados se presentan en la Tabla 5.1.

Chunk tamaño fijo	***Answer Relevancy***	***Faithfulness***	***Contextual Precision***	***Contextual Recall***	***Contextual Relevancy***
256	0.78	0.83	0.63	0.65	0.39
512	0.80	0.81	0.45	0.57	0.40
1024	0.79	0.79	0.70	0.71	0.36

Tabla 5.1: Métricas de evaluación según el tamaño de ***chunk*** fijo utilizado en **RAGDWAREuz**

Los resultados muestran que no hay un tamaño único óptimo. El tamaño de 512 *tokens* logró la mejor *Answer Relevancy* (0.80), aunque con diferencias marginales respecto a los otros tamaños. En *Faithfulness*, los *chunks* de 256 *tokens* obtuvieron el valor más alto (0.83), lo que sugiere que fragmentos más pequeños y específicos reducen el riesgo de que el LLM genere información desviada. Por el contrario, los *chunks* de 1024 *tokens* destacaron en las métricas de recuperación (*Contextual Precision* y *Contextual Recall* con 0.70 y 0.71), ya que un contexto más amplio facilita la localización de información. La *Contextual Relevancy* se mantuvo baja en todos los casos, indicando un área de mejora.

5.2.2. Evaluación con estrategia adaptativa

Para superar las limitaciones del enfoque fijo, se implementó una estrategia adaptativa que ajusta la granularidad de la información dinámicamente. Los resultados se muestran en la Tabla 5.2.

Estrategia inicial	***Answer Relevancy***	***Faithfulness***	***Contextual Precision***	***Contextual Recall***	***Contextual Relevancy***
256	0.77	0.84	0.64	0.71	0.40
512	0.81	0.81	0.63	0.82	0.39
1024	0.80	0.79	0.70	0.76	0.35

Tabla 5.2: Métricas de evaluación con estrategia de ***chunking*** adaptativo

La estrategia adaptativa mejoró significativamente el *Contextual Recall*, con incrementos de hasta el 44 % (para 512 *tokens*), y también la *Contextual Precision*. Sin embargo, la *Contextual Relevancy* se mantuvo baja, sugiriendo que la optimización del tamaño por sí sola no es suficiente. Estos resultados, al contrastarlos con estudios previos [24], subrayan que el rendimiento del *chunking* depende fuertemente del dominio y la configuración del sistema RAG.

5.2.3. Evaluación con segmentación semántica mediante NLTK

Como alternativa, se investigó la segmentación semántica con NLTK, que alinea los fragmentos con unidades lingüísticas naturales. Este enfoque se basa en la estructura semántica de los documentos de entrada que, como se detalló en el Capítulo 4, sigue la lógica del modelos de datos de SEGEDA.

En este modelo, una oración suele definir una medida o un atributo; un párrafo completo (delimitado por marcadores) describe una dimensión o medida en su totalidad; y una sección agrupa conjuntos de dimensiones o medidas. Se evaluaron varias configuraciones para determinar su impacto. (En el Anexo C.7 se presenta el script de análisis del corpus utilizado para la segmentación semántica.) La Tabla 5.3 y la Tabla 5.4 presentan los resultados de dos configuraciones distintas.

Estrategia NLTK (167, 307, 755)	***Answer Relevancy***	***Faithfulness***	***Contextual Precision***	***Contextual Recall***	***Contextual Relevancy***
167	0.75	0.81	0.59	0.55	0.44
307	0.76	0.81	0.68	0.68	0.46
755	0.79	0.85	0.65	0.61	0.49

Tabla 5.3: Métricas de evaluación con estrategia adaptativa y NLTK (tamaños 167, 307, 755)

En la primera configuración, el tamaño de 755 *tokens* destacó en *Answer Relevancy* (0.79) y *Faithfulness* (0.85). En la segunda, un tamaño de 646 *tokens* alcanzó un rendimiento notablemente alto en *Contextual Precision* (0.76) y *Contextual Recall* (0.75), superando a las configuraciones de tamaño fijo.

Estrategia NLTK (369, 646, 1094)	***Answer Relevancy***	***Faithfulness***	***Contextual Precision***	***Contextual Recall***	***Contextual Relevancy***
646	0.77	0.84	0.76	0.75	0.46

Tabla 5.4: Métricas de evaluación con estrategia adaptativa y NLTK (tamaño 646)

Estos hallazgos sugieren que la segmentación semántica con NLTK es una alternativa prometedora, ya que al respetar los límites de las oraciones puede mejorar la calidad del contexto recuperado.

5.3. Evaluación con expertos

Además de la evaluación cuantitativa, se realizó una evaluación cualitativa con un experto en el dominio para valorar la calidad y utilidad de las respuestas del sistema en un escenario de uso realista. La evaluación fue realizada por María Belén Gracia, jefa de proyectos de SICUZ (Servicio de Informática y Comunicaciones de la Universidad de Zaragoza), quien posee amplia experiencia en el manejo y consulta de los sistemas de información universitarios.

Es importante señalar que parte del conjunto de preguntas de evaluación se deriva de consultas y dudas reales planteadas a la unidad de Innovación de la Universidad de Zaragoza. Sin embargo, existe un sesgo inherente en este proceso: cuando la misma persona que formula las preguntas también las evalúa, puede interpretar implícitamente el contexto o la intención de preguntas mal planteadas o incompletas, una capacidad interpretativa que no necesariamente estará presente cuando otros usuarios interactúen con el sistema.

Tras la primera ronda de evaluación, se identificaron varias áreas de mejora en las respuestas generadas por el sistema. En la Tabla 5.5 se presentan las métricas de evaluación obtenidas en esta primera fase. Las trazas de ejecución de las rondas de evaluación están disponibles online[3] [4]. Las notas numéricas se calcularon con la media de la conversión de las notas de los expertos a escala de 0 a 1.

Los comentarios HITL pusieron de manifiesto problemas como la falta de precisión en algunas respuestas, la inclusión de información irrelevante o confusa, y la omisión de detalles clave o dimensiones relevantes para la consulta[5]. Además, se observó que en ocasiones las respuestas eran excesivamente genéricas, no se adaptaban al ámbito específico de la pregunta o no explicaban adecuadamente el procedimiento para obtener la información solicitada.

[3] `https://drive.google.com/file/d/1xlktDLw_iJhuIgpcYVs1DmXkPgX3Ryxs/view?usp=drive_link`

[4] `https://drive.google.com/file/d/1F7QZP5SXoth8veKC7PPfBCqH_FGuMbCk/view?usp=drive_link`

[5] `https://docs.google.com/spreadsheets/d/182SGJM5O5bK5zNL44m2Bg4oyxu3rIz_b/edit?usp=drive_link&ouid=109629430432486032975&rtpof=true&sd=true`

Tabla 5.5: Métricas de evaluación con expertos

Métrica	**Valor**
Contextual Relevancy	0.62
Contextual Recall	0.56
Faithfulness	0.66
Answer Relevancy	0.68
Answer Recall	0.59

Estos hallazgos evidenciaron la necesidad de ajustar los *prompts* para mejorar la claridad, la relevancia y la utilidad de las respuestas, así como para asegurar una mayor alineación con las expectativas de los expertos y las necesidades reales de los usuarios.

Como resultado de este proceso de revisión y ajuste de los *prompts*, se realizó una segunda evaluación cuyos resultados se presentan en la Tabla 5.6 (también disponibles online [6]). Esta comparación permite valorar el impacto de las mejoras introducidas y sirve como base para la discusión sobre la evolución de la calidad de las respuestas del sistema.

Tabla 5.6: Métricas de evaluación con expertos

Métrica	**Valor**
Contextual Relevancy	0.73
Contextual Recall	0.67
Faithfulness	0.76
Answer Relevancy	0.82
Answer Recall	0.74

Esta segunda evaluación reveló una mejora significativa en la calidad de las respuestas tras el ajuste de los *prompts*, especialmente en términos de relevancia, precisión y alineación con las necesidades de los usuarios. Sin embargo, aunque el sistema es capaz de generar respuestas coherentes y relevantes, aún persisten ciertos retos en la adaptación a consultas complejas o muy específicas. Estos resultados subrayan la importancia de un proceso iterativo de revisión y ajuste de los *prompts*, así como la necesidad de mantener una evaluación continua con expertos para seguir optimizando la utilidad y fiabilidad del sistema.

5.4. Conclusiones

La evaluación de **RAGDWAREuz** permite extraer varias conclusiones clave. En primer lugar, no existe un tamaño de *chunk* único que sea óptimo para todas las métricas: los fragmentos pequeños (256) maximizan la fidelidad, los grandes (1024) optimizan la recuperación y un tamaño intermedio (512) ofrece un buen equilibrio. Las estrategias de *chunking* adaptativo y la segmentación semántica con NLTK (con tamaños como 646 o 755 *tokens*) demuestran ser alternativas prometedoras que mejoran la recuperación y precisión.

La persistente limitación en *Contextual Relevancy* sugiere que el tamaño reducido del corpus y la especificidad del dominio de SEGEDA pueden estar afectando la efectividad del modelo de *embeddings* para capturar relaciones semánticas en este contexto académico específico. Finalmente, se confirma que el uso de técnicas avanzadas de *prompt engineering* como CoT resulta fundamental para mejorar la calidad de las respuestas, y que las mejores prácticas en sistemas RAG deben adaptarse cuidadosamente a las particularidades específicas de cada dominio de aplicación.

[6] `https://drive.google.com/file/d/1AlYu9CaYjyyck9UXhnkyex1YaJqlS9-w/view?usp=drive_link`

Capítulo 6

Conclusiones y trabajo futuro

Este capítulo presenta las conclusiones generales del trabajo y sintetiza los hallazgos clave tras la evaluación técnica detallada. Se analiza la relevancia de los resultados en el contexto de los sistemas inteligentes de consulta y aprendizaje académico. También se examinan las limitaciones encontradas durante el desarrollo y se proponen líneas de trabajo futuro para expandir y mejorar los resultados obtenidos.

6.1. Viabilidad de RAGDWAREuz

La evaluación de RAGDWAREuz demuestra la factibilidad de implantar un sistema RAG avanzado y flexible en un entorno universitario con recursos limitados. Basado en una arquitectura modular, equilibra el rendimiento y la eficiencia para garantizar su sostenibilidad.

Uno de los pilares fundamentales del diseño fue la optimización del uso de memoria. Como se detalló en el Capítulo 5, se optó por LLMs de alto rendimiento pero tamaño reducido, decisión que se refleja en los datos presentados en la Tabla 6.1.

Modelo/Proceso	**Consumo GPU (MiB)**	**Consumo GPU (GB)**
llama3.2:3bm	3000	3,15
mistral-small-3.1:24b	18994	19,92
qwen2.5:1.5b	1750	1,84
BAII MULTILINGUAL - LARGE	2444	2,56
Total	**26188**	**27,47**

Tabla 6.1: Consumo de memoria GPU de los modelos desplegados en **RAGDWAREuz**.

El consumo total de 27,5 GB de VRAM confirma que el sistema puede desplegarse en servidores universitarios estándar sin requerir *hardware* especializado, aspecto clave para garantizar su extensibilidad y adopción institucional. Además, no es estrictamente necesario utilizar GPU, ya que Ollama permite ejecutar los modelos en CPU, lo que reduce aún más los requisitos de *hardware* y facilita el despliegue en diversos entornos.

El tiempo de respuesta constituye otro factor crítico en la evaluación del sistema. Aunque los flujos de trabajo completos requieren entre 12 y 38 segundos, este tiempo está significativamente afectado por una latencia de red no optimizada durante la fase de evaluación. Esta latencia se debe a que la conexión entre el sistema y Milvus se realiza a través de un túnel SSH entre el campus de San Francisco y el campus de la EINA, lo que añade un retraso considerable al proceso. En un entorno de producción optimizado, donde ambas instancias estarían ubicadas en la misma red local, este tiempo se reduciría considerablemente a menos de 5 segundos, mejorando sustancialmente la experiencia del usuario.

Más allá de la eficiencia, la fortaleza del sistema es su adaptabilidad. La evaluación del Capítulo 5 (véase Tabla 5.2) demostró que una estrategia de *chunking* adaptativo supera a los enfoques de tamaño fijo.

Por último, todas las herramientas utilizadas son de código abierto, lo que permite que el sistema sea accesible y replicable en otros entornos académicos o de investigación. Esto es crucial para fomentar la colaboración y la innovación en el ámbito de la inteligencia artificial aplicada a la educación.

En términos de estabilidad para producción, el *stack* tecnológico elegido ofrece garantías diferenciadas: Milvus presenta alta estabilidad como base de datos vectorial madura en su versión 2.x, con amplia adopción empresarial y respaldo comercial de Zilliz. LangChain, aunque requiere gestión cuidadosa de versiones debido a su rápido desarrollo y cambios frecuentes en la *API*, ofrece estabilidad media suficiente si se fijan versiones específicas. Ollama proporciona una solución robusta para el despliegue de modelos de lenguaje con flexibilidad de *hardware*. Esta combinación asegura que **RAGDWAREuz** pueda desplegarse de manera confiable en entornos universitarios sin comprometer la funcionalidad.

