



Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**



Máster en Ciencia de Datos

Trabajo Fin de Máster

**SoroIA: Diseño e Implementación de un
Sistema Conversacional basado en
Modelos Grandes de Lenguaje para el
Museo Sorolla**

Autor: Federico Castejón Lozano

Tutores: Pedro Larrañaga Múgica y Concha Bielza Lozoya

Madrid, Julio, 2025

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid.

Este trabajo no es fruto de una colaboración entre la Universidad Politécnica de Madrid y el Museo Sorolla ni la Fundación Museo Sorolla.

Trabajo Fin de Máster
Máster en Ciencia de Datos

Título: SorolA: Diseño e Implementación de un Sistema Conversacional basado en Modelos Grandes de Lenguaje para el Museo Sorolla
Julio, 2025

Autor: Federico Castejón Lozano
Tutores: Pedro Larrañaga Múgica y Concha Bielza Lozoya
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Este trabajo de fin de máster presenta *SoroIA*, un sistema conversacional inteligente basado en modelos grandes de lenguaje (*LLMs*) diseñado para el Museo Sorolla. Su objetivo es mejorar la interacción del usuario con la información de la colección, del autor y del museo, ofreciendo un acceso intuitivo y dinámico al patrimonio cultural público.

El sistema se estructura en tres módulos principales: un clasificador de intención del usuario, un sistema de generación aumentada por recuperación (*RAG*) y un traductor de lenguaje natural a consulta *SQL* (*Text2SQL*) para interactuar con la base de datos estructurada del museo.

La evaluación del clasificador de intención demostró una alta precisión en modo *zero-shot*, subrayando su capacidad para comprender y clasificar las preguntas del usuario sin entrenamiento previo específico. El módulo *RAG* exhibió un rendimiento notable en consultas específicas, con alta relevancia contextual y precisión en respuestas cerradas, aunque se identificaron oportunidades de mejora en la recuperación de información para preguntas más abiertas debido a limitaciones en la recuperación de información relevante. Por otro lado, el módulo *Text2SQL* alcanzó una excelente precisión de ejecución, garantizando la recuperación de datos correcta y sin alucinaciones, destacando la importancia de la ingeniería de *prompts* para guiar al *LLM*.

SoroIA fue desarrollado y desplegado sobre una arquitectura en la nube, lo que asegura su escalabilidad, baja latencia y disponibilidad para un uso público. En conjunto, este trabajo demuestra la efectividad del *pipeline* diseñado y la integración de *LLMs* en entornos culturales, proporcionando una solución innovadora para el acceso a las ricas fuentes de información museísticas públicas.

Palabras clave: Inteligencia Artificial Generativa, Inteligencia Artificial Conversacional, Sistemas Conversacionales, Modelos Grandes de Lenguaje, Generación Aumentada por Recuperación, Amazon Web Services, Museos

Abstract

This master's thesis presents *SoroIA*, an intelligent conversational system based on large language models (*LLMs*) designed for the Sorolla Museum. Its objective is to improve user interaction with information about the collection, the author and the museum, offering an intuitive and dynamic access to public cultural heritage.

The system is structured in three main modules: a user intent classifier, a retrieval-augmented generation (*RAG*) system and a natural language to *SQL* query translator (*Text2SQL*) to interact with the structured database of the museum.

Evaluation of the intent classifier demonstrated high accuracy in zero-shot mode, highlighting its ability to understand and classify user queries without specific prior training. The *RAG* module exhibited remarkable performance on specific queries, with high contextual relevance and accuracy on closed answers, although opportunities for improvement in information retrieval for more open-ended questions were identified due to limitations in retrieving relevant information. On the other hand, the *Text2SQL* module achieved excellent execution accuracy, ensuring correct and hallucination-free data retrieval, highlighting the importance of prompts engineering to guide the *LLM*.

SoroIA was developed and deployed on a cloud architecture, ensuring scalability, low latency and availability for public use. Overall, this work demonstrates the effectiveness of the designed pipeline and the integration of *LLMs* in cultural environments, providing an innovative solution for accessing rich public museum information sources.

Keywords: Generative Artificial Intelligence, Conversational Artificial Intelligence, Conversational Assistants, Large Language Models, Retrieval-Augmented Generation, Text2SQL, Amazon Web Services, Museums

Agradecimientos

Quiero agradecer a mis padres, Luis y Beatriz, por darme la oportunidad de estudiar en Madrid y motivarme siempre a seguir adelante, dándome fuerzas en todo momento. Gracias por inculcarme vuestros valores.

A mis hermanos, Luis, Pedro y Beatriz, por ser únicos y ser los mejores amigos que uno puede tener.

A mis abuelos, Pedro y Margarita, y Federico y Juana, por formar una familia tan especial y tan unida. Gracias a mi familia por enseñarme a apreciar el arte.

A Joaquín Sorolla, por retratar mi querida Jávea con una fidelidad y belleza que me han inspirado en todo este proceso.

Por último, quiero agradecer a mis tutores, Concha y Pedro, por vuestra cercanía y por hacer de cada reunión un espacio de aprendizaje.

Tabla de contenidos

1. Introducción	1
1.1. Contexto	1
1.2. Objetivos	2
1.3. Estructura del documento	2
2. Estado del arte	5
2.1. Museos en el siglo XXI	5
2.2. Modelos grandes de lenguaje	7
2.3. Clasificación de preguntas con <i>LLMs</i>	11
2.4. <i>Retrieval-augmented generation</i> (RAG)	13
2.5. Traducción de lenguaje natural a <i>SQL</i>	17
2.6. <i>Web scraping</i> como acceso a fuentes de conocimiento	18
3. Desarrollo	21
3.1. <i>Scraping</i>	21
3.1.1. Elección de una fuente de conocimiento	21
3.1.2. Desarrollo de código de <i>scraping</i>	22
3.1.3. Limpieza	24
3.1.4. Ingesta en una base de datos	24
3.1.5. Información de la página web del museo	24
3.2. Desarrollo de <i>SoroIA</i>	25
3.2.1. Clasificador de la intención	26
3.2.2. Tarea asignada	28
3.2.2.1. Traducción de la consulta a <i>SQL</i>	28
3.2.2.2. <i>RAG</i>	29
3.2.3. Generación de la respuesta final	30
3.2.4. Interfaz	31
3.3. Diseño y despliegue de la arquitectura en la nube	32
3.3.1. Almacenamiento de las imágenes: <i>AWS S3 Bucket</i>	33
3.3.2. Base de datos: <i>AWS RDS</i>	33
3.3.3. Alojamiento del servicio: <i>AWS Elastic Compute Cloud</i>	34
3.3.4. Conectividad	34
3.3.5. Acceso a servicios externos	35
3.3.6. Flujo general	35
3.3.7. Escalabilidad	36
4. Evaluación	37
4.1. Clasificación de la intención	37

4.2. Evaluación del sistema RAG	41
4.2.1. Evaluación con preguntas abiertas	42
4.2.2. Evaluación de preguntas con respuesta esperada	45
4.3. Evaluación de la traducción a consulta SQL	47
4.4. Evaluaciones adicionales	49
5. Conclusiones	55
5.1. Objetivos cumplidos	55
5.2. Conclusiones obtenidas	55
5.3. Limitaciones y líneas futuras	56
6. Impacto y consideraciones éticas	59
Bibliografía	63
A. Conjunto de preguntas para la evaluación	65
B. Gastos asociados a la infraestructura en la nube	71

Capítulo 1

Introducción

1.1. Contexto

En la era actual, la digitalización y la inteligencia artificial (IA) están transformando la forma en que las instituciones culturales, como los museos, interactúan con su público y gestionan su patrimonio. Los museos buscan constantemente innovar en cómo presentar sus colecciones y acercarse a los visitantes. Tradicionalmente, el acceso a información detallada sobre objetos, artistas o contextos históricos ha dependido de guías físicas, audioguías, paneles informativos o personal especializado, lo cual, aunque valioso, puede limitar la inmediatez y la personalización de la interacción. Además, la gran cantidad de información sobre colecciones y exhibiciones suele estar dispersa en repositorios de bases de datos públicas y en documentos no estructurados (textos, biografías, archivos históricos), lo que dificulta su acceso directo y rápido por parte de usuarios no técnicos.

Este trabajo de fin de máster (TFM) se sitúa en este contexto de innovación, conectando de una manera inteligente el usuario y la información museística. Se propone el desarrollo de *SoroIA*, un sistema conversacional diseñado específicamente para el Museo Sorolla. *SoroIA* busca enriquecer la experiencia del usuario al ofrecer un acceso intuitivo y dinámico a la información, permitiendo a los visitantes explorar la colección y la historia del museo mediante el lenguaje natural. Para lograr esta capacidad, el sistema integra capacidades avanzadas de procesamiento de lenguaje natural (NLP) y modelos grandes de lenguaje (LLMs), definiendo tres módulos clave: un clasificador de intención para comprender la demanda del usuario, un sistema de generación aumentada por recuperación (RAG) para proporcionar información contextual a partir de fuentes documentales y un traductor de lenguaje natural a consulta SQL (*Text2SQL*) para interactuar con la base de datos estructurada de la colección.

La información base para estos módulos, incluyendo los datos detallados de la colección y los textos de apoyo, fue obtenida mediante técnicas de *web scraping* de datos públicos de Colecciones En Red españolas (*CER.es*) y otras fuentes relevantes como la propia web del museo o *Wikipedia*.

La integración de estos módulos en una arquitectura desplegada en la nube busca no solo la precisión en la respuesta, sino también una experiencia de usuario fluida y escalable en un futuro.

1.2. Objetivos

El objetivo principal de este *TFM* es diseñar, implementar y evaluar un sistema conversacional inteligente, *SoroIA*, capaz de interactuar con los usuarios en lenguaje natural para proporcionar información relevante y precisa sobre la colección y los objetos del Museo Sorolla, ubicado en Madrid.

Para alcanzar este objetivo general, se establecen los siguientes objetivos específicos:

- **Aplicar técnicas de *web scraping*** para descargar los datos sobre el Museo Sorolla ofrecidos en la rica fuente *CER.es* del Ministerio de Cultura, accesible públicamente.
- Desarrollar un sistema de **clasificación de intenciones**, capaz de identificar automáticamente de qué manera debe responderse a cada consulta dependiendo de unos filtros establecidos, negando la respuesta en casos no definidos dentro del caso de uso.
- Integrar un **LLM para traducción a SQL (*Text2SQL*)**, capaz de generar consultas *SQL* precisas a partir de preguntas en lenguaje natural, sin necesidad de entrenamiento específico.
- Implementar un **sistema de RAG** que permita responder preguntas mediante la recuperación semántica de información textual procedente de documentos relacionados con el museo (*Wikipedia* y web oficial del museo).
- **Unificar ambos enfoques (*SQL* y *RAG*)** en un único flujo conversacional, permitiendo una experiencia de usuario fluida, coherente y útil para visitantes, investigadores o cualquier persona interesada en el museo.
- **Desplegar el sistema en un entorno en la nube** utilizando tecnologías de *Amazon Web Services (AWS)* para que sea accesible de forma remota y escalable en el futuro.
- **Evaluar el rendimiento del sistema** mediante métricas novedosas tanto para la clasificación de intenciones como para la calidad de las respuestas generadas mediante *LLMs*.

1.3. Estructura del documento

La estructura del documento está dividida de la siguiente manera:

- *Capítulo 1: Introducción:* Se explica el contexto del proyecto y los objetivos fijados.
- *Capítulo 2: Estado del arte:* Revisión del estado del arte de las nuevas tecnologías en el contexto museístico, así como revisión de la literatura de los *LLMs*, *RAG* y *Text2SQL*.
- *Capítulo 3: Desarrollo:* Diseño y desarrollo de *SoroIA* utilizando técnicas de *RAG* y *Text2SQL*, desde la versión local hasta la migración de la arquitectura en la nube.
- *Capítulo 4: Evaluación:* Elección de una metodología para evaluar el rendimiento de *SoroIA* en cada uno de sus módulos, en diferentes contextos y con diferentes

Introducción

métricas. Se incluye la exposición de los resultados obtenidos.

- *Capítulo 5: Conclusiones:* Exposición de las conclusiones obtenidas del proyecto y su rendimiento, así como las limitaciones que se han encontrado y las líneas futuras de investigación que se han identificado.
- *Capítulo 6: Impacto:* Análisis del impacto que tiene el proyecto desde un punto de vista social, ético, medioambiental y cultural.
- *Bibliografía:* Documentación utilizada para el desarrollo del TFM.
- *Anexos:* Se incluyen tablas del conjunto de datos usado en la evaluación y los gastos asociados al proyecto.

Capítulo 2

Estado del arte

Este capítulo trata de abordar el estado del arte de las nuevas tecnologías en los museos. También se ahonda en las tecnologías de *LLMs* más utilizadas y las técnicas que permiten utilizarlos en combinación con otros módulos.

2.1. Museos en el siglo XXI

La revolución tecnológica de las últimas décadas ha impactado todos los sectores culturales, y los museos no han sido una excepción. Lejos de ser espacios estáticos y silenciosos, los museos del siglo XXI están evolucionando hacia entornos dinámicos, interactivos y más inclusivos, impulsados por tecnologías emergentes como la *IA*, la realidad aumentada, la web semántica, la visión por computador y, más recientemente, los *LLMs*. Estos avances permiten repensar el rol del museo como mediador entre el conocimiento y la experiencia del público.

Digitalización, acceso y virtualización de museos

Uno de los primeros pasos que muchas instituciones dieron con la llegada de las nuevas tecnologías fue la digitalización de sus colecciones. Esta práctica, inicialmente centrada en la conservación, ha evolucionado hacia nuevos usos: la creación de catálogos interactivos, motores de búsqueda visuales, y recorridos virtuales. Un ejemplo relevante es la visita virtual del **Museo Sorolla**, que permite al usuario recorrer las salas del museo y observar las obras en alta resolución desde cualquier parte del mundo¹.

De forma más avanzada, el **Museo del Prado** de Madrid ha desarrollado una línea del tiempo interactiva vinculada a una ontología del museo basada en datos abiertos como *Wikipedia* y *Wikidata*. Este recurso permite a los visitantes explorar las obras junto a su contexto histórico, político y cultural, creando una experiencia inmersiva y contextualizada desde una página *web* donde puede navegar en las obras ordenadas temporalmente². Asimismo, el museo ha aplicado técnicas de visión por computador para tareas analíticas, como el conteo automático de personajes en cuadros complejos como *El jardín de las delicias*, de El Bosco, que presenta desafíos incluso para

¹<https://www.cultura.gob.es/msorolla/exposicion/visita-virtual.html>

²<https://www.museodelprado.es/grafico-de-conocimiento/linea-de-tiempo>

expertos humanos³. Otra iniciativa destacada dentro del museo es *El Prado de Babel*, que mejora la accesibilidad multilingüe traduciendo automáticamente las explicaciones de las visitas guiadas a seis idiomas mediante tecnología de procesamiento de voz⁴.

Otros museos como el **Metropolitan Museum of Art (MET)** de Nueva York han apostado por la publicación de APIs para promover la exploración de sus colecciones⁵.

Aplicaciones de IA y modelos de lenguaje

Más allá de la digitalización, la aplicación de sistemas inteligentes ha comenzado a transformar la interacción entre el visitante y el contenido museográfico. En el ámbito hispanohablante, los avances han sido más limitados, con falta de ejemplos de IA más avanzados.

A nivel internacional, en Estados Unidos, el **Smithsonian Institution** investiga en la IA Generativa de LLMs para enriquecer la navegación por sus archivos⁶. Además, **MoMA y Google Arts & Culture Lab** utilizaron IA basada en visión artificial para identificar más de 20,000 obras en fotos de exposiciones desde 1929. El proyecto conectó imágenes históricas con la colección en línea, creando nuevas relaciones entre obras, épocas y exposiciones⁷.

Un caso destacado es la startup francesa **Ask Mona**, que ha desarrollado asistentes conversacionales para museos y teatros. Uno de sus proyectos más conocidos permite dialogar con una versión ficticia de la mujer del retrato *Mona Lisa*, donde un LLM responde de forma contextual y creativa a preguntas de los visitantes, combinando información histórica con una personalidad simulada⁸. El **Musée d'Orsay** (París) ha creado una herramienta que permite conversar con el pintor Van Gogh sobre sus obras utilizando IA generativa⁹.

Otro ejemplo notable es el *Living Museum*, una web no oficial basada en el **British Museum** de Londres, donde los objetos cobran voz mediante un sistema de IA que genera respuestas en primera persona. Esta forma de personificación transforma la visita en una experiencia narrativa, educativa y emocional¹⁰. Se trata de un caso de uso similar al presente trabajo.

Hacia un museo inteligente: retos y oportunidades

La utilización de LLMs para crear interfaces conversacionales abre un horizonte de posibilidades para construir un museo inteligente, donde el visitante no solo consume contenido, sino que interactúa con él. Este tipo de soluciones permite:

- Personalizar la experiencia del visitante según su perfil o intereses.