Este resultado valida la filosofía de diseño del sistema: RAGDWAREuz es un flujo de trabajo flexible e inteligente, no una aplicación monolítica. Su capacidad para adaptar la recuperación a cada consulta garantiza la calidad de la respuesta y la evolución del sistema, demostrando ser una solución robusta, eficiente y adaptable para el dominio académico.

6.2. Cronograma del proyecto

El proyecto se desarrolló en cinco meses (febrero-junio 2025) siguiendo un enfoque iterativo. La Figura 6.1 muestra la planificación temporal.

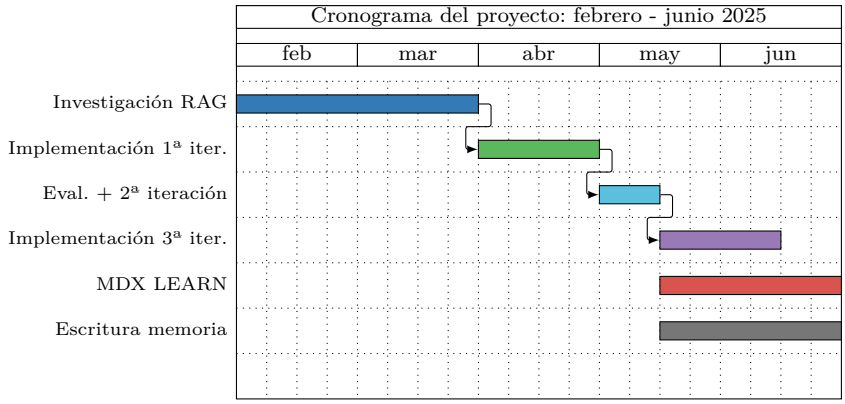

Figura 6.1: Cronograma del Trabajo de Fin de Grado

Las fases principales fueron:

- **Investigación (febrero-marzo):** estudio de arquitecturas RAG, LangChain/LangGraph, Milvus y fundamentos OLAP/MDX.
- **1ª iteración (abril):** prototipo básico de RAGDWAREuz con recuperación vectorial e interfaz Chainlit.
- **2ª iteración (2 semanas):** detección de ámbito, reescritura de consultas y evaluación documental CRAG.
- **3ª iteración (mayo):** sistema avanzado con evaluación granular, *chunking* adaptativo y *workflow* completo.

- **MDX LEARN (mayo-junio):** entorno educativo con datos sintéticos y *notebooks* educativos, desarrollado en paralelo.
- **Memoria (mayo-junio):** documentación en paralelo con el desarrollo final.

Esta planificación temporal demostró ser efectiva, permitiendo un desarrollo iterativo que facilitó la mejora continua del sistema y la incorporación de técnicas cada vez más sofisticadas. La superposición de la fase de documentación con el desarrollo final permitió una mejor integración de los resultados de evaluación en la memoria del proyecto.

6.3. Conclusiones principales

El desarrollo de RAGDWAREuz demuestra la viabilidad de aplicar sistemas RAG para la consulta de información institucional universitaria. Este sistema comprende consultas en lenguaje natural y recupera información relevante de documentos normativos, facilitando un acceso más intuitivo y eficiente para los usuarios autorizados.

MDX LEARN se presenta como una herramienta eficaz para la enseñanza de MDX, permitiendo a los usuarios comprender y practicar este lenguaje de consulta para el análisis multidimensional de datos.

La integración de ambos sistemas sugiere un potencial considerable para mejorar el acceso y la utilización de la información en las universidades, no solo para la toma de decisiones administrativas sino también para el apoyo a la investigación y el aprendizaje.

6.4. Limitaciones del trabajo

Una limitación clave del trabajo es el uso de datos sintéticos generados con Faker. Aunque estos datos son funcionales para las pruebas y validación del sistema, el empleo de datos reales anonimizados de la Universidad de Zaragoza ofrecería una validación más robusta y resultados extrapolables al entorno real de producción.

La arquitectura del sistema, aunque interactúa efectivamente con una base de datos, carece de un entorno OLAP completo y funcional. La implementación de una infraestructura OLAP integral mejoraría significativamente la capacidad de análisis multidimensional y la precisión de las respuestas generadas por el sistema.

La selección de modelos de *embedding* y LLMs constituyó un compromiso entre rendimiento computacional y disponibilidad de recursos. Las futuras investigaciones podrían explorar modelos más avanzados o específicamente adaptados al dominio académico para optimizar el rendimiento del sistema.

6.5. Líneas de trabajo futuro

Para superar las limitaciones identificadas y expandir las capacidades de los sistemas desarrollados, se proponen las siguientes líneas de trabajo futuro:

- Integración de datos reales: sustituir los datos sintéticos por datos reales y anonimizados de la Universidad de Zaragoza para una evaluación más precisa del sistema en un entorno operativo.
- Implementación de capa OLAP: integrar una capa OLAP completa para que el sistema realice análisis multidimensionales, permitiendo consultas más complejas.

- Enriquecimiento con ontologías: incorporar ontologías del dominio universitario para mejorar la comprensión del contexto y la precisión de la recuperación.
- Bases de conocimiento en grafo: investigar el uso de bases de conocimiento en grafo para representar relaciones complejas y mejorar la inferencia de conocimiento.
- Optimización de modelos: explorar LLMs y modelos de *embedding* alternativos, especialmente los adaptados al ámbito académico o al español, para optimizar la calidad y eficiencia.
- Expansión de MDX LEARN: ampliar MDX LEARN con más ejemplos, tutoriales interactivos y conexión a diversos cubos OLAP.

El trabajo realizado, enmarcado en una colaboración entre el proyecto de investigación NEAT-AMBIENCE (PID2020-113037RBI00) de la Universidad de Zaragoza y DATUZ (Datos Abiertos y Transparencia de la Universidad de Zaragoza), puede sentar las bases para futuras investigaciones y desarrollos del equipo de dicho proyecto, así como para futuras adaptaciones e implantaciones de la propuesta. Por otro lado, MDX LEARN puede resultar de gran utilidad en el marco de asignaturas del Grado en Ingeniería Informática como "Almacenes y Minería de Datos".

Índice de figuras

2.1. Proceso de creación y uso de *embeddings* vectoriales para búsqueda semántica 6
2.2. Proceso de la fase de recuperación en sistemas RAG 7
2.3. Proceso de la fase de generación en sistemas RAG 7

3.1. Diagrama de componentes de RAGDWAREuz. 18
3.2. Diagrama del secuencia de preparación de la base de conocimiento. 19
3.3. Ejemplo de contexto enriquecido generado por el LLM. 20
3.4. Diagrama del flujo de recuperación de información. 21
3.5. Diagrama del flujo de generación de respuesta. 21

4.1. Diagrama de secuencia del componente de identificación de ámbito. 26
4.2. Diagrama de secuencia de la evaluación granular de documentos. 28
4.3. Diagrama de secuencia del análisis de la consulta para la recuperación adaptativa. . . 29
4.4. Diagrama de secuencia de la actualización de la base de conocimiento. 31

6.1. Cronograma del Trabajo de Fin de Grado 38

C.1. Diagrama de componentes de RAGDWAREuz. 53
C.2. Diagrama de secuencia del uso basico de RAGDWAREuz. 54
C.3. Diagrama de secuencia del *workflow* principal completo. 56
C.4. Ejemplo de consulta y respuesta en RAGDWAREuz. 61
C.5. Mensaje de incio mostrado. 61

D.1. Ejemplo de un analisis de la matrícula de los estudiantes por sexo y tipo de estudio. . 63
D.2. Ejemplo de un analisis de la matrícula de los estudiantes por sexo. 64
D.3. Ejemplo de un analisis de los creditos totales por rama de estudios. 64
D.4. Ejemplo de un analisis de la evolución de los créditos de los estudiantes a lo largo de los cursos académicos. 65

D.5. Ejemplo de un analisis demografico de los estudiantes por edad y rama. 65

F.1. Flujo de trabajo básico de ***RAG clásico*** o ***naive RAG*** 69

F.2. Arquitectura de ***RAG modular*** con módulos independientes 69

F.3. Flujo de trabajo de ***RAG avanzado*** 70

F.4. Flujo de trabajo del ***Corrective Retrieval Augmented Generation*** (**CRAG**) . . . 70

F.5. Flujo de trabajo del ***Adaptive Retrieval Augmented Generation*** (**Adaptive-RAG**) 71

F.6. Flujo de trabajo del ***Self-Retrieval Augmented Generation*** (**Self-RAG**) 71

Índice de cuadros

3.1. Comparativa de motores de recuperación vectorial: Milvus, Chroma y Qdrant 16

3.2. Comparativa de *frameworks* RAG: LangChain, LlamaIndex y Haystack 17

5.1. Métricas de evaluación según el tamaño de ***chunk*** fijo utilizado en **RAGDWAREuz** 34

5.2. Métricas de evaluación con estrategia de ***chunking*** adaptativo 34

5.3. Métricas de evaluación con estrategia adaptativa y NLTK (tamaños 167, 307, 755) . . 35

5.4. Métricas de evaluación con estrategia adaptativa y NLTK (tamaño 646) 35

5.5. Métricas de evaluación con expertos 36

5.6. Métricas de evaluación con expertos 36

6.1. Consumo de memoria GPU de los modelos desplegados en **RAGDWAREuz**. 37

Bibliografía

[1] Akooda. *What is Retrieval Augmented Generation (RAG)?* Accedido: Mayo 29 de 2025. 2024. URL: https://www.akooda.co/blog/what-is-retrival-augmented-generation-rag.

[2] Omar Alonso y Ricardo Baeza-Yates. *Information Retrieval: Advanced Topics and Techniques*. 1st. Vol. 60. ACM Books. Accedido: Mayo 29 de 2025. New York, NY, USA: Association for Computing Machinery, 2024. ISBN: 9798400710506. DOI: 10.1145/3674127.

[3] Akari Asai et al. *Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection*. 2023. arXiv: 2310.11511 [cs.CL]. URL: https://arxiv.org/abs/2310.11511.

[4] Atoti. *What is Atoti?* Accedido: Mayo 29 de 2025. 2025. URL: https://www.atoti.io/what-is-atoti/.

[5] Confident AI. *DeepEval: The open-source LLM evaluation framework*. Accedido: Mayo 29 de 2025. 2025. URL: https://deepeval.com/.

[6] DataCamp. *What is OLAP?* Accedido: Mayo 29 de 2025. 2024. URL: https://www.datacamp.com/blog/what-is-olap.

[7] DeepLearning.AI. *Building and evaluating advanced RAG*. Accedido: Mayo 29 de 2025. 2025. URL: https://learn.deeplearning.ai/courses/building-evaluating-advanced-rag/.

[8] Google Cloud. *Retrieval Augmented Generation (RAG)*. Accedido: Mayo 29 de 2025. 2023. URL: https://cloud.google.com/use-cases/retrieval-augmented-generation.

[9] IBM. *Retrieval Augmented Generation (RAG) architectural pattern*. Accedido: Mayo 29 de 2025. 2023. URL: https://www.ibm.com/architectures/patterns/genai-rag.

[10] Nick R. Jennings y Michael Wooldridge. «Software Agents». En: *IEE Review* (1996). Accedido: 23 de junio de 2025. URL: https://www.cs.ox.ac.uk/people/michael.wooldridge/pubs/iee-review96.pdf.

[11] Christian S. Jensen, Torben B. Pedersen y Christian Thomsen. *Multidimensional Databases and Data Warehousing*. Accedido: Mayo 29 de 2025. Morgan & Claypool Publishers, 2010. DOI: 10.1007/978-3-031-01841-1.

[12] Soyeong Jeong et al. *Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity*. 2024. arXiv: 2403.14403 [cs.CL]. URL: https://arxiv.org/abs/2403.14403.

[13] Knowage. *What is an OLAP Cube?* Accedido: Mayo 29 de 2025. 2024. URL: https://www.knowage-suite.com/what-is-olap-cube/.

[14] Label Studio. *Why Human Review is Essential for Better RAG Systems*. Accedido: Mayo 29 de 2025. 2024. URL: https://labelstud.io/blog/why-human-review-is-essential-for-better-rag-systems/.

[15] LangChain. *LangChain: Build generative AI applications from prototype to production*. Accedido: Mayo 29 de 2025. 2025. URL: https://www.langchain.com/.

[16] LeewayHertz. *Naive RAG vs Advanced RAG vs Modular RAG: Which one to choose?* Accedido: Mayo 29 de 2025. 2025. URL: https://www.leewayhertz.com/advanced-rag/.

[17] LlamaIndex Team. *LlamaIndex - LlamaIndex*. https://docs.llamaindex.ai/. Accedido: Mayo 29 de 2025. 2024.

[18] Milvus Documentation Team. *Búsqueda Multi-Vector*. https://milvus.io/docs/es/multi-vector-search.md. Accedido: 23 de junio de 2025. 2024.

[19] Milvus Documentation Team. *Introducción a Milvus*. https://milvus.io/es/intro. Accedido: 23 de junio de 2025. 2024.

[20] Risbin RH. *Haystack: The Open Source Memory System for LLM Applications*. https://blog.adyog.com/2025/01/11/haystack-the-open-source-memory-system-for-llm-applications/. Accessed on June 17, 2025. Ene. de 2025.

[21] María del Carmen Rodríguez Hernández et al. «AUTO-DataGenCARS+: An Advanced User-Oriented Tool to Generate Data for the Evaluation of Recommender Systems». En: *22nd International Conference on Advances in Mobile Computing and Multimedia Intelligence (MoMM 2024)*. Vol. 15341. Lecture Notes in Computer Science. Accedido: Mayo 29 de 2025. Springer, dic. de 2024, págs. 176-191. ISBN: 978-3-031-78049-3. DOI: 10.1007/978-3-031-78049-3_16.

[22] Abhishek Tikku. *Rethinking RAG evaluations: Lessons from Human-in-the-Loop (HITL) and LLM-Judge practices*. Accedido: Mayo 29 de 2025. 2024. URL: https://medium.com/@1993Tikku/rethinking-rag-evaluations-lessons-from-human-in-the-loop-hitl-and-llm-judge-practices-1be1cb9da0f1.

[23] TruEra. *RAG Triad: Evaluating LLM applications for hallucination*. Accedido: Mayo 29 de 2025. 2025. URL: https://www.trulens.org/getting_started/core_concepts/rag_triad/.

[24] Xiaohua Wang et al. *Searching for Best Practices in Retrieval-Augmented Generation*. 2024. arXiv: 2407.01219 [cs.CL]. URL: https://arxiv.org/abs/2407.01219.

[25] Jason Wei et al. «Chain-of-Thought Prompting Elicits Reasoning in Large Language Models». En: *Advances in Neural Information Processing Systems*. Vol. 35. Accedido: Mayo 29 de 2025. Curran Associates, Inc., 2022, págs. 24824-24837. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

[26] Shi-Qi Yan et al. *Corrective Retrieval Augmented Generation*. 2024. arXiv: 2401.15884 [cs.CL]. URL: https://arxiv.org/abs/2401.15884.

[27] Wayne Xin Zhao et al. *A Survey of Large Language Models*. Accedido: Mayo 29 de 2025. 2023. URL: https://arxiv.org/abs/2303.18223.

[28] Zhengyu Zhou, Zhangyin Chung y Aixin Sun. «Mix of Granularity: Towards Balanced Text Granularity for Retrieval-Augmented Generation». En: *arXiv* (2024). Accedido: Mayo 29 de 2025. URL: https://arxiv.org/abs/2406.00456.

Anexos

Anexo A

Glosario de términos

Este glosario unifica la terminología técnica utilizada a lo largo del documento, proporcionando definiciones claras de conceptos clave en el contexto del proyecto.

Términos generales

Agent Componente software de LangChain que permite a los LLMs acceder y utilizar herramientas externas de forma autónoma para resolver tareas complejas.

Answer Recall Métrica que evalúa si la respuesta generada aborda todos los aspectos necesarios de la pregunta.

Answer Relevancy Métrica que evalúa si la respuesta generada es directamente pertinente a la pregunta del usuario.

Chunk Fragmento de texto de tamaño definido en el que se dividen documentos grandes para facilitar su procesamiento por los LLMs.

Chunking Proceso de división de documentos extensos en fragmentos más pequeños y manejables.

Context Recall Métrica que mide la completitud de los hechos necesarios para responder una pregunta.

Contextual Precision Métrica que evalúa la relevancia de los fragmentos recuperados para la pregunta.

Contextual Relevancy Métrica que mide la relevancia del contexto recuperado en relación con la consulta.

Data Warehouse Almacén de datos que centraliza información de múltiples fuentes para facilitar el análisis y la toma de decisiones.

Document Grader Componente especializado que evalúa la relevancia de documentos recuperados antes de incluirlos en el contexto de generación.

Drill-down Operación OLAP que permite descender a niveles de mayor detalle en una jerarquía dimensional.

Embeddings Representaciones vectoriales numéricas de texto que capturan relaciones semánticas en un espacio de alta dimensión.

Faithfulness Métrica que determina si la respuesta generada es consistente con la información del contexto recuperado.

Frontend Interfaz de usuario que permite la interacción con el sistema mediante una interfaz gráfica o conversacional.

Framework Conjunto de herramientas, bibliotecas y convenciones que proporcionan una base para el desarrollo de aplicaciones.

***Human in the Loop* (HITL)** Técnica de evaluación que integra evaluadores humanos en el proceso de revisión de respuestas del sistema.

LLM *Large Language Model*. Modelo de aprendizaje automático de gran tamaño especializado en procesamiento de lenguaje natural.

LLM as Judge Enfoque que utiliza un modelo de lenguaje para evaluar automáticamente la calidad de respuestas según criterios predefinidos.

Notebook Documento computacional interactivo que combina código ejecutable, visualizaciones y texto explicativo.

OLAP *Online Analytical Processing*. Tecnología que facilita el análisis multidimensional de datos empresariales.

Pipeline Secuencia de procesos automatizados que transforma datos de entrada en resultados específicos.

Prompt Instrucción o consulta en lenguaje natural proporcionada a un modelo de lenguaje para obtener una respuesta específica.

Query Rewriting Técnica que optimiza consultas del usuario reformulándolas para mejorar la recuperación de información.

RAG *Retrieval Augmented Generation*. Paradigma que combina recuperación de información con generación de texto para producir respuestas precisas.

Roll-up Operación OLAP que permite ascender a niveles de mayor agregación en una jerarquía dimensional.

Workflow Flujo de trabajo estructurado que define la secuencia de tareas y decisiones en un proceso automatizado.

Términos específicos del proyecto

DATUZ Datos Abiertos y Transparencia de la Universidad de Zaragoza. Iniciativa institucional para democratizar el acceso a información universitaria.

MDX *Multidimensional Expressions*. Lenguaje estándar para consultar y manipular datos en cubos OLAP.

MDX LEARN Entorno educativo autocontenido desarrollado para facilitar el aprendizaje práctico de MDX y conceptos OLAP.

RAGDWAREuz Asistente conversacional basado en arquitectura RAG desarrollado para facilitar el acceso a información académica de la Universidad de Zaragoza.

SEGEDA Servicio de Gestión de Datos. Sistema de inteligencia de negocio de la Universidad de Zaragoza basado en Pentaho Server.

Herramientas y tecnologías

Atoti *Framework* de código abierto para análisis OLAP en memoria que permite construir y consultar cubos multidimensionales en tiempo real.

Chainlit Herramienta de código abierto para crear interfaces de usuario conversacionales para aplicaciones de inteligencia artificial.

DeepEval *Framework* especializado en evaluación de sistemas RAG que proporciona métricas automáticas de calidad.

Faker Biblioteca de Python para generar datos sintéticos realistas que preservan la estructura de datos reales.

LangChain Biblioteca de código abierto que facilita la creación de aplicaciones con LLMs mediante componentes modulares.

LangGraph Extensión de LangChain que permite crear flujos de trabajo complejos mediante grafos de estados finitos.

Milvus Base de datos vectorial de código abierto optimizada para búsquedas de similitud semántica a gran escala.

Ollama Herramienta de código abierto para el despliegue local de modelos de lenguaje grandes.

Saiku Herramienta de análisis OLAP que proporciona una interfaz web para la construcción de consultas MDX.

Anexo B

Herramientas y Tecnologías

B.1. Ollama

Ollama es una herramienta de código abierto, está diseñada para simplificar la integración de LLMs directamente en aplicaciones y sistemas locales. Está dirigida principalmente a desarrolladores y organizaciones que priorizan un mayor control y privacidad en sus aplicaciones de IA. La característica fundamental de Ollama es su capacidad de desplegar LLMs localmente, asegurando que todo el procesamiento y almacenamiento de datos ocurran dentro del entorno de las organizaciones, sin depender de servicios en la nube externos. Operacionalmente, Ollama carga el LLM elegido en memoria (RAM) del servidor y utiliza la unidad de procesamiento gráfico (GPU) para acelerar la inferencia. Además, expone una *API* REST, lo que permite integrarse fácilmente en los flujos de trabajo.

B.2. HuggingFace

Empresa y comunidad dedicada a avanzar y democratizar la IA a través del código abierto. Es conocida por su biblioteca Transformers, ampliamente utilizada para procesamiento del lenguaje natural. Cuenta con su propia plataforma que permite a los usuarios compartir y utilizar modelos de aprendizaje automático.

Ofrece herramientas para crear, descubrir y colaborar en proyectos de aprendizaje automático. Permite alojar modelos, conjuntos de datos y aplicaciones ilimitados de forma pública. HuggingFace juega un papel crucial al hacer que modelos como LLama3.2:3b, Mistral Small 24B y Qwen2.5 estén disponibles para la comunidad, fomentando la innovación y colaboración.

B.3. LLMs utilizados

B.3.1. Llama3.2

Modelo de lenguaje desarrollado por Meta AI con 3 mil millones de parámetros. Destaca por su tamaño relativamente pequeño. Supera a modelos más grandes en tareas como el seguimiento de instrucciones. Utiliza una arquitectura de transformador optimizada, con un contexto de 128k *tokens* (en el sistema solo se utilizan 8k *tokens*). Por otro lado, soporta idiomas como inglés, alemán, francés, italiano, portugués, español, hindi, chino, japonés y coreano. Su rendimiento en *benchmarks* como IFEval (77.4 para 3B bf16) indica que puede seguir instrucciones complejas. Aunque posiblemente no sea "muy bueno" como evaluación de generación, se ha optado por él debido a su tamaño y capacidad de respuesta. En la tabla se muestran más medidas y métricas sobre el rendimiento de LLama, extraídas

de HuggingFace [1].

B.3.2. Mistral 3.1 Small 24B

Mistral Small 24B, creado por Mistral AI, es un modelo eficiente que compite con modelos más grandes. Ofrece respuestas rápidas y capacidades avanzadas de razonamiento. Soporta decenas de idiomas, incluyendo inglés, español, francés, alemán, italiano, portugués, ruso, árabe, chino y japonés. Ofrece capacidades de conversación y razonamiento de vanguardia, ideales para tareas complejas. Puede ejecutarse localmente en un MacBook 32 GB en RAM estando cuantizado. Está bajo la licencia Apache 2.0, lo que permite su uso comercial y no comercial. Es ideal para agentes conversacionales de respuesta rápida, llamadas de funciones de baja latencia, creación de expertos en temas específicos mediante ajuste fino, e inferencia local a datos sensibles. Su eficiencia lo posiciona como una excelente opción para aplicaciones que requieren velocidad y precisión. [2].

B.3.3. Qwen2.5 1.5B

Qwen2.5 1.5B es parte de la familia Qwen2.5 desarrollada por el equipo Qwen de Alibaba Cloud. Este modelo es particularmente notable por su aplicación en sistemas RAG. Destaca por su capacidad de ajustarse a tareas que involucran documentos complejos, como manuales técnicos o materiales educativos.[3]

B.4. Componentes de Haystack

Los **componentes clave** de Haystack incluyen:

- **Tuberías (Pipelines):** definen la secuencia de operaciones para procesar y generar información. Estas tuberías pueden ser predefinidas para tareas comunes (como sistemas de pregunta-respuesta) o completamente personalizadas mediante el encadenamiento de diversos componentes, ofreciendo una gran flexibilidad en el diseño del flujo de trabajo.
- **Componentes (Components):** son los bloques de construcción fundamentales de Haystack. Cada componente realiza una tarea específica dentro de la tubería, como la recuperación de documentos, la generación de texto o el almacenamiento de datos. Haystack proporciona una rica biblioteca de componentes listos para usar, que abarcan una amplia gama de funcionalidades:
  - ***Fetchers:*** recuperan datos de diversas fuentes, como páginas web o archivos locales.
  - ***Converters:*** transforman datos brutos (por ejemplo, PDF, TXT) en formatos que Haystack puede procesar.
  - ***Preprocessors:*** realizan tareas de limpieza y segmentación del texto (*chunking*) para optimizar el rendimiento de los modelos de lenguaje.
  - ***Embedders:*** generan representaciones vectoriales (*embeddings*) semánticas del texto, esenciales para la búsqueda por similitud.
  - ***Retrievers:*** recuperan los fragmentos de información más relevantes de una base de conocimiento en respuesta a las consultas de los usuarios.
  - ***Generators:*** utilizan LLMs para generar respuestas en lenguaje natural basadas en la información recuperada.
  - ***Writers:*** almacenan datos procesados (como *embeddings* o documentos) en bases de datos vectoriales u otros sistemas de almacenamiento para su uso posterior.