³<https://www.museodelprado.es/actualidad/noticia/el-museo-del-pradopone-a-la-inteligencia/6d362d3a-6f0b-320b-d913-20a44efb3471>

⁴<https://www.museodelprado.es/actualidad/noticia/el-prado-de-babel-unainiciativa-para-celebrar/5d828013-5436-78b4-1f53-95622a6d2f52>

⁵<https://metmuseum.github.io/>

⁶<https://www.si.edu/support/impact/age-of-ai>

⁷<https://www.moma.org/calendar/exhibitions/history/identifying-art>

⁸<https://www.askmona.ai/culture>

⁹<https://www.musee-orsay.fr/es/node/275618>

¹⁰<https://livingmuseum.app/>

- Traducir y adaptar contenidos para visitantes internacionales, con discapacidad o expertos.
- Proporcionar explicaciones contextualizadas, dialogadas y adaptativas.
- Integrar datos estructurados (catálogos) y no estructurados (textos históricos, audios) en un mismo flujo informativo.

Además, redes como la **AI Museums Network**¹¹ (entre los que están el *Museo del Prado*, *National Gallery* o *Metropolitan Museum of Art*) están promoviendo marcos éticos y guías técnicas para facilitar la integración responsable de la IA en entornos museísticos, con especial atención a la equidad, la transparencia y la preservación del rol de las personas.

En este contexto, el desarrollo de una herramienta basada en *LLMs* como la planteada en este TFM, se alinea con las tendencias internacionales más innovadoras y crea una vía para instituciones que aún no disponen de grandes recursos tecnológicos.

2.2. Modelos grandes de lenguaje

Los modelos grandes de lenguaje (*LLMs*, por sus siglas en inglés, *large language models*) son programas avanzados diseñados para generar y comprender texto en lenguaje natural. Estos modelos, como GPT¹², Gemini¹³ o LLaMA¹⁴, se caracterizan por haber sido preentrenados con una gigantesca cantidad de datos procedente principalmente de *internet*. Esta sección se fundamentará principalmente en el curso introductorio de *LLMs* de *Hugging Face*¹⁵.

La trayectoria hacia los *LLMs* actuales es un reflejo del crecimiento exponencial en el campo del *NLP*. Previamente a los *LLMs*, los modelos de lenguaje se basaban en arquitecturas como las redes neuronales recurrentes (*RNN*) y sus variantes, como las *long short-term memory (LSTM)*, que mejoraron la gestión de la memoria, aunque se enfrentaban a desafíos con secuencias largas. La introducción de *embeddings* de palabras como *Word2Vec* (Mikolov et al., 2013) marcó un progreso significativo al representar palabras como vectores numéricos.

Sin embargo, el punto de inflexión fue la introducción de la arquitectura *transformer* (Vaswani et al., 2017). Los *transformers*, con su mecanismo de “atención” (*self-attention*), permiten el procesamiento paralelo de secuencias y una mejor captura de dependencias a largo plazo.

Utilizando este gran avance, surgen otros modelos como *Bidirectional Encoder Representations from Transformers (BERT)* (Devlin et al., 2019) y los modelos *Generative Pre-trained Transformer (GPT)* (Radford et al., 2018), que demostraron la eficacia del preentrenamiento en grandes *corpus* y el *fine-tuning* para tareas específicas. La culminación de esta trayectoria se observa en modelos como GPT-3 (Brown et al., 2020), con la salida de la aplicación *ChatGPT*.

¹¹<https://themuseumsai.network/toolkit/>

¹²<https://openai.com/es-ES/chatgpt/overview/>

¹³<https://gemini.google/overview/>

¹⁴<https://www.llama.com/>

¹⁵<https://huggingface.co/learn/llm-course/es/chapter1/4>

Arquitectura y principios de funcionamiento

La base arquitectónica de la mayoría de los *LLMs* modernos es el *transformer* (Vaswani et al., 2017). Esta arquitectura permite a los modelos procesar grandes volúmenes de texto de manera eficiente y capturar dependencias a largo plazo dentro de las secuencias de palabras. El *transformer* consta de dos componentes principales: un codificador (*encoder*) y un decodificador (*decoder*), aunque la mayoría de *LLMs* actuales utilizan variantes, lo que da lugar a la clasificación de *LLMs* general:

- **Modelos solo *encoder*:** Son modelos bidireccionales. Esto significa que procesan el texto en ambas direcciones, de izquierda a derecha y de derecha a izquierda, de tal forma que están especializados en la comprensión profunda del contexto. Son ideales para tareas como clasificación de texto o respuestas a preguntas donde la respuesta está contenida en el texto fuente. En esta categoría se encuentran modelos como BERT.
- **Modelos solo *decoder*:** También llamados modelos autorregresivos, son modelos unidireccionales y están diseñados para generar texto de una manera secuencial, prediciendo el siguiente *token* basándose en los tokens previos. Utilizados para generar contenido, continuar texto o conversaciones. Aquí se encuentra el modelo más famoso, GPT.
- **Modelos *encoder-decoder*:** Estos modelos usan ambas unidades, lo que les hace versátiles para tareas que impliquen comprender una entrada y generar una salida, como traducir o resumir. Dentro de esta categoría, se encuentran T5 (Raffel et al., 2023) o BART (Lewis et al., 2019).

El funcionamiento de un *LLM* tiene varios pasos clave:

- **Tokenización:** El texto de entrada se convierte en unidades más pequeñas llamadas *tokens*. Estos *tokens* pueden ser palabras, partes de palabras o incluso caracteres.
- **Embeddings:** Cada *token* se transforma en un vector numérico denso, conocido como *embedding*. Estos *embeddings* capturan el significado semántico y las relaciones contextuales del *token*.
- **Mecanismo de atención:** Este es el componente fundamental del *transformer*. Permite al modelo ponderar la importancia de diferentes *tokens* en la secuencia de entrada al procesar cada *token*. De esta forma, el modelo puede entender cómo las palabras se relacionan entre sí a lo largo de una frase, independientemente de su posición, construyendo una representación rica del texto.
- **Decodificación (en modelos generativos):** Una vez procesada la entrada, los modelos autorregresivos generan texto prediciendo el siguiente *token*, hasta que se completa la respuesta. Este proceso se basa en las probabilidades calculadas por el modelo.

Entrenamiento de *LLMs*

El proceso de entrenamiento de un *LLM* se divide generalmente en dos fases:

- **Preentrenamiento:** Los modelos son expuestos a cantidades enormes de texto sin etiquetar y aprenden a predecir la siguiente palabra en una secuencia, a

rellenar palabras enmascaradas y a predecir relaciones entre oraciones. Este proceso no supervisado dota al modelo de un conocimiento lingüístico y factual muy amplio. Requiere un gran *corpus* de datos y el entrenamiento puede tomar varias semanas.

- **Ajuste fino (*fine-tuning*):** Tras el preentrenamiento, los *LLMs* pueden ser ajustados para tareas específicas mediante *fine-tuning* supervisado con conjuntos de datos más pequeños y etiquetados. Más recientemente, surgen técnicas como el aprendizaje por refuerzo a partir de la retroalimentación humana (del inglés *reinforcement learning from human feedback, RLHF*). (Li et al., 2023), que consiste en introducir la evaluación humana a la hora de hacer el *fine-tuning*. Este paso es crucial para determinar las preferencias del modelo, mejorando su utilidad, seguridad y el seguimiento de instrucciones desde un punto de vista humano.

Capacidades y aplicaciones

Los *LLMs* han demostrado una versatilidad sin precedentes en una multitud de tareas de *NLP* y más allá, gracias a sus habilidades emergentes que se manifiestan a medida que aumenta la escala del modelo. Sus capacidades incluyen:

- **Generación de texto:** Crear contenido coherente y creativo, como artículos, historias, correos electrónicos, y código de programación.
- **Comprensión del lenguaje:** Responder preguntas (*QA*), resumir documentos, extraer información y realizar análisis de sentimientos, traducir texto entre diferentes idiomas.
- **Razonamiento y resolución de problemas:** Aunque de forma limitada (Wang et al., 2025), los *LLMs* pueden realizar tareas de razonamiento lógico, matemáticas básicas y resolución de problemas, especialmente con técnicas como el *chain-of-thought prompting* (Wei et al., 2023).
- **Conversación:** Mantener diálogos coherentes y contextualizados, lo que ha impulsado el desarrollo de chatbots avanzados y asistentes virtuales.
- **Capacidades multimodales:** En los modelos más recientes, se extiende la capacidad de procesar y generar no solo texto, sino también otros tipos de datos como imágenes, audio y vídeo (Yin et al., 2024).

Estas capacidades se ven fuertemente influenciadas por técnicas de ingeniería de *prompts* (*prompt engineering*) (Vatsal and Dubey, 2024), que implican el diseño cuidadoso de las instrucciones y preguntas de entrada para guiar al *LLM* hacia las respuestas deseadas, maximizando su precisión y relevancia para tareas específicas. Además, el parámetro de *temperatura* juega un rol importante en la generación de texto, controlando la aleatoriedad de las respuestas: valores más bajos (cerca de 0) producen resultados más deterministas y consistentes, mientras que valores más altos fomentan una mayor creatividad y diversidad en la generación con riesgo de producir alucinaciones (Renze and Guven, 2024).