[1] `https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct`
[2] `https://mistral.ai/news/mistral-small-3-1`
[3] `https://huggingface.co/Qwen/Qwen2.5-1.5B`

B.5. Ecosistema LangChain y LangSmith

LangChain es un *framework* de desarrollo de aplicaciones basadas en LLMs. Proporciona herramientas y abstracciones para construir cadenas (*chains*) y agentes complejos. Langsmith, por su parte, es una plataforma para la depuración, el seguimiento y la evaluación de aplicaciones construidas con Langchain. En **RAGDWAREuz**, Langchain fue fundamental para orquestar el flujo RAG, incluyendo la recuperación de documentos, la interacción con el LLM y el manejo del historial de conversación. Langsmith se utilizó durante el desarrollo para trazar la ejecución de las cadenas, analizar los *prompts* y las respuestas intermedias, y depurar el comportamiento del agente. LangSmith tiene un límite de uso gratuito, por lo que puede elegirse si utilizarse o no. Para utilizarlo solo hay que configurar la variable de entorno `LANGSMITH_TRACING` a `true` y la variable de entorno `LANGSMITH_API_KEY` con la clave de *API* de LangSmith.

B.6. Interfaz de Usuario con Chainlit

Chainlit es una herramienta de código abierto para crear rápidamente interfaces de usuario para aplicaciones de IA basadas en Python. Se utilizó para desarrollar la interfaz de chat de **RAGDWAREuz**, permitiendo una interacción intuitiva con el sistema. Chainlit facilita la visualización de los pasos de pensamiento del agente (si se exponen) y la presentación de las respuestas y fuentes de información. Capturas de pantalla de la interfaz se incluyen en el Anexo C.11.

El mensaje de bienvenida del sistema **RAGDWAREuz** se configura en el archivo *chainlit_app.py* del directorio *frontend* del proyecto. Este mensaje se muestra al usuario al iniciar la aplicación y proporciona una introducción al sistema, destacando su propósito y funcionalidades principales. El mensaje de bienvenida es una parte importante de la experiencia del usuario, ya que establece el tono para la interacción con el asistente conversacional. [4]

[4] `https://github.com/carloss4dv/RAGDWAREuz/blob/main/frontend/chainlit_app.py`

Anexo C

Implementación de RAGDWAREuz

C.1. Diagramas de componentes y secuencia de RAGDWAREuz

En la Figura C.1 se muestra el diagrama de componentes de **RAGDWAREuz**. En la Figura C.2 se muestra el diagrama de secuencia del *workflow* principal completo de **RAGDWAREuz**.

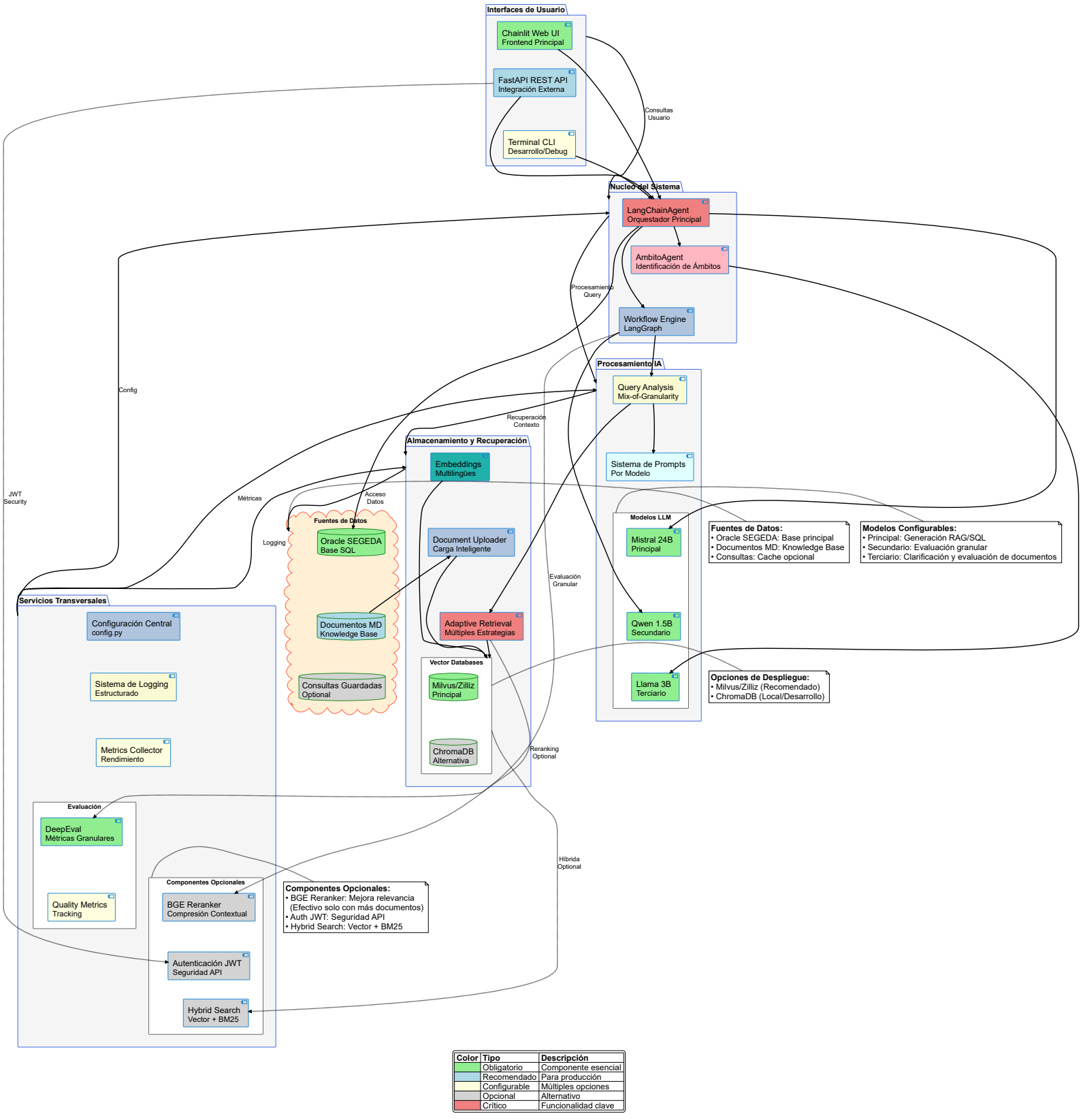

Figura C.1: Diagrama de componentes de RAGDWAREuz.

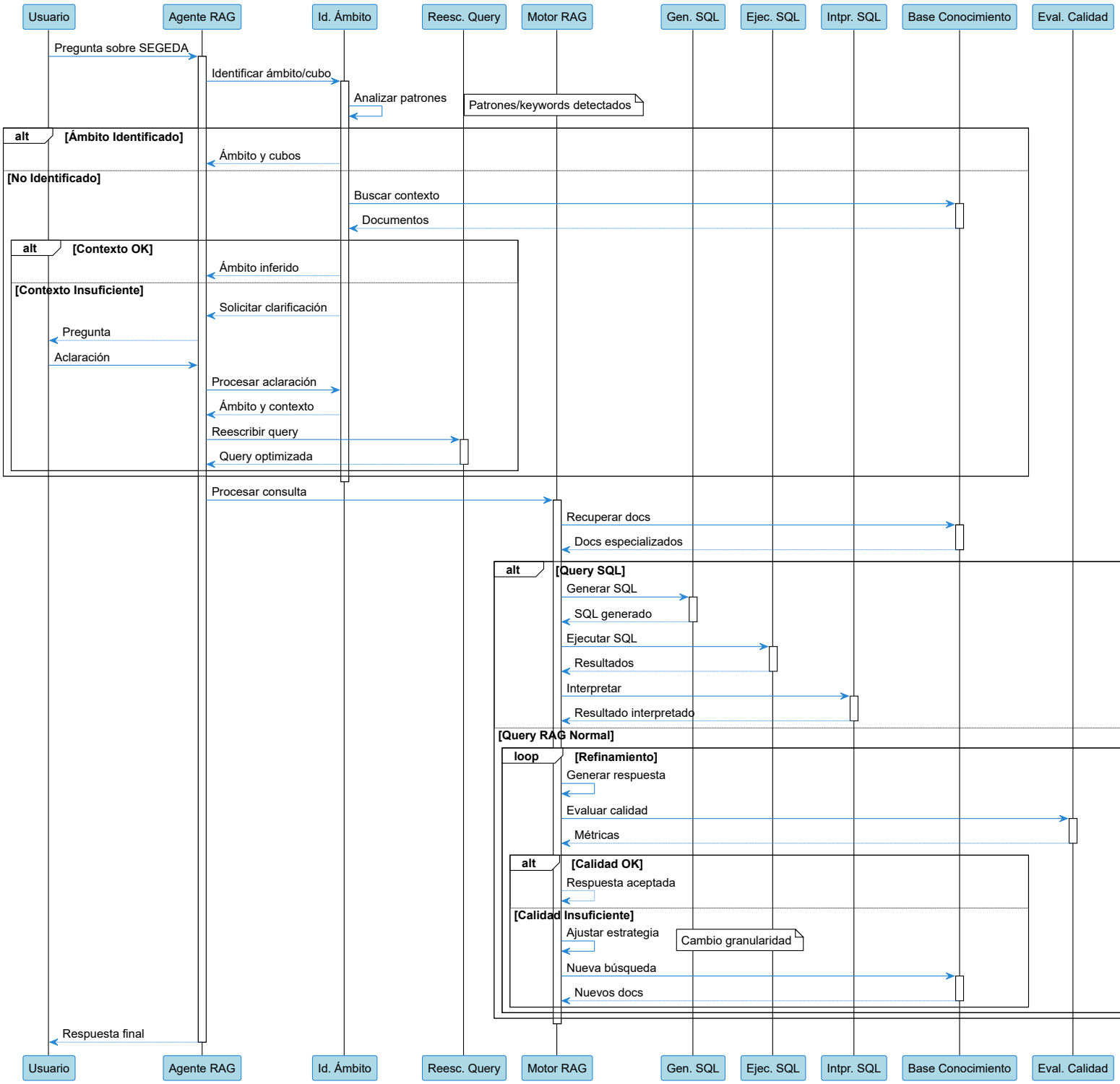

Figura C.2: Diagrama de secuencia del uso basico de RAGDWAREuz.

C.2. Diagrama de secuencia del *workflow* principal completo

En la Figura C.3 se muestra el diagrama de secuencia del *workflow* principal completo de **RAGDWAREuz**.

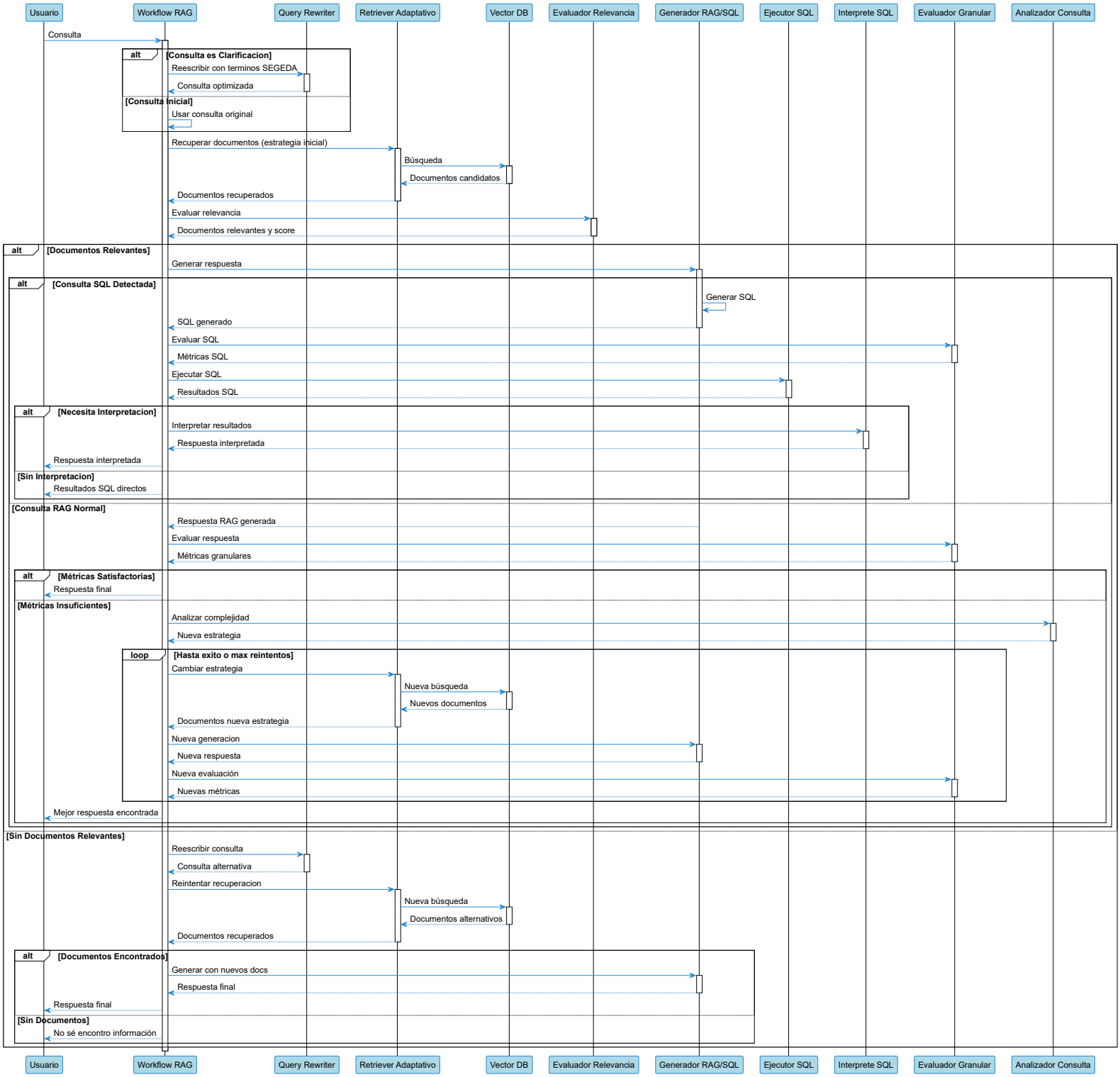

Figura C.3: Diagrama de secuencia del *workflow* principal completo.

C.3. Repositorio del Proyecto

El proyecto **RAGDWAREuz** está disponible en GitHub.[1]. Este repositorio contiene todo el código fuente, documentación y recursos necesarios para replicar y entender el sistema de Recuperación Aumentada por Generación de Respuestas (RAG) desarrollado para la Universidad de Zaragoza.

C.4. Parser de documentos DATUZ

El parser desarrollado para transformar los documentos tabulares de **DATUZ** a formato Markdown estructurado está disponible en el repositorio del proyecto.[2]. Este script es fundamental para la preparación de la base de conocimiento de **RAGDWAREuz**, permitiendo una correcta segmentación y posterior recuperación de la información.

C.5. Documentos procesados de DATUZ en formato Markdown

Los documentos procesados de **DATUZ** en formato Markdown, que sirven como base para la generación de *chunks* y su posterior indexación en la base de datos vectorial, están disponibles en el repositorio del proyecto.[3]. Estos documentos son el resultado del proceso de conversión realizado por el parser mencionado en el Anexo C.4.

C.5.1. Documento de ejemplo

Los documentos sobre SEGEDA contienen información sobre medidas y dimensiones de un cubo de datos. Un ejemplo sería el cubo de matrícula que contiene medidas como el numero de estudiantes matriculados, creditos matriculados y dimensiones como asignatura, centro, etc. [4]

C.6. Carga de documentos en Milvus

El código del *document uploader* utilizado para cargar los documentos de **DATUZ** en la base de datos vectorial Milvus está disponible en el repositorio del proyecto.[5].Esta clase es esencial para la actualización del *corpus* de documentos.

C.7. Análisis del corpus con NLTK

El script utilizado para analizar el corpus de documentos de **DATUZ** con la librería NLTK está disponible en el repositorio del proyecto **RADWAREuz**.[6]. Este script permite realizar un análisis detallado de los documentos, incluyendo la tokenización, el conteo de palabras y la identificación de entidades nombradas, lo que facilita la comprensión y mejora del corpus para su uso en el sistema RAG.

[1] https://github.com/carloss4dv/RAGDWAREuz
[2] https://github.com/carloss4dv/datuz_parser
[3] https://github.com/carloss4dv/RAGDWAREuz/tree/main/output_md
[4] https://github.com/carloss4dv/RAGDWAREuz/blob/main/output_md/info_cubo_matricula_v24.md
[5] https://github.com/carloss4dv/RAGDWAREuz/blob/main/vectorstore/document_uploader.py
[6] https://github.com/carloss4dv/RAGDWAREuz/blob/main/chunk_analyzer_v2.py

C.8. Chain of Thought (CoT) en RAGDWAREuz

Chain of Thought (CoT) es una técnica de *prompting* que instruye a los LLMs para que generen una secuencia de pasos de razonamiento intermedios antes de llegar a una respuesta final. Este anexo describe cómo se implementó CoT en **RAGDWAREuz**. Los detalles específicos de los prompts que implementan CoT se pueden encontrar en los archivos de prompts referenciados en el Anexo C.9.

C.8.1. Fundamentos del razonamiento paso a paso

El enfoque tradicional de *prompting* solicita directamente una respuesta al modelo. En contraste, CoT descompone el problema en etapas intermedias, emulando el proceso de pensamiento humano. Esta descomposición permite al modelo abordar cada subtarea de manera secuencial, lo que puede reducir errores y mejorar la coherencia del resultado final.

La técnica se basa en dos principios fundamentales:

- **Descomposición de problemas:** fragmentación de consultas complejas en pasos manejables.
- **Transparencia del razonamiento:** exposición del proceso de inferencia para facilitar la verificación y depuración.

C.8.2. Implementación en RAGDWAREuz

En el contexto de **RAGDWAREuz**, CoT se implementa mediante *prompts* estructurados que guían al modelo a través de cinco etapas de razonamiento:

1. **Identificación del tema:**
   Determinación del ámbito temático de la consulta (académico, I+D+i, RRHH, movilidad, estudios propios, etc.).
2. **Análisis de información relevante:**
   Evaluación del contexto recuperado para identificar datos pertinentes a la consulta.
3. **Extracción de datos:**
   Selección de la información específica dentro del contexto que responde directamente a la consulta.
4. **Análisis de relaciones:**
   Identificación de conexiones o relaciones entre diferentes elementos de datos si es necesario.
5. **Síntesis de respuesta:**
   Formulación de una respuesta coherente basada en el análisis previo y los datos extraídos.

C.8.3. Estructura de prompts CoT

Los prompts **CoT** implementados en el sistema siguen una estructura específica que combina instrucciones de razonamiento con ejemplos de demostración:

```
CHAIN OF THOUGHT EVALUATION PROCESS:
Step 1: Parse the original question to identify information requirements and
    SEGEDA scope
Step 2: Analyze retrieved documents for relevance, completeness, and accuracy
Step 3: Examine the generated response for factual consistency with documents
Step 4: Assess response completeness and directness in addressing the
    question
```

```
Step 5: Calculate numerical scores for each dimension based on systematic
    analysis
Step 6: Provide brief diagnostic reasoning for each metric

EXAMPLES:
Example 1 - High Quality Response:
Question: "¿Cuántos profesores titulares hay en el ámbito académico?"
Documents: [PDI cube with professor data]
Response: {{"faithfulness": 0.9, "context_precision": 0.9, "context_recall":
    0.8, "answer_relevance": 0.9, "diagnosis": {{"faithfulness_reason": "Data
    directly from PDI cube", "context_precision_reason": "PDI cube directly
    relevant", "context_recall_reason": "Complete professor data retrieved", "
    answer_relevance_reason": "Direct answer to question"}}}}

Example 2 - Low Quality Response:
Question: "¿Cuál es la tasa de graduación en ingeniería?"
Documents: [MATRÍCULA cube only]
Response: "Hay muchos estudiantes matriculados en ingeniería"
Output: {{"faithfulness": 0.6, "context_precision": 0.4, "context_recall":
    0.3, "answer_relevance": 0.2, "diagnosis": {{"faithfulness_reason": "Vague
     claim without specific data", "context_precision_reason": "MATRÍCULA cube
     lacks graduation rates", "context_recall_reason": "Missing EGRESADOS cube
     data", "answer_relevance_reason": "Doesn't answer graduation rate
    question"}}}}

Consulta: [CONSULTA_USUARIO]
Contexto: [DOCUMENTOS_RECUPERADOS]
Respuesta: [RESPUESTA_GENERADA]
```

C.8.4. Beneficios observados

La implementación de CoT en **RAGDWAREuz** produjo mejoras medibles en múltiples aspectos:

- **Precisión aumentada:** Reducción de respuestas incorrectas o irrelevantes.
- **Consistencia mejorada:** Mayor uniformidad en el formato y calidad de las respuestas.
- **Trazabilidad del razonamiento:** Capacidad de verificar el proceso de inferencia del modelo.
- **Reducción de alucinaciones:** Menor tendencia a generar información no fundamentada en el contexto.

C.8.5. Limitaciones y consideraciones

Aunque CoT mejora significativamente el rendimiento, presenta algunas limitaciones:

- **Aumento de latencia:** Los *prompts* más largos requieren mayor tiempo de procesamiento.
- **Consumo de *tokens*:** El razonamiento explícito incrementa el costo computacional.
- **Dependencia de la calidad del contexto:** La efectividad está limitada por la relevancia de los documentos recuperados.

La evaluación experimental demostró que los beneficios en precisión y coherencia compensan estas limitaciones, especialmente en un dominio especializado como el análisis de datos universitarios donde la exactitud es prioritaria.

C.9. Prompt Templates

Los *prompt templates* utilizados en el sistema **RAGDWAREuz** se detallan a continuación y están disponibles en el repositorio del proyecto, organizados por modelo de lenguaje para facilitar su comprensión y reutilización.

C.9.1. Prompt Templates iniciales

Los *prompt templates* iniciales utilizados en las primeras fases de desarrollo del sistema **RAGDWAREuz** están disponibles en el repositorio del proyecto.[7]. Estos sirvieron como base para la iteración y optimización posterior.

C.9.2. Prompt Templates finales

Los *prompt templates* optimizados utilizados en la versión final del sistema se encuentran en los siguientes enlaces:

- **Llama3.2**: Los *prompts* específicamente adaptados para el modelo Llama3.2 se encuentran disponibles en el repositorio.[8].
- **Mistral Small 24B**: La implementación de los *prompts* para el modelo Mistral Small 24B está documentada en el repositorio del proyecto.[9].
- **Qwen2.5 1.5B**: Los *templates* de *prompts* optimizados para el modelo Qwen2.5 1.5B pueden consultarse en el repositorio.[10].

C.10. Ejemplos de ejecución en LangSmith

LangSmith permite visualizar la ejecución de las cadenas y agentes desarrollados con Langchain. A continuación se presentan ejemplos de ejecuciones del sistema **RAGDWAREuz** en LangSmith.

- **Ejemplos de ambito agent en ejecución**: Estos ejemplos ilustran cómo el sistema maneja diferentes tipos de consultas y genera respuestas utilizando los modelos de lenguaje implementados. Los ejemplos incluyen consultas relacionadas con el ámbito académico, I+D+i, recursos humanos, movilidad y estudios propios.[11][12].
- **Ejemplos de ejecución de RAGDWAREuz**: Muestran cómo se procesan las consultas y se generan las respuestas.[13][14].
- **Ejemplos de generación de contexto**: Estos ejemplos ilustran cómo el sistema recupera información relevante de la base de conocimiento y la utiliza para generar respuestas precisas a las consultas del usuario. [15][16].