Estas capacidades han permitido su aplicación en diversos dominios, desde asistentes de escritura y herramientas de código de programación, hasta *chatbots* de servicio al cliente, investigación científica y, como es el caso de este trabajo, en sistemas de asistencia conversacional en contextos especializados como los museos.

Modelos relevantes de código abierto

Aunque muchos *LLMs* son de acceso restringido, el ecosistema de código abierto ha avanzado rápidamente. Algunos de los modelos más relevantes para entornos como el del presente TFM son:

- **LLaMA 3 (Meta):** Modelo versátil de alto rendimiento disponible en variantes de 8B y 70B parámetros (Grattafiori et al., 2024). Para el presente trabajo, se ha optado por LLaMA 3 con 70 mil millones de parámetros a través de la API de Groq, por su rapidez y bajo coste.
- **Mistral:** Modelos basados en la arquitectura *mixture-of-experts*, eficientes para tareas de generación y clasificación (Jiang et al., 2023).
- **Phi-3 (Microsoft):** Diseñado para funcionar en dispositivos con pocos recursos, útil para soluciones embebidas o *edge*, por ejemplo en dispositivos móviles (Abdin et al., 2024).

Desafíos y consideraciones éticas

A pesar de su rendimiento tan asombroso y efectivo, los *LLMs* presentan unos riesgos y limitaciones que deben ser tenidos en cuenta a la hora de utilizarlos en proyectos reales:

- **Alucinaciones:** Este es uno de los problemas más comunes en *LLMs*. Los modelos pueden generar información incorrecta, inventada o sin fundamento, presentándola como verdadera (Huang et al., 2025). También es común las alucinaciones a la hora de citar bibliografías. Por tanto, se trata de un riesgo muy importante en una tarea que requiere alta precisión.
- **Sesgo:** Los modelos heredan los sesgos que se encuentran en los conjuntos de entrenamiento, donde pueden existir actitudes discriminatorias, injustas o perjudiciales.
- **Coste computacional:** El entrenamiento y ejecución de *LLMs* requiere una gran cantidad de recursos computacionales y energéticos.
- **Transparencia y explicabilidad:** La complejidad de los *LLMs* los convierte en “cajas negras”, lo que dificulta entender por qué generan una respuesta específica o cómo llegan a una conclusión.
- **Seguridad y robustez:** Pueden ser vulnerables a ataques o *jailbreaking*, donde *prompts* pueden evitar los filtros de seguridad del modelo para generar contenido dañino o inapropiado. La implementación de guardarrailes es esencial para mitigar estos riesgos (Dong et al., 2024).
- **Conocimiento desactualizado:** El conocimiento de un *LLM* está limitado al momento de su entrenamiento, lo que limita su capacidad para acceder a información muy reciente. Esta limitación es la que motivó el desarrollo de técnicas como RAG, como se verá en la Sección 2.4.

Superar estos desafíos es un área activa de investigación y desarrollo, y es crucial para la implementación responsable y efectiva de los *LLMs* en el mundo real.

Aplicación en un caso de uso

La aplicación de *LLMs* en entornos museísticos es muy prometedora, y el presente *TFM* representa un ejemplo de su implementación. En concreto, se exploran tres capacidades clave de los modelos:

1. **Clasificación de intenciones** (Sección 2.3): Distinguir entre tipos de preguntas (sobre historia del museo, obras concretas, o no permitir su respuesta).
2. **Traducción de lenguaje natural a SQL** (Sección 2.5): Cuando la pregunta se refiere a datos de obras concretas del museo (ej. “¿Qué cuadros pintó Sorolla en 1910?”), el sistema traduce la consulta a *SQL*, la ejecuta sobre una base de datos relacional y genera su respuesta en base a los resultados.
3. **RAG** (Sección 2.4): Si la pregunta requiere contexto narrativo o información del museo, se activa un módulo que genera texto a partir de una base de conocimiento textual.

Este enfoque modular permite explotar al máximo las capacidades de los *LLMs* sin depender de su conocimiento interno, mitigando errores y alucinaciones, y ofreciendo una interacción más informativa, personalizada y fiable.

2.3. Clasificación de preguntas con *LLMs*

Uno de los principales retos a la hora de diseñar un sistema conversacional sobre un museo es interpretar correctamente la intención del usuario. No todas las preguntas requieren el mismo tratamiento: algunas pueden resolverse mediante consultas estructuradas sobre bases de datos (por ejemplo, “¿Cuántas obras pintó Sorolla en 1910?”), mientras que otras necesitan una respuesta contextual extraída de documentos no estructurados (“¿Qué servicios ofrece el Museo Sorolla?”). Por ello, es necesario introducir una capa intermedia que clasifique las preguntas antes de ser procesadas.

El objetivo de la clasificación de preguntas es asignar cada consulta de usuario a una categoría predefinida que determine el tipo de respuesta que se dará. Este paso resulta crítico, ya que evita errores y optimiza el rendimiento del sistema, dirigiendo cada consulta al flujo más adecuado.

Es muy relevante mencionar la importancia que tienen los guardarrailes o *guardrails* en los *LLMs*. Los guardarrailes son mecanismos esenciales para garantizar que las respuestas generadas sean seguras, relevantes y éticamente adecuadas. Estos *guardrails* contienen filtros o reglas que bloquean contenidos inapropiados mediante el diseño de instrucciones y estrategias dentro del modelo para evitar respuestas problemáticas (Dong et al., 2024). La implementación de guardarrailes es especialmente importante en cualquier tipo de aplicación, pero especialmente en sistemas conversacionales en museos, donde es crucial ser riguroso para preservar la calidad y adecuación de la información ofrecida al usuario. Encontramos modelos como *Llama Safeguard* (Inan et al., 2023) o *NeMo Guardrails* (Rebedea et al., 2023).

Métodos tradicionales

Tradicionalmente, la clasificación de intenciones se ha abordado mediante técnicas de aprendizaje automático supervisado (Li et al., 2021), mediante el entrenamiento de modelos clasificadores sobre conjuntos de ejemplos etiquetados, usando por ejemplo *tf-idf* (Das and Chakraborty, 2018). Sin embargo, esto requiere la creación de un gran conjunto de datos etiquetado, lo cual no siempre es factible. Otras técnicas utilizadas están basadas en *Deep Learning* como lo son las redes neuronales *LSTMs* o las *RNNs* (Wang et al., 2024c).

El modelo `BART-large-MNLI`¹⁶ es una variante del modelo *BART* preentrenado específicamente para tareas de *natural language inference (NLI)*. Se utiliza comúnmente en tareas de clasificación de texto en *zero-shot*, donde se compara una entrada del usuario con múltiples etiquetas candidatas. Calcula la probabilidad de que la consulta pertenezca a cada una de las correspondientes etiquetas, permitiendo así clasificar sin entrenamiento adicional. Este tipo de modelos presenta la limitación de ser poco configurables, ya que no permiten ser alimentados con unas instrucciones previas para poder añadir matices. Esta limitación es importante, especialmente cuando la clasificación, como se verá en el presente trabajo, no es tan directa y necesita entender la intención real y los matices detrás de la pregunta del usuario. Esto es algo que los LLMs sí nos permiten afinar con instrucciones previas para entender qué significa cada clase.

Uso de LLMs para clasificación semántica

Los LLMs permiten realizar clasificación semántica sin entrenamiento específico, mediante enfoques *zero-shot* (Wang et al., 2023) o *few-shot*. En el primero, el modelo recibe una instrucción clara con la lista de posibles categorías. En el segundo, se le muestran además ejemplos por cada clase. En ambos casos, el modelo generaliza su comprensión del lenguaje para inferir la clase más probable de la consulta dada.

Ejemplo de prompt zero-shot:

“Clasifica la siguiente pregunta en una de estas categorías según el tipo de respuesta que necesita:

(1) Base de datos SQL, (2) Textos, (3) Fuera del dominio.

Pregunta: ¿Quién hizo la pintura con inventario 000123?”

En la práctica, se ha comprobado que modelos como `Llama-3-8B` o `Qwen-7B` son capaces de realizar esta tarea con gran precisión y sin datos de entrenamiento, superando con creces a los métodos tradicionales o basados en *deep learning* (Wang et al., 2024c).

Frente a los sistemas clásicos de clasificación, los LLMs presentan ventajas claras:

- No requieren *corpus* etiquetado.
- Pueden funcionar en distintos idiomas.
- Entienden el significado de las preguntas.

Sin embargo, también presentan desafíos, como la introducción de una mayor latencia al sistema o el coste asociado a su uso en producción. La utilización de APIs

¹⁶<https://huggingface.co/facebook/bartlarge-mnli>

como *Groq* con inferencia en la nube o la utilización de *GPUs* reduce la latencia, pero incrementa el coste.