[7] https://github.com/carloss4dv/RAGDWAREuz/blob/deepeval/prompts.py
[8] https://github.com/carloss4dv/RAGDWAREuz/blob/main/prompts/llama.py
[9] https://github.com/carloss4dv/RAGDWAREuz/blob/main/prompts/mistral.py
[10] https://github.com/carloss4dv/RAGDWAREuz/blob/main/prompts/qwen.py
[11] https://smith.langchain.com/public/c13ff41e-544e-46ad-ae6c-49c1a95d1435/r
[12] https://smith.langchain.com/public/fb4e9e1f-1b8b-4c74-8cbc-2d131c7221b5/r
[13] https://smith.langchain.com/public/fc09f2b6-378c-4b42-a918-ab035f3af5b5/r
[14] https://smith.langchain.com/public/b7082376-2c0a-429f-8cbe-7d31040ae1c8/r
[15] https://smith.langchain.com/public/798cb0e6-616a-43da-ad07-e26c790b0871/r
[16] https://smith.langchain.com/public/994707fd-086b-41d4-8c75-12ee5b55914a/r

C.11. Capturas de Pantalla de la Interfaz

En esta sección se presentan capturas de pantalla de la interfaz de usuario de **RAGDWAREuz**, desarrollada con Chainlit, para ilustrar la interacción del usuario con el sistema. Debido a problemas de memoria en el entorno de desarrollo no se ha podido capturar las consultas que generan sql con la base de datos parcial de DATUZ utilizada en **MDX LEARN**. En su lugar, se muestran videos utilizando sqlite como base de datos. [17]

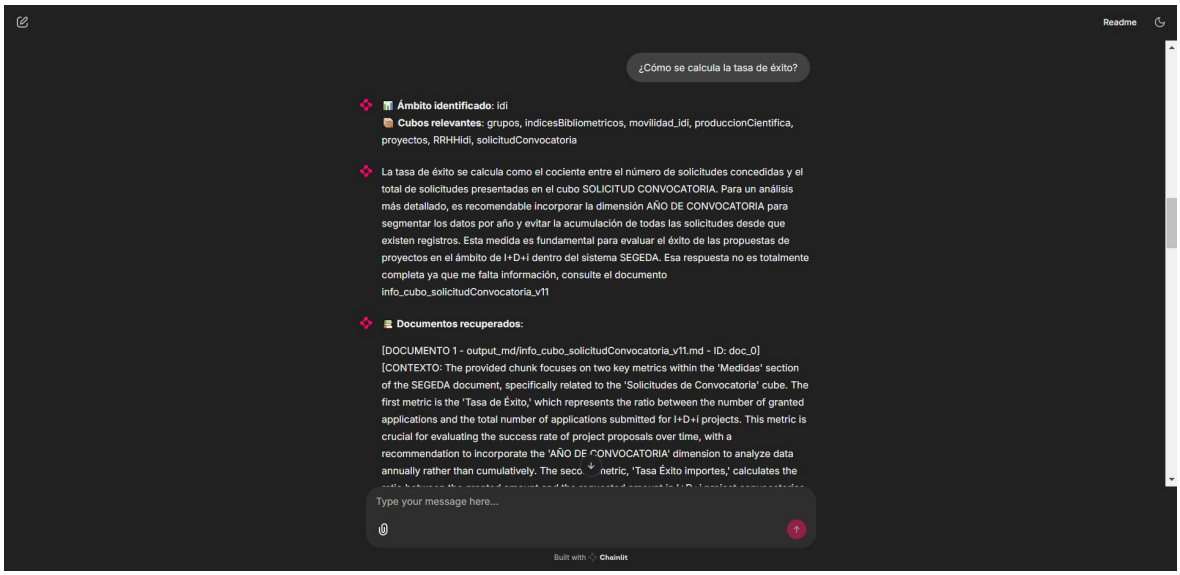

Figura C.4: Ejemplo de consulta y respuesta en RAGDWAREuz.

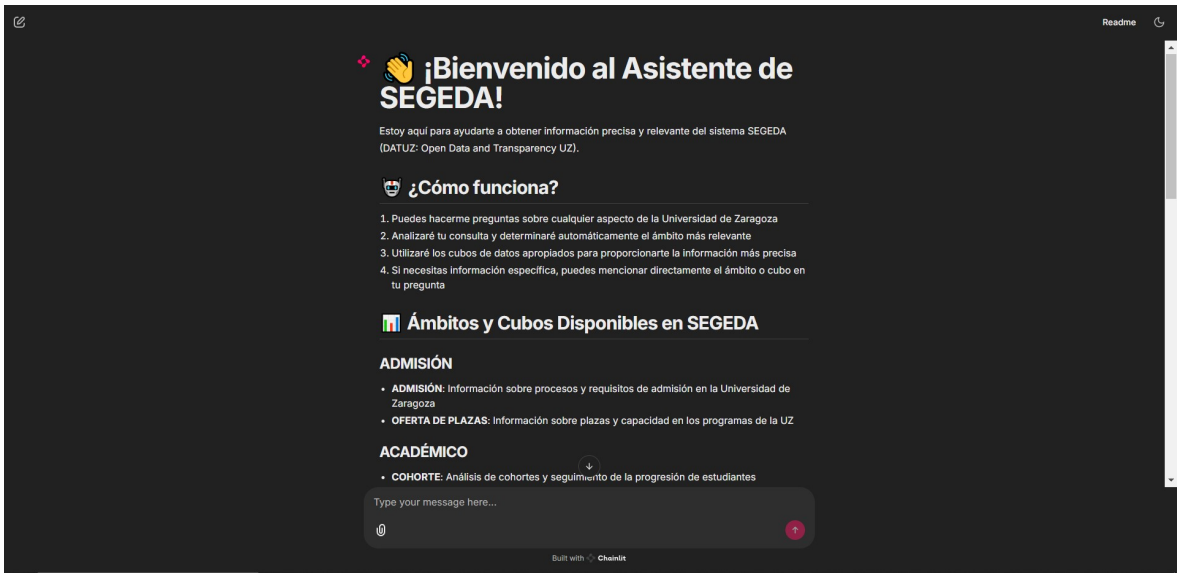

Figura C.5: Mensaje de incio mostrado.

[17] https://drive.google.com/drive/folders/1JwGIcDpw8NqwwAaKsbgg57M3r8DMoSH7?usp=sharing

Anexo D

Implementación de MDX LEARN

D.1. Repositorio del Proyecto

El proyecto **MDX LEARN** está disponible en GitHub.[1]. Este repositorio contiene el código fuente y los recursos necesarios para comprender y replicar el sistema de aprendizaje sobre almacenes de datos MDX, que sirve como base para la implementación de **RAGDWAREuz**.

D.2. Esquemas de Oracle

Los esquemas de las tablas de Oracle utilizados en el proyecto **MDX LEARN** se encuentran disponibles en el repositorio del proyecto.[2].

D.3. Generación de datos sintéticos con Faker

El código utilizado para la generación de datos sintéticos con la librería Faker, que se empleó para crear ejemplos de datos de prueba en los proyectos **MDX_LEARN** y **RAGDWAREuz**, está disponible en el repositorio del proyecto.[3].

D.4. Notebooks de Jupyter

Los *notebooks* de Jupyter utilizados para la enseñanza y práctica de MDX en el proyecto **MDX LEARN** están disponibles en el repositorio del proyecto.[4]. Estos *notebooks* contienen ejemplos prácticos, ejercicios y explicaciones detalladas sobre el uso de MDX para consultas en almacenes de datos multidimensionales.

[1] `https://github.com/carloss4dv/MDX_LEARN`
[2] `https://github.com/carloss4dv/MDX_LEARN/blob/master/data/dm_academico.sql`
[3] `https://github.com/carloss4dv/MDX_LEARN/tree/master/dm_academico_faker`
[4] `https://github.com/carloss4dv/MDX_LEARN/tree/master/notebooks`

D.5. Saiku

En este anexo se encuentran ejemplos de analisis de datos realizados con Saiku. La Figura D.1 muestra un ejemplo de un analisis de la matrícula de los estudiantes por sexo y tipo de estudio. La Figura D.2 muestra un ejemplo de un analisis de la matrícula de los estudiantes por sexo.

La Figura D.3 muestra un ejemplo de un analisis de los creditos totales por rama de estudios. La Figura D.5 muestra un ejemplo de un analisis demografico de los estudiantes por edad y rama. La Figura D.4 muestra un ejemplo de un analisis de la evolución de los creditos de los estudiantes a lo largo de los cursos académicos.

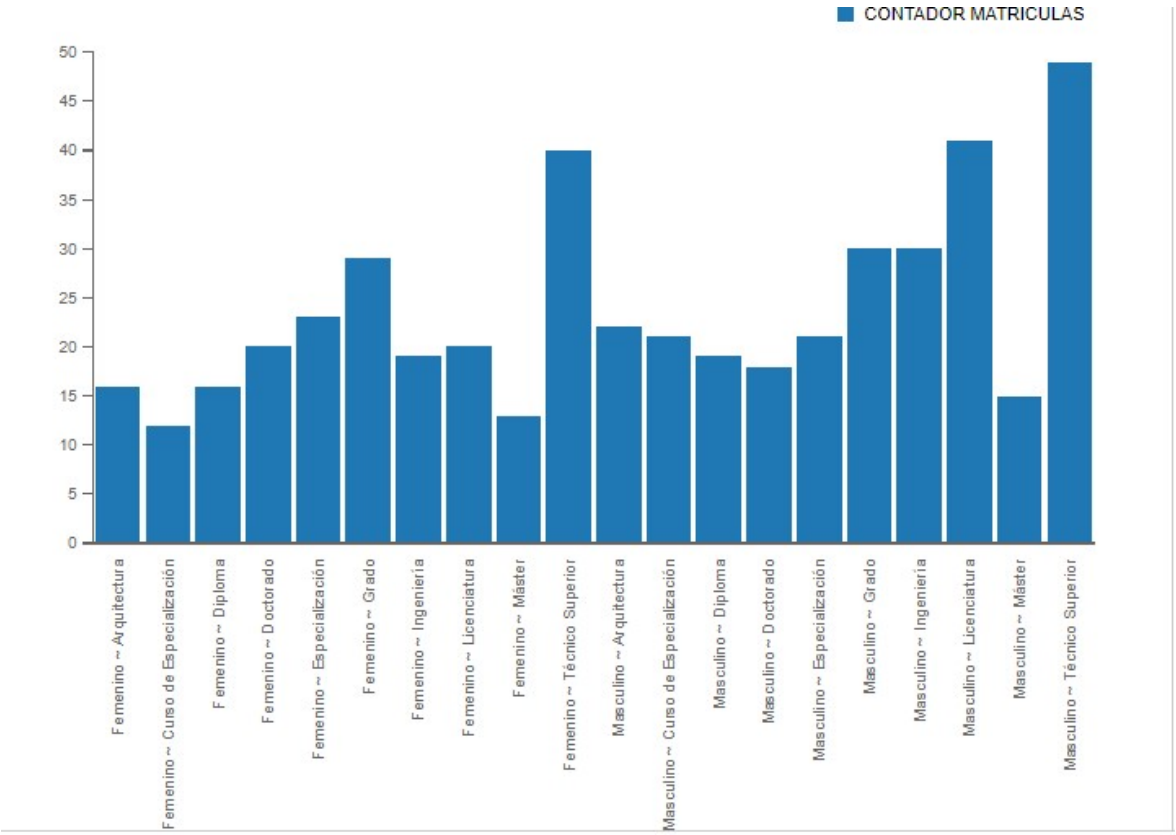

Figura D.1: Ejemplo de un analisis de la matrícula de los estudiantes por sexo y tipo de estudio.

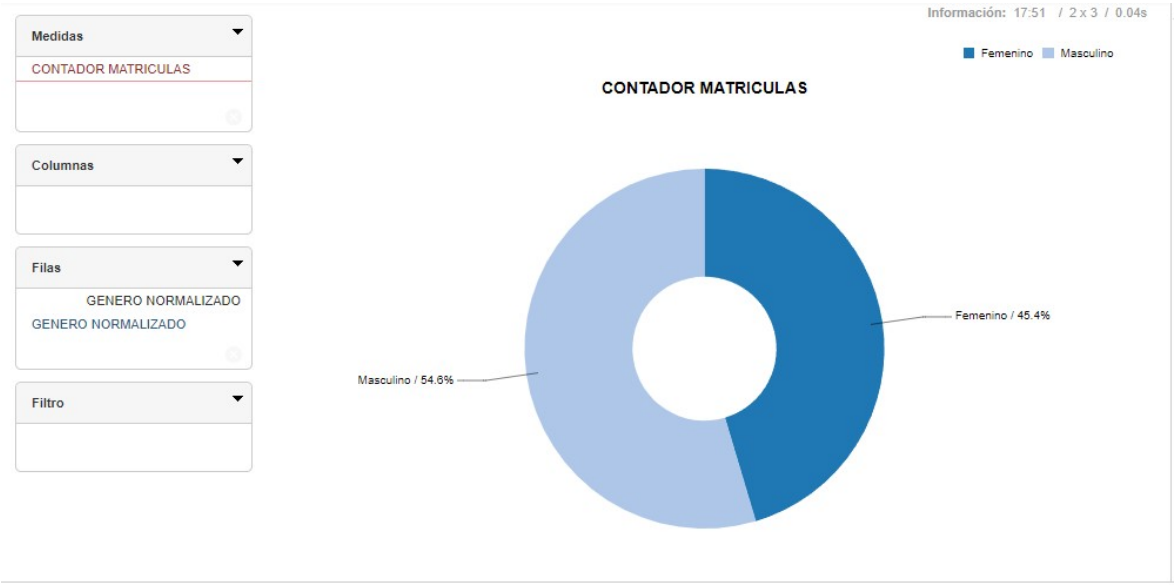

Figura D.2: Ejemplo de un analisis de la matrícula de los estudiantes por sexo.

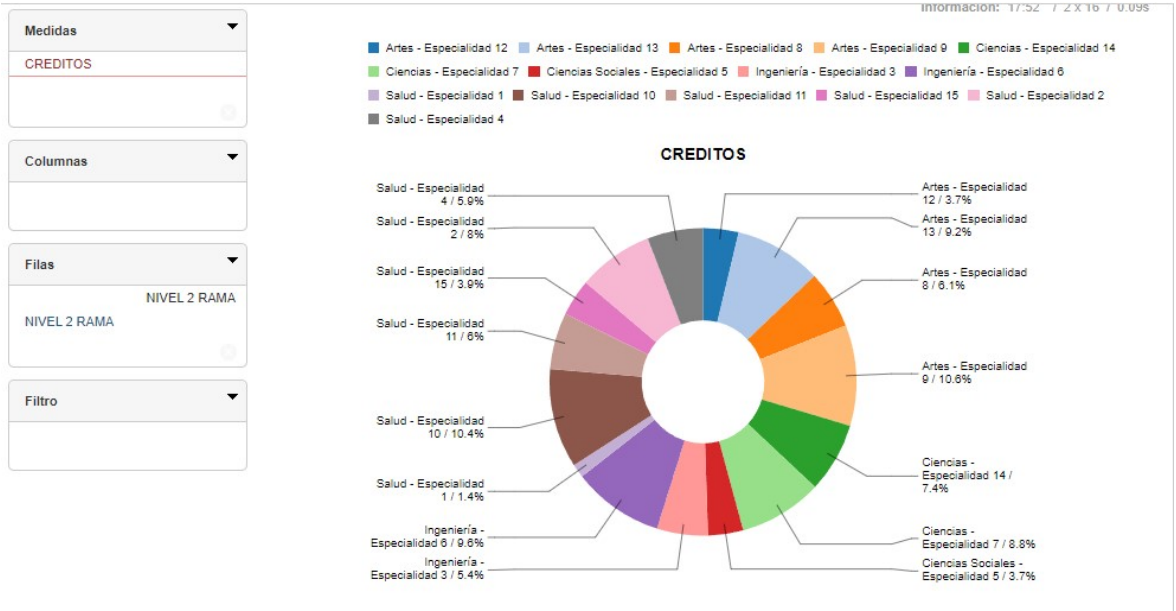

Figura D.3: Ejemplo de un analisis de los creditos totales por rama de estudios.

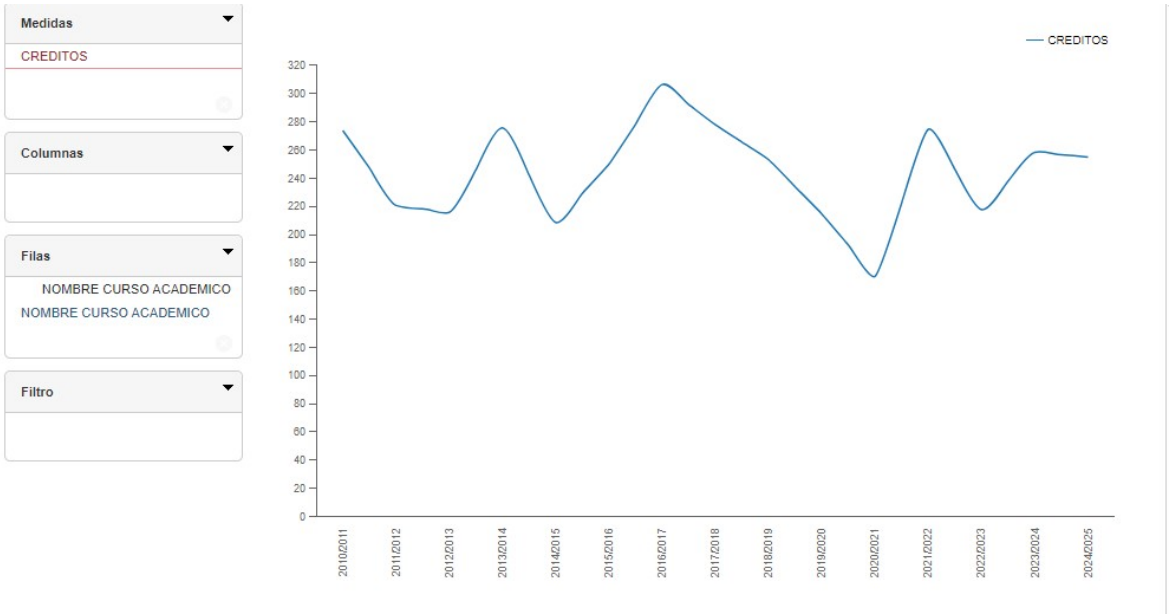

Figura D.4: Ejemplo de un analisis de la evolución de los créditos de los estudiantes a lo largo de los cursos académicos.

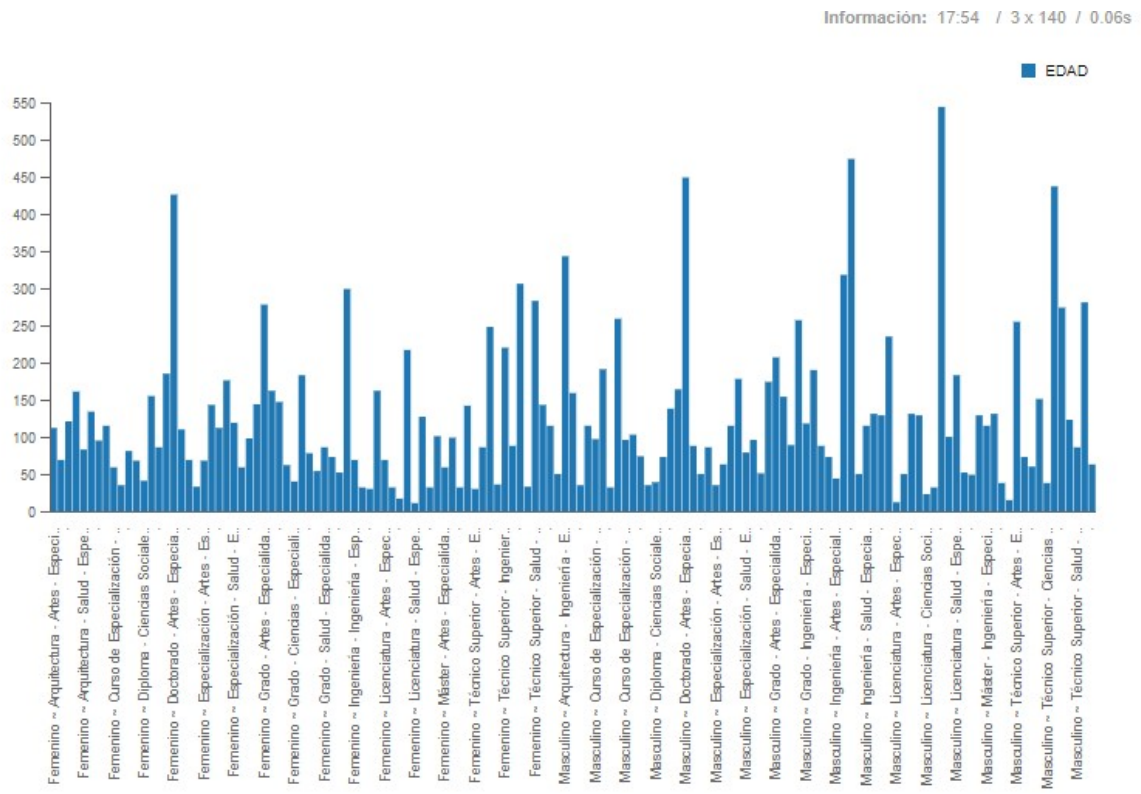

Figura D.5: Ejemplo de un analisis demografico de los estudiantes por edad y rama.

Anexo E

Evaluación del RAGDWAREuz

E.1. Creación de tests con DeepEval

La evaluación sistemática de sistemas RAG (Retrieval-Augmented Generation) requiere métricas especializadas que puedan medir tanto la calidad de la información recuperada como la coherencia de la generación de respuestas. Este anexo detalla la implementación del sistema de evaluación para **RAGDWAREuz** utilizando **Deepeval**. El código completo y los conjuntos de datos de evaluación se encuentran en el repositorio del proyecto.[1].

E.1.1. Arquitectura del sistema de evaluación

La implementación del sistema de evaluación se estructura mediante la clase `AgentEvaluator`, la cual encapsula la funcionalidad de testing automatizado utilizando **Deepeval** como ***framework*** de evaluación. Esta arquitectura permite la evaluación sistemática del agente conversacional desarrollado, proporcionando métricas cuantitativas sobre el rendimiento del sistema.

```
class AgentEvaluator:
    def __init__(self, data_dir=None, vectorstore_dir=None,
                 vector_db_type="milvus", local_llm=None):
        self.agent = LangChainAgent(
            data_dir=data_dir,
            vectorstore_dir=vectorstore_dir,
            vector_db_type=vector_db_type,
            local_llm=local_llm
        )
```

La instanciación del evaluador establece la conexión directa con el agente RAG, permitiendo la ejecución de casos de prueba de manera controlada y reproducible. La configuración modular permite la evaluación con diferentes modelos de lenguaje y configuraciones de bases de datos vectoriales.

E.1.2. Métricas de evaluación implementadas

El sistema implementa cinco métricas fundamentales proporcionadas por **Deepeval**, cada una diseñada para evaluar aspectos específicos del rendimiento del sistema RAG:

[1] `https://github.com/carloss4dv/RAGDWAREuz/tree/main/tests`

Answer Relevancy Metric

Esta métrica evalúa la pertinencia de la respuesta generada en relación con la consulta de entrada. Utiliza técnicas de similaridad semántica para determinar si la respuesta aborda adecuadamente la pregunta formulada.

Faithfulness Metric

La métrica de fidelidad (*Faithfulness*) mide la consistencia entre la respuesta generada y el contexto recuperado de la base de conocimientos. Resulta crucial para sistemas RAG, ya que busca garantizar que las respuestas se fundamenten exclusivamente en la información disponible en los documentos recuperados.

Contextual Precision y Contextual Recall

Estas métricas evalúan la calidad de la recuperación de información. La precisión contextual (*Contextual Precision*) mide la proporción de documentos relevantes entre los recuperados que son efectivamente útiles. El *recall* contextual (*Contextual Recall*) evalúa la capacidad del sistema para recuperar todos los documentos relevantes disponibles en la base de conocimiento para una consulta dada.

Contextual Relevancy Metric

Esta métrica complementa las anteriores evaluando la relevancia global del contexto recuperado en relación con la consulta de entrada, proporcionando una medida holística de la calidad de la recuperación.

```
metrics = [
    deepeval.metrics.AnswerRelevancyMetric(),
    deepeval.metrics.FaithfulnessMetric(),
    deepeval.metrics.ContextualPrecisionMetric(),
    deepeval.metrics.ContextualRecallMetric(),
    deepeval.metrics.ContextualRelevancyMetric()
]
```

E.1.3. Casos de prueba y configuración

El sistema utiliza objetos `LLMTestCase` de deepeval para estructurar los casos de prueba. Cada caso de prueba encapsula la consulta de entrada, la respuesta esperada, la respuesta generada por el sistema, y el contexto recuperado.

```
def convertir_goldens_a_test_cases(self, goldens):
    test_cases = []
    for golden in goldens:
        resultado = self.agent.run(golden.input)
        test_case = LLMTestCase(
            input=golden.input,
            actual_output=generation,
            expected_output=golden.expected_output,
            retrieval_context=context,
            completion_time=tiempo_completado
        )
        test_cases.append(test_case)
    return test_cases
```

La transformación de casos `Golden` a `LLMTestCase` implica la ejecución del agente RAG para cada consulta, la extracción del contexto relevante, y la medición de métricas de rendimiento como el tiempo de completado.

E.1.4. Sistema de checkpointing

Para permitir la reproducibilidad de experimentos, se implementó un sistema de checkpointing que persiste los resultados de evaluación. Este mecanismo previene la re-ejecución innecesaria de evaluaciones costosas computacionalmente.

```
def guardar_checkpoint(self, test_cases: List, preguntas: List[str]):
    checkpoint_data = {
        "test_cases": test_cases,
        "preguntas": preguntas,
        "timestamp": datetime.now().isoformat(),
        "agent_config": {
            "local_llm": self.agent.local_llm,
            "vector_db_type": self.agent.vector_db_type
        }
    }
    with open(archivo_checkpoint, 'wb') as f:
        pickle.dump(checkpoint_data, f)
```

El sistema de checkpointing almacena tanto los casos de prueba como los metadatos de configuración, permitiendo la trazabilidad completa de los experimentos de evaluación.

E.1.5. Configuración asíncrona

Se utiliza configuración asíncrona proporcionada por deepeval, permitiendo la evaluación concurrente de múltiples casos de prueba.

```
results = evaluate(
    async_config=AsyncConfig(throttle_value=5, max_concurrent=10),
    test_cases=casos_evaluables,
    metrics=metrics
)
```

La configuración de throttling y concurrencia permite balancear la eficiencia computacional con las limitaciones de los servicios de modelos de lenguaje, evitando la saturación de APIs externas.

E.1.6. Manejo de casos especiales

El sistema implementa manejo especializado para casos que requieren clarificación adicional, identificando automáticamente consultas ambiguas que no pueden ser evaluadas mediante métricas estándar.

```
if resultado.get("type") == "clarification_needed":
    test_case.clarification_needed = True
    # Excluir de evaluación con métricas
```

Anexo F

Conceptos Teóricos Adicionales

En este anexo se presentan conceptos teóricos adicionales que complementan el trabajo realizado.