La clasificación de intenciones mediante *LLMs* es clave en un sistema conversacional y será el primer módulo de la aplicación. La clasificación semántica con *LLMs* no solo mejora la precisión del sistema, sino que actúa como un filtro inicial fundamental para garantizar respuestas útiles, seguras y coherentes.

2.4. Retrieval-augmented generation (RAG)

En los últimos años, con el auge de los *LLMs*, ha surgido la necesidad de que estos no solo generen texto basándose en su conocimiento preentrenado, sino que también puedan acceder a información externa y actualizada de alguna temática concreta. Aquí es donde entran en juego las técnicas de *RAG*, que combinan las capacidades de generación de texto con la recuperación de información. Se elimina así la necesidad de realizar *fine-tuning* del *LLM*, evitando las limitaciones de creación de *dataset* o computacionales asociadas.

El concepto fundamental detrás de *RAG* es mejorar la capacidad de los modelos para producir respuestas precisas al consultar una base de datos o un *corpus* de documentos relevantes en tiempo real. En lugar de depender únicamente de la información “memorizada” durante su entrenamiento, un sistema *RAG* primero *recupera* los documentos relevantes de una fuente de conocimiento externa y luego *aumenta* la entrada del modelo con esa información recuperada antes de *generar* una respuesta (Lewis et al., 2020).

RAG vs fine-tuning

RAG a menudo se compara con el ajuste (*fine-tuning*) y la ingeniería de *prompts*. La ingeniería de *prompts* requiere mínima adaptación del modelo y conocimiento externo, aprovechando las capacidades inherentes de los *LLMs*. Como mencionan Gao et al. (2024), *RAG* podría entenderse como proporcionar un libro de texto a un modelo; ideal para tareas de recuperación de información precisa y para entornos dinámicos con actualizaciones de conocimiento en tiempo real. Ofrece alta interpretabilidad pero puede tener mayor latencia y consideraciones éticas sobre la recuperación de datos.

Fine-tuning es comparable a un estudiante que estudia un libro. Es adecuado para replicar estructuras o estilos específicos. Es más estático y requiere reentrenamiento para actualizaciones, demandando recursos computacionales.

RAG y *fine-tuning* no son mutuamente excluyentes y pueden complementarse para optimizar el rendimiento.

Componentes clave de un sistema RAG

Según Gao et al. (2024), un sistema *RAG* básico consta de tres componentes interconectados que trabajan en conjunto (véase la Figura 2.1):

1. **Indexing:** Se comienza extrayendo y limpiando datos en crudo en diferentes formatos como *PDF*, *HTML*, *Word* o *Markdown*. Ese contenido es convertido a un formato de texto plano. Para abordar el problema de limitación de contexto

2.4. Retrieval-augmented generation (RAG)

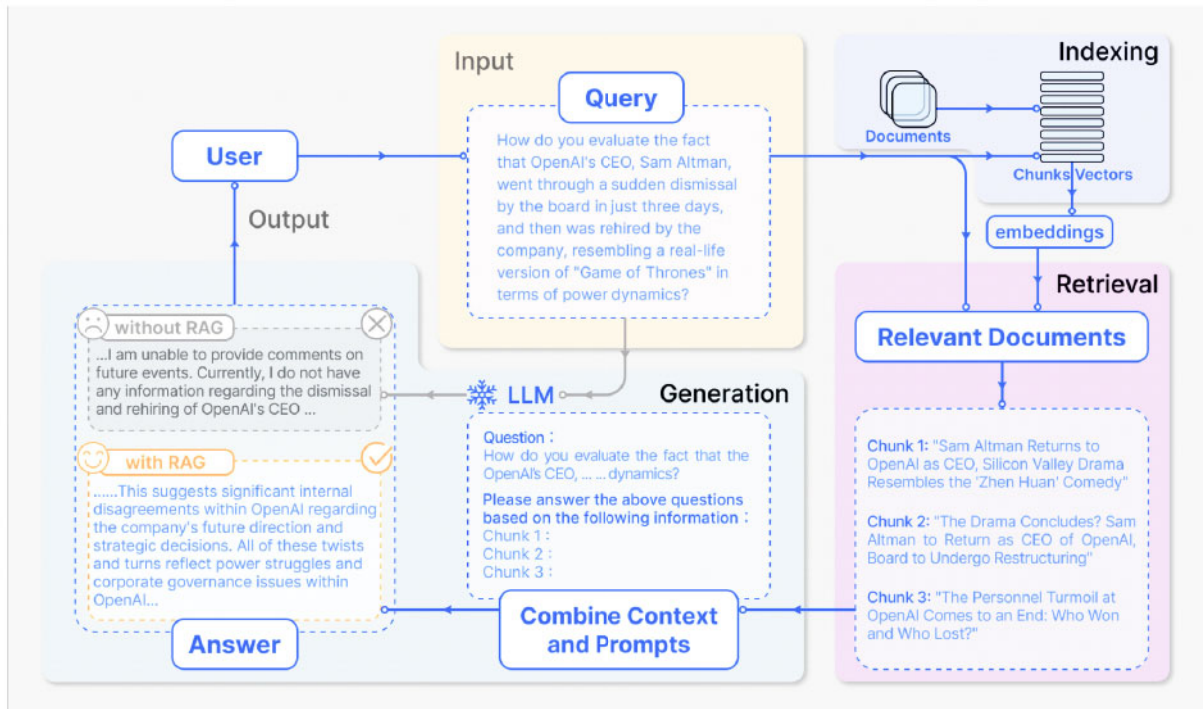


Figura 2.1: Ejemplo del funcionamiento de RAG (Fuente: Gao et al. (2024))

de entrada en los *LLMs*, los textos se dividen en fragmentos más pequeños, llamados “*chunks*”. La estrategia de *chunking* es crucial, ya que un tamaño inadecuado puede resultar en la pérdida de contexto esencial o en la inclusión de información irrelevante. Estos *chunks* son codificados usando un modelo de *embeddings*, es decir, codificados en un vector de alta dimensionalidad, y almacenados en una base de datos vectorial. La calidad y la actualidad de esta base determinará la calidad de las respuestas generadas por el sistema RAG. Dentro de este componente, se encuentran algunos conceptos clave:

- **Estrategias de chunking:** La efectividad del RAG depende en gran medida de cómo se segmenta la información. Algunas estrategias comunes mencionadas incluyen **chunks de tamaño fijo** (dividir el texto en segmentos de un número predefinido de caracteres o *tokens*, con alguna superposición para preservar el contexto entre segmentos adyacentes) o **chunking semántico** (dividir el texto basándose en su estructura lógica, por ejemplo, párrafos, secciones, títulos, obteniendo fragmentos coherentes). También se encuentra el **chunking recursivo**, un enfoque que prueba diferentes tamaños de *chunk* y estrategias de división hasta que los *chunks* son lo suficientemente pequeños para ser procesados, como la división en capítulos, secciones y párrafos.
- **Modelos de embeddings:** Son redes neuronales entrenadas para transformar texto en representaciones numéricas densas (*embeddings*). La particularidad de estos vectores es que textos con significados similares se representan como vectores cercanos en el espacio de alta dimensionalidad. La elección del modelo de *embeddings* es crítica, ya que su calidad determina la capacidad del sistema para encontrar los “*chunks*” más re-

levantes. Entre ellos, se encuentran algunos ejemplos de modelos como `all-miniLM-L6-v2` de *Sentence Transformers*¹⁷, `multilingual-e5-large` de *Intfloat* (Wang et al., 2024a), `text-embedding-3-large` de *OpenAI*¹⁸ o *Amazon* con su modelo `titan-embed-text-v2`¹⁹.

- **Bases de datos vectoriales:** Una vez se generan los *embeddings* de los *chunks* de la base de conocimiento, se almacenan en una base de datos vectorial. Este tipo de base de datos permite la búsqueda por similitud (por ejemplo, similitud coseno) y la recuperación de los *chunks* relevantes de una manera eficiente. Entre muchas opciones, se encuentra *FAISS* (Douze et al., 2025), *ChromaDB*²⁰, o las propias bases de datos relacionales como *PostgreSQL* de tipo `pgvector`²¹.
2. **Retrieval:** Ante una consulta del usuario, se genera el *embedding* de la consulta usando el mismo modelo de *embeddings* con el que se ha generado el *index*. Este componente es responsable de buscar y extraer documentos relevantes de la base de conocimiento. Utiliza técnicas de búsqueda por similitud con indexación vectorial para encontrar la información más relevante para la consulta del usuario. La precisión del *retriever* es crucial, ya que una recuperación deficiente resultará en respuestas de baja calidad, llamado el fenómeno *garbage in garbage out*, como comentan Gurawa and Dharmik (2025).
 3. **Generation:** Una vez que el *retriever* ha identificado los documentos relevantes, el generador los incorpora al *prompt* como contexto que se le proporciona al *LLM*. Esta información adicional “aumenta” el conocimiento del modelo, permitiéndole generar una respuesta más precisa, detallada y contextualizada sobre el tema tratado.

Otros sistemas *RAG* incluyen módulos adicionales para mitigar las limitaciones. Por ejemplo, *modular RAG* introduce componentes adicionales antes y después de la recuperación, como la optimización del índice, el *re-ranking* de resultados o la compresión del contexto, permitiendo mayor control y eficiencia en cada etapa del proceso (Gao et al., 2024). También hay otros enfoques que incluyen técnicas de evaluación del contexto obtenido y la posibilidad de solicitar más información al *retriever*²².

Ventajas

La implementación de *RAG* ofrece múltiples beneficios significativos:

- **Acceso a información actualizada:** Los modelos preentrenados tienen un conocimiento limitado al momento de su entrenamiento. *RAG* permite incorporar información reciente y dinámica, lo que es vital para aplicaciones que requieren datos en tiempo real o que cambian frecuentemente.
- **Reducción de alucinaciones:** Al basar las respuestas en información recuperada de fuentes fiables, los sistemas *RAG* minimizan la tendencia de los modelos a

¹⁷<https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>

¹⁸<https://platform.openai.com/docs/models/text-embedding-3-large>

¹⁹<https://docs.aws.amazon.com/bedrock/latest/userguide/titan-embedding-models.html>

²⁰<https://docs.trychroma.com/>

²¹<https://github.com/pgvector/pgvector>

²²<https://www.pinecone.io/learn/advanced-rag-techniques/>

“inventar” datos o generar información incorrecta. Esto mejora la fiabilidad y veracidad de las salidas.

- **Transparencia y explicabilidad:** Dado que las respuestas se basan en documentos específicos recuperados, es posible trazar la fuente de la información utilizada, lo que facilita la verificación y añade un nivel de explicabilidad al proceso de generación, muy importante en el campo de la IA (Luo and Specia, 2024).
- **Manejo de conocimiento específico del dominio:** RAG es muy útil para aplicaciones en dominios muy específicos, donde el conocimiento necesario no está representado en los datos de entrenamiento. Permite a los sistemas acceder a bases de conocimiento especializadas sin la necesidad de *fine-tuning*.
- **Eficiencia computacional:** En lugar de reentrenar un modelo de lenguaje completo con nuevos datos, lo cual es costoso y computacionalmente intensivo, RAG permite actualizar el conocimiento del sistema simplemente actualizando la base de datos de documentos del recuperador.

Limitaciones

La utilización de sistemas RAG también presenta limitaciones, como muestran Gupta et al. (2024):

- **Escalabilidad y eficiencia:** La escalabilidad de RAG es una de las mayores limitaciones. La utilización del *retriever* en grandes bases de conocimiento introduce problemas de latencia y computación en sistemas reales con restricciones de capacidad.
- **Calidad y relevancia del *retriever*:** Como se ha comentado previamente, la precisión del *retriever* es crucial en el sistema. Recuperar *chunks* irrelevantes, dejando información importante fuera del contexto para la generación, es un problema que ocurre frecuentemente (Gupta et al., 2024).
- **Sesgo y coherencia:** Respecto a la fase de generación, se encuentran problemas relacionados con la alucinación o coherencia en la generación de la respuesta, además de una posible irrelevancia o sesgo en la respuesta. Este sesgo puede ser introducido igualmente por la base de conocimiento y debe ser tenido en cuenta a la hora de desarrollar un sistema real.

Evolución y aplicaciones

Desde su conceptualización, las técnicas RAG han evolucionado rápidamente, integrándose en diversas arquitecturas y aplicaciones. Inicialmente propuestas para mejorar la generación de texto en tareas de respuesta a preguntas abiertas, su aplicabilidad se ha expandido a la generación de contenido, resúmenes, *chatbots* conversacionales y sistemas de búsqueda específicos de empresa. Además, se encuentran diferentes tipos de RAG basados en fuentes de conocimiento de texto, en audio, vídeo o multimodales (Abootorabi et al., 2025), lo que los hace muy útiles en distintos casos de uso.

La capacidad de combinar el razonamiento del modelo de lenguaje con el acceso a un repositorio de información dinámico la convierte en una técnica fundamental en la ingeniería de aplicaciones de IA de próxima generación, consolidándose como un

pilar esencial para el desarrollo de sistemas de información inteligentes capaces de operar con conocimiento actualizado y contextualizado.

2.5. Traducción de lenguaje natural a SQL

La traducción de lenguaje natural a SQL (*Text2SQL*) es una tarea fundamental en el *NLP* que busca permitir a los usuarios interactuar con bases de datos relacionales utilizando consultas en lenguaje humano, en lugar de requerir conocimientos técnicos de SQL o de intermediarios. Estos sistemas han revelado un gran rendimiento en esta tarea, como demuestran Shi et al. (2024).

Ejemplos de aplicación incluyen sistemas de *business intelligence*, atención al cliente automatizada, interfaces de datos para científicos no informáticos y, como es el caso de este trabajo, la consulta de información estructurada en museos.

Técnicas tradicionales

Los sistemas *Text2SQL* han evolucionado significativamente a lo largo del tiempo. Según Zhu et al. (2024), inicialmente, se basaban en enfoques heurísticos, reglas predefinidas y gramáticas específicas, siendo funcionales para dominios muy limitados, pero inflexibles y poco escalables ante la complejidad de las consultas o la diversidad de las bases de datos.

Con el auge del *deep learning*, la tarea se abordó con modelos de secuencia a secuencia (*Seq2Seq*), como aquellos que incorporaban redes *LSTM*. Estos mejoraron el rendimiento considerablemente. Sin embargo, seguían enfrentándose a desafíos significativos relacionados con la ambigüedad del lenguaje natural, la complejidad del SQL y la vinculación precisa con el esquema de la base de datos, lo que limitaba su generalización.

Más recientemente, la irrupción de las arquitecturas *transformer* y los *LLMs* ha permitido tratar consultas cada vez más complejas y ambiguas, acercándose a la capacidad humana para formular preguntas en lenguaje cotidiano.

Técnicas actuales

Las aproximaciones modernas que se utilizan para la traducción de lenguaje natural a SQL pueden distinguirse, según la clasificación de Zhu et al. (2024), en varias categorías principales basadas en sus estrategias de entrenamiento:

- **Prompt engineering (sin entrenamiento):** Esta técnica usa *prompts* detallados para guiar a los *LLMs* para generar SQL más preciso, desde un enfoque *zero-shot* (sin ejemplos previos) o *few-shot* (con pocos ejemplos previos). Es la opción más simple en cuanto a computación y será usada en el presente trabajo.
- **Fine-tuning:** Reentrenar el modelo *LLM* para adaptar su rendimiento en tareas de *Text2SQL*.
- **Task-training:** Modelos *LLMs* entrenados desde 0 para la tarea de *Text2SQL* utilizando la arquitectura *transformer*.
- **Agente LLM:** Colaboración de múltiples inteligencias que generan dinámicamente *queries SQL* y las evalúan.

2.6. Web scraping como acceso a fuentes de conocimiento

Desafíos en el dominio museístico y en español

Aunque existen datasets públicos para *Text2SQL* en inglés tales como *Spider* (Yu et al., 2019), la disponibilidad de recursos para el español es limitada, lo que complica el desarrollo de sistemas específicos para museos españoles. Además, el lenguaje natural en español puede presentar variaciones regionales y ambigüedades que dificultan la correcta generación de consultas. Existen *LLMs* reentrenados en consultas *SQL* en castellano, como el modelo `LLM_SQL_BaseDatosEspanol_Mistral` presentado por Somos NLP²³, basado en el modelo *Mistral fine-tuned* para consultas *SQL* en español. Sin embargo, se encontraron dificultades técnicas para implementar este modelo en el caso de estudio.

Por otro lado, las preguntas de los usuarios suelen ser heterogéneas y a veces imprecisas, lo que obliga al sistema a interpretar correctamente la intención y a manejar errores o consultas fuera de dominio.

Aplicación en el TFM

En este proyecto se ha integrado un modelo `LLaMA 3` con una capa de traducción *Text2SQL*, orientada a consultas sobre la base de datos del Museo Sorolla. El modelo recibe la pregunta del usuario y genera una consulta *SQL* que es ejecutada directamente sobre la base de datos *PostgreSQL*. Los resultados se presentan al usuario de manera conversacional, mostrando los objetos en una forma intuitiva.

2.6. Web scraping como acceso a fuentes de conocimiento

El *web scraping* es una técnica utilizada para la extracción automatizada de datos de sitios web usando *bots* o *scripts*. Esta técnica es un método fundamental para la adquisición de grandes volúmenes de información no estructurada o semi-estructurada que, de otro modo, sería inaccesible para su procesamiento automatizado.