F.1. Diagramas de secuencia de arquitecturas RAG avanzadas

En esta sección se presentan las figuras detalladas de los diferentes enfoques y arquitecturas RAG mencionados en el Capítulo 2.

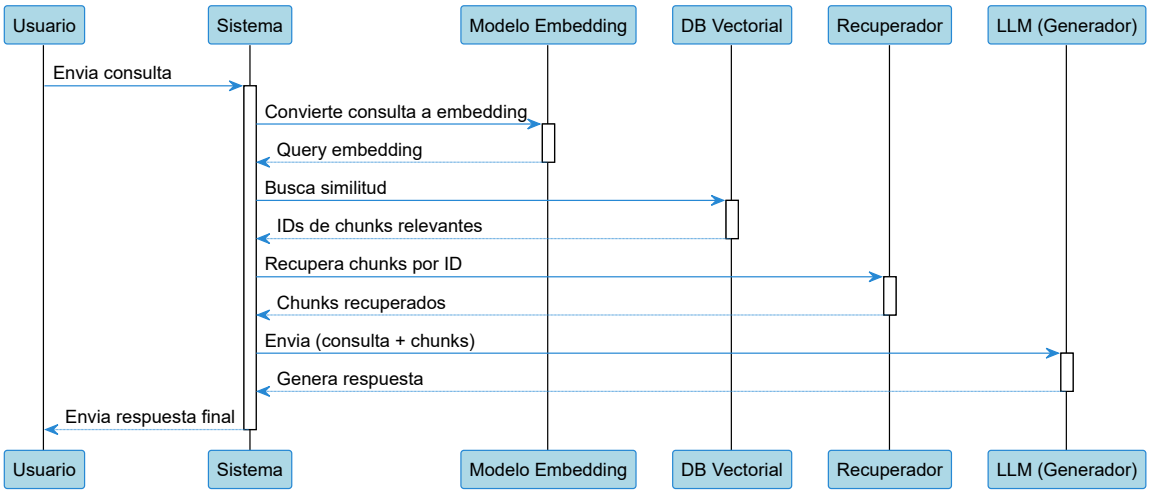

Figura F.1: Flujo de trabajo básico de ***RAG clásico*** o ***naive RAG***

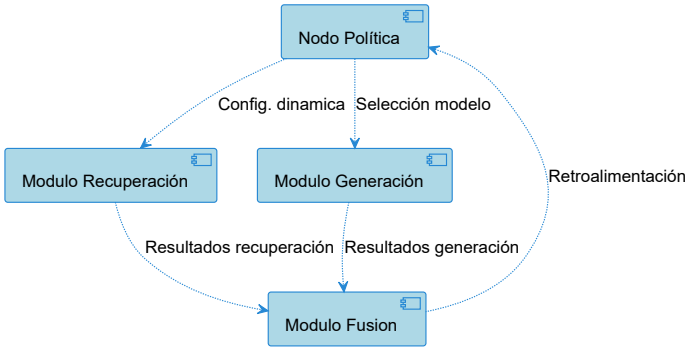

Figura F.2: Arquitectura de ***RAG modular*** con módulos independientes

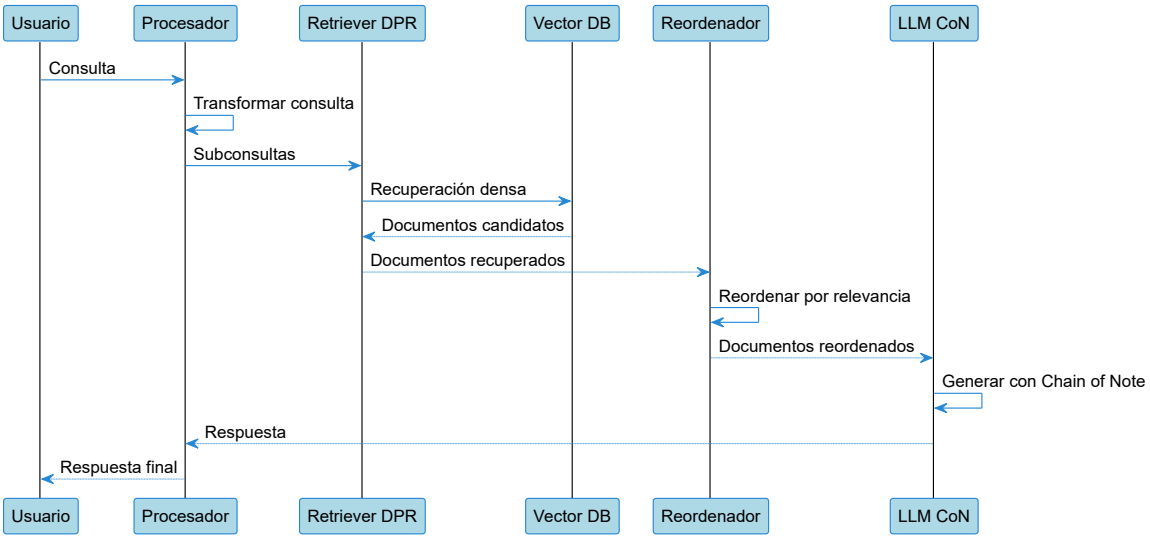

Figura F.3: Flujo de trabajo de ***RAG avanzado***

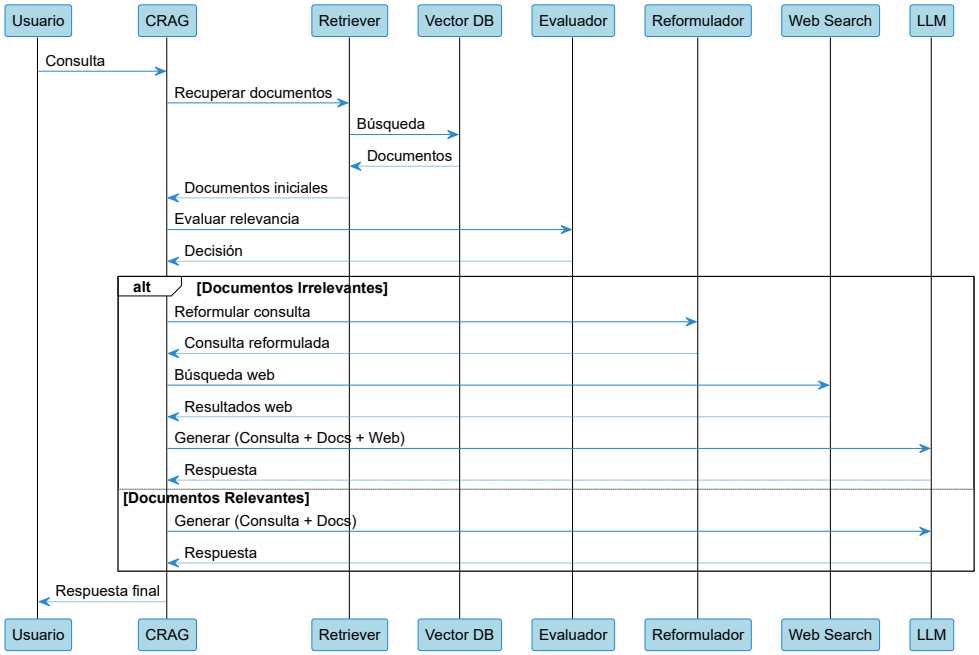

Figura F.4: Flujo de trabajo del ***Corrective Retrieval Augmented Generation*** (**CRAG**)

F.2. El proceso de embeddings y búsqueda semántica

Un ***embedding*** vectorial es una representación numérica de texto en un espacio vectorial de alta dimensión. La característica fundamental de estos ***embeddings*** radica en su capacidad para capturar relaciones semánticas del contenido: fragmentos con significados similares se sitúan en puntos cercanos dentro de dicho espacio. Modelos de redes neuronales específicos generan estos vectores.

En el contexto de RAG, su función resulta esencial. Permiten indexar grandes volúmenes de documentos para realizar búsquedas de similitud semántica posteriormente. Cuando un usuario formula una consulta, esta también se convierte en un vector. El sistema puede entonces encontrar eficientemente los fragmentos de texto más relevantes en la base de conocimiento midiendo la proximidad entre estos vectores. La Figura 2.1 ilustra este proceso.

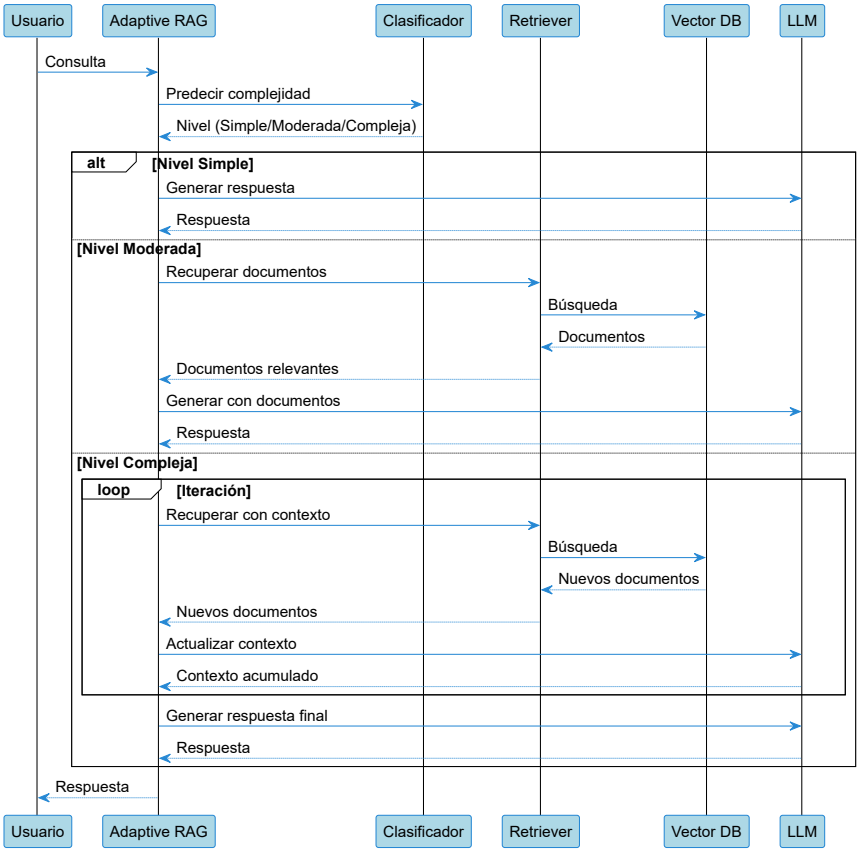

Figura F.5: Flujo de trabajo del ***Adaptive Retrieval Augmented Generation*** (**Adaptive-RAG**)

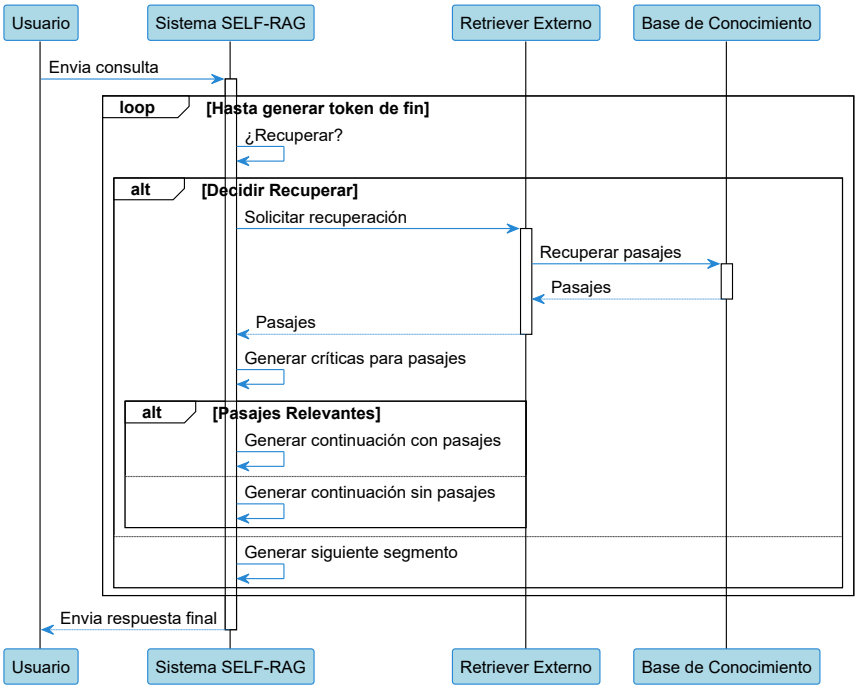

Figura F.6: Flujo de trabajo del ***Self-Retrieval Augmented Generation*** (**Self-RAG**)