En el contexto de la consulta de información cultural, la disponibilidad de datos estructurados no siempre es inmediata. Por ello, el *web scraping* se convierte en una herramienta fundamental para obtener datos específicos desde sitios *web*, portales institucionales o bases de datos públicas, facilitando así la construcción de *corpus* relevantes para sistemas de *IA*.

En sistemas basados en *LLMs*, el *web scraping* se convierte en una vía esencial para obtener la base de conocimiento. Al recopilar datos directamente de fuentes *web* relevantes, se puede alimentar al *LLM* con información específica del dominio.

Metodologías y Herramientas

La ejecución del *web scraping* varía en complejidad según la naturaleza del sitio *web*. Se utilizan comúnmente librerías de *parsing* como *Beautiful Soup*²⁴ en *Python*, que permiten navegar y extraer elementos específicos de la página.

Sin embargo, muchos sitios web modernos generan su contenido de forma dinámica mediante *JavaScript*. Para estos casos, se requieren herramientas más avanzadas

²³https://huggingface.co/somosnlp/LLM_SQL_BaseDatosEspanol_Mistral

²⁴<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

como *Selenium*²⁵, que simulan la interacción de un navegador *web* real. *Frameworks* como *Scrapy*²⁶ ofrecen una solución más robusta y escalable para proyectos de *scraping* complejos.

Desafíos y consideraciones

El *web scraping* presenta varios desafíos técnicos, éticos y legales que deben ser cuidadosamente abordados:

- **Desafíos técnicos:** Las medidas *anti-scraping*, como *CAPTCHAs*, bloqueos de *IP* o la detección de *bots*, requieren el uso de técnicas avanzadas (como *proxies*, cambiar la identidad del navegador a través de la rotación de *User-Agents*²⁷ o simulación de comportamiento humano), así como el manejo de contenido generado por *JavaScript*.
- **Consideraciones legales y éticas:** La legalidad del *web scraping* varía según la jurisdicción y el tipo de datos. El archivo `robots.txt` del sitio web especifica qué partes del sitio no deben ser rastreadas, donde a veces se prohíbe el *scraping*. Además, se requiere un respeto de los derechos de autor sobre el contenido extraído y la privacidad de los datos, especialmente si se trata de información personal.

Aplicación en el caso de uso

El *web scraping* es una fase crítica para la construcción de la base de conocimiento del sistema conversacional en el caso de estudio. En el presente trabajo, se ha utilizado esta técnica para extraer información relevante de una página *web* con información sobre museos, como se presenta en la Sección 3.1.

²⁵<https://www.selenium.dev/documentation/>

²⁶<https://scrapy.org/>

²⁷<https://medium.com/@datajournal/user-agents-for-web-scraping-d0407b70a42d>

Capítulo 3

Desarrollo

Este capítulo detalla las fases de desarrollo del proyecto *SoroIA*, abarcando desde la adquisición de datos hasta la implementación completa del sistema del asistente. El objetivo principal ha sido construir una aplicación capaz de responder a preguntas formuladas en lenguaje natural por parte de los visitantes del Museo Sorolla, utilizando técnicas de *LLMs*, *Text2SQL* y *RAG*.

Para ello, se han llevado a cabo distintas etapas: en primer lugar, se realizó el *scraping* de datos culturales desde plataformas como *CER.es* y la página oficial del Museo Sorolla. Posteriormente, se procesaron y limpiaron los datos para su integración en una base de datos estructurada. El sistema de *backend* combina modelos *LLM* capaces de traducir preguntas a *SQL*, así como módulos de *RAG* que permiten responder a consultas más generales sobre el museo o sobre la vida de Joaquín Sorolla. Además, se desarrolló una interfaz visual e interactiva con *Streamlit*, orientada a ofrecer una experiencia clara y accesible para todo tipo de usuarios. Finalmente, se expone el diseño de *SoroIA* en una arquitectura en la nube. Los siguientes capítulos muestran detalladamente cada uno de estos pasos.

El código asociado a todo el desarrollo de *SoroIA* se encuentra disponible en un repositorio de GitHub¹.

3.1. Scraping

3.1.1. Elección de una fuente de conocimiento

El primer paso del proyecto consistió en hacer *scraping* de la página de las *Colecciones En Red Españolas (CER.es)*² del Ministerio de Cultura. *CER.es* es un catálogo colectivo en línea que reúne información e imágenes de una importante selección de los bienes culturales de los museos, centros e instituciones integrantes de la *Red Digital de Colecciones de Museos de España*. Esta red reúne instituciones museísticas de distintas especialidades, ámbitos temáticos y geográficos, y de diferentes titularidades, públicas y privadas, con el objetivo de hacer accesibles en línea contenidos digitales sobre sus bienes culturales y crear un espacio de difusión del conocimiento sobre los mismos. La enorme cantidad de información que se halla alojada en este

¹<https://github.com/fedecasl0/tfm-soroia-ceres>

²<https://ceres.mcu.es/pages/SimpleSearch?index=true>

repositorio será usada como una fuente de conocimiento para el sistema de consultas con SQL (Sección 3.2.2.1).

En un primer momento, el proyecto trató de hacer *web scraping* de todos los documentos que se encuentran en la web (un total de alrededor de 684.000 imágenes y 383.000 bienes culturales). Sin embargo, surgieron diferentes limitaciones. La página no cuenta con una herramienta de búsqueda que permita abarcar todos los bienes, siempre se debe realizar una búsqueda por términos (por ejemplo, “pintura”, “Valencia” o “1902”). Otro problema que surgió es que los resultados de una búsqueda están limitados a 21.000 objetos, por lo que también era difícil hacer búsquedas generales que tuvieran menos de ese número de objetos. Otra limitación era que no todas las páginas de resultados de búsqueda tenían el mismo formato, por lo que había que estudiar la estructura *HTML* de cada caso, lo que sería costoso de realizar para tantas búsquedas.

Por ello, se optó por realizar una descarga de los objetos de un museo concreto. Se buscaron diferentes museos que pudieran ser interesantes como caso de uso. Entre los museos seleccionados, algunos volvían a mostrar problemas, como el Museo Antropológico Nacional, que tenía más de 21.000 objetos, por lo que su obtención no era sencilla.

Entre otros museos que se seleccionaron, se optó finalmente por hacer la descarga de los bienes del Museo Sorolla. Cumplía los requisitos: alrededor de 20.000 objetos. Sin embargo, la búsqueda general del término “Museo Sorolla” devolvía cualquier objeto de la colección que mencionase al museo, sin tener que pertenecer a este. Esto hacía que se incluyeran objetos irrelevantes, requiriendo un filtrado posterior. Sin embargo, la existencia de un acceso directo desde la página web del museo a cada una de las colecciones del museo³ en *CER.es*⁴ facilitó el trabajo. Esto hace que la descarga de cada colección esté encapsulada, asegurando que cada colección se descarga completamente. Como se mencionó antes, la estructura *HTML* de los resultados varía, por lo que se desarrollaron tres códigos diferentes para poder hacer *scraping* de todo el museo (véase la Sección 3.1.2).

Finalmente, la descarga final del Museo Sorolla contó con las siguientes colecciones con un total de 19.100 objetos: *Fotografía* (6597 objetos), *Cartas* (5226 objetos), *Dibujo* (4987 objetos), *Pintura* (1358 objetos), *Joyería* (256 objetos), *Escultura* (229 objetos), *Mobiliario* (183 objetos), *Cerámica* (129 objetos) y *Textiles* (64 objetos).

3.1.2. Desarrollo de código de *scraping*

Una vez seleccionadas las colecciones del Museo Sorolla, fue necesario desarrollar código de *web scraping* para automatizar la descarga de los datos. Para ello, se implementaron varios *scripts* en Python, donde cada colección podía presentar variaciones tanto en el *HTML* como en la forma de acceder a los datos.

El proceso de *scraping* se dividió en dos fases: la recolección de la ficha completa de cada objeto y la descarga de sus imágenes asociadas.

Obtención de fichas

³<https://www.cultura.gob.es/msorolla/colecciones/colecciones-del-museo.html>

⁴<https://ceres.mcu.es/pages/SimpleSearch?Museo=MSM>

El *script* está basado en una función que recorre todas las páginas de resultados de una colección concreta. Para cada página, se localizan los elementos *HTML* que permiten acceder a la ficha completa de cada objeto. Como estos enlaces no son directos, fue necesario simular una solicitud *POST* utilizando los valores ocultos (*hidden values*) presentes en el *HTML*. Esta técnica permite obtener el contenido completo de la ficha en una nueva vista, incluyendo campos como título, autor, cronología, técnica o dimensiones, entre otros campos.

Una vez recuperada la ficha, se utiliza la librería *BeautifulSoup*⁵ para parsear el contenido y convertirlo en un diccionario clave-valor. El número de inventario se utiliza como identificador único de cada objeto. Los objetos son almacenados en un diccionario general, y se van guardando en formato *JSON* (Figura 3.1) a medida que avanza la recolección.

```
"00839": {
  "Museo": "Museo Sorolla",
  "Inventario": "00839",
  "Clasificación Genérica": "Pintura",
  "Objeto/Documento": "Cuadro",
  "Componentes": "Cuadro(1);Marco(1)",
  "Autor/a": "Sorolla Bastida, Joaquín",
  "Título": "El baño del caballo",
  "Materia/Soporte": "Cuadro:LienzoMarco:Madera de pino",
  "Técnica": "Cuadro:ÓleoMarco:Dorado al agua[Dorado con pan de oro falso bruñado.]",
  "Dimensiones": "Altura = 205 cm; Anchura = 250 cmMarco: Anchura = 293 cm; Grosor = 15 cm; Altura = 237 cm",
  "Descripción": "Un adolescente, desnudo, de perfil y cubierta su cabeza con amplio sombrero de paja, saca del mar un cab",
  "Iconografía": "Baño;Playa;Caballo",
  "Firmas/Marcas/Etiquetas": "Parte inferior izquierdaJ. Sorolla y Bastida / 1909 [En azul oscuro]",
  "Datación": "Cuadro:1909 (1909) [Tomada de la firma de la pieza. Se pinta en verano.]Marco:2001=2025",
  "Contexto Cultural/Estilo": "Reinado de Alfonso XIII (1902-1931)",
  "Lugar de Producción/Ceca": "Valencia",
  "Historia del Objeto": "Exposiciones:2009, Madrid, \"Joaquín Sorolla 1863-1923\", Museo Nacional del Prado, 26/05/2009-1",
  "Clasificación Razonada": "Lienzo comprendido en la serie de escenas de la playa de El Cabañal de Valencia que realiza",
  "Bibliografía": "DOMENECH, Rafael.Sorolla. Sa vie et son oeuvre.1910. il. 109.ECHEVARRÍA, Rosa.Los genios de la pintura.",
  "Forma de Ingreso": "Legado fundacional",
  "Imágenes": [
    "imagenes/00839/00839_1.jpg"
  ]
},
```

Figura 3.1: Ejemplo del formato *JSON* para las fichas de los objetos descargados del Museo Sorolla.

Descarga de imágenes

Esta tarea presentaba dos retos principales: por un lado, la localización del enlace que dirige al visor de imágenes de alta calidad (habitualmente, bajo un botón llamado “Ampliar Imagen”), y, por otro lado, la descarga secuencial de imágenes cuando existen múltiples vistas del mismo objeto.

Para evitar imágenes duplicadas, se implementó una verificación mediante el *hash* de los primeros *bytes* de cada imagen. Si una imagen ya se había descargado previamente, el ciclo se detenía. Esto permitió optimizar el número de peticiones al servidor y evitar duplicados.

Las imágenes se almacenan en carpetas estructuradas por inventario, siguiendo el esquema *imagenes/<ID>/<ID>_1.jpg, <ID>_2.jpg, etc.*

La conexión con *CER.es* se cerraba en algunas ocasiones, por lo que se introdujeron

⁵<https://beautiful-soup-4.readthedocs.io/en/latest/>

esperas aleatorias entre peticiones (entre 1 y 4 segundos) para respetar los tiempos de respuesta del servidor y evitar ser bloqueados por actividad sospechosa⁶. De esta manera, se descargaron todas las imágenes correctamente.

Variaciones entre colecciones

Dado que las estructuras *HTML* no eran completamente uniformes entre colecciones, se implementaron versiones específicas del código para adaptarse a las diferencias detectadas. Por ejemplo, en la colección de pintura era necesario tratar algunos caracteres especiales en los nombres de archivos, por lo que se incluyó una función para limpiar rutas de archivos. Asimismo, en otras colecciones, como cartas, se añadieron filtros para excluir ciertos objetos que no cumplían con el formato esperado (por ejemplo, inventarios que no comenzaban por “CS”).

En total, se desarrollaron tres versiones del *script* que comparten una misma estructura base, pero que se adaptan a cada colección concreta. Gracias a ello, se logró recolectar las fichas de más de 19.000 objetos distribuidos en nueve colecciones del museo, junto con sus imágenes asociadas, un total de 49.592.

3.1.3. Limpieza

Una vez los datos son obtenidos, la limpieza supuso una parte fundamental en el funcionamiento de la aplicación. Como mencionan Figuerola et al. (2007), la limpieza es crucial para recuperar la información más relevante. Las fichas obtenidas de *CER.es* contenían mayúsculas, tildes o caracteres no alfanuméricos, por lo que se tomó la decisión de eliminarlos. El resultado fue un *JSON* con texto alfanumérico en minúsculas. De esta manera, al buscar en la base de datos, se evitan los problemas por búsquedas con mayúsculas o tildes.

Además, el campo *Datación* contenía texto sin patrón concreto. En ocasiones, la fecha aparecía estimada, a veces el siglo o a veces como un número. Por ello, se aplicó una búsqueda regular de cuatro números en el campo, obteniendo de esta manera un nuevo campo con el año asignado al objeto, ayudando a hacer búsquedas por año de una manera más efectiva.

3.1.4. Ingesta en una base de datos

Una vez se han descargado los datos, se introducen en una base de datos relacional, usando como clave primaria el campo “*Inventario*”. En el caso de uso local, se optó por *PostgreSQL*, que ofrece gratuitamente el servicio. La tabla tiene los campos mostrados en la Figura 3.2, donde no son obligatorios todos los campos para cada instancia, ya que las fichas presentan irregularidades.

3.1.5. Información de la página web del museo

Otra fuente de información usada fue la propia página web del Museo Sorolla⁷. Dentro de esta página, se incorporó más información sobre la biografía del autor, así como mucha documentación de la historia del museo. Además, se encuentran descritas las salas del museo con detalle, dando al usuario un resumen de cada sala. También se

⁶<https://webscraping.ai/faq/homegate-scraping/what-is-the-ideal-time-delay-between-requests-to-avoid-throttling-when-scraping-homegate>

⁷<https://www.cultura.gob.es/msorolla/inicio.html>

Column	Type	Collation	Nullable	Default
inventario	text		not null	
coleccion	text			
nombre_especifico	text			
tipologia_estado	text			
lugar_de_produccion_ceca	text			
inscripciones_leyendas	text			
historia_del_objeto	text			
iconografia	text			
dimensiones	text			
contexto_cultural_estilo	text			
descripcion	text			
forma_de_ingreso	text			
clasificacion_razonada	text			
componentes	text			
tecnica	text			
clasificacion_generica	text			
autor_a	text			
datacion	text			
conjunto	text			
lugar_de_procedencia	text			
objeto_documento	text			
materia_soporte	text			
titulo	text			
imagenes	text			
firmas_marcas_etiquetas	text			
bibliografia	text			
fecha_ano	integer			

Indexes:
 "fichas_raw_pkey" PRIMARY KEY, btree (inventario)

Figura 3.2: Esquema de la base de datos de las fichas del Museo Sorolla.

usan textos acerca de la información del museo, como el acceso, tarifas, derechos, actividades adicionales u otros servicios que da el museo. Estos textos, junto a los obtenidos de *Wikipedia*, son los que alimentarán el módulo de RAG (Sección 3.2.2.2).

3.2. Desarrollo de *SoroIA*

En esta sección se detalla el proceso de diseño y desarrollo de *SoroIA*. Se abordará la implementación de la aplicación en un entorno local como fase inicial. La arquitectura de la aplicación sigue un flujo modular definido, como se ilustra en la Figura 3.3, que permite procesar las consultas de los usuarios desde su recepción hasta la generación de una respuesta final. A grandes rasgos, el proceso incluye los siguientes pasos:

1. El usuario accede a la interfaz y envía una consulta.
2. Un *LLM* (guardarraíles) clasifica la intención de la pregunta en una de las siguientes categorías, según el tipo de respuesta que se busca:
 - Vida de Sorolla o Museo Sorolla → *RAG* → Obtener textos más relevantes.
 - Objetos del museo → *LLM* para convertir texto a *SQL* → Recuperar los objetos más relevantes.
 - Preguntas de interacción: Se devuelve una respuesta amable generada por un *LLM*.

- Preguntas no relacionadas con el museo → Respuesta: *"Lo siento, tu consulta no parece estar relacionada con el museo."*
3. Se ejecuta la tarea correspondiente según la clasificación (*RAG*, *SQL*, interacción o respuesta de rechazo).
 4. Se genera la respuesta con un LLM, usando el contexto de la tarea asignada (*RAG* o *SQL*).
 5. Se presenta al usuario la respuesta final en la interfaz.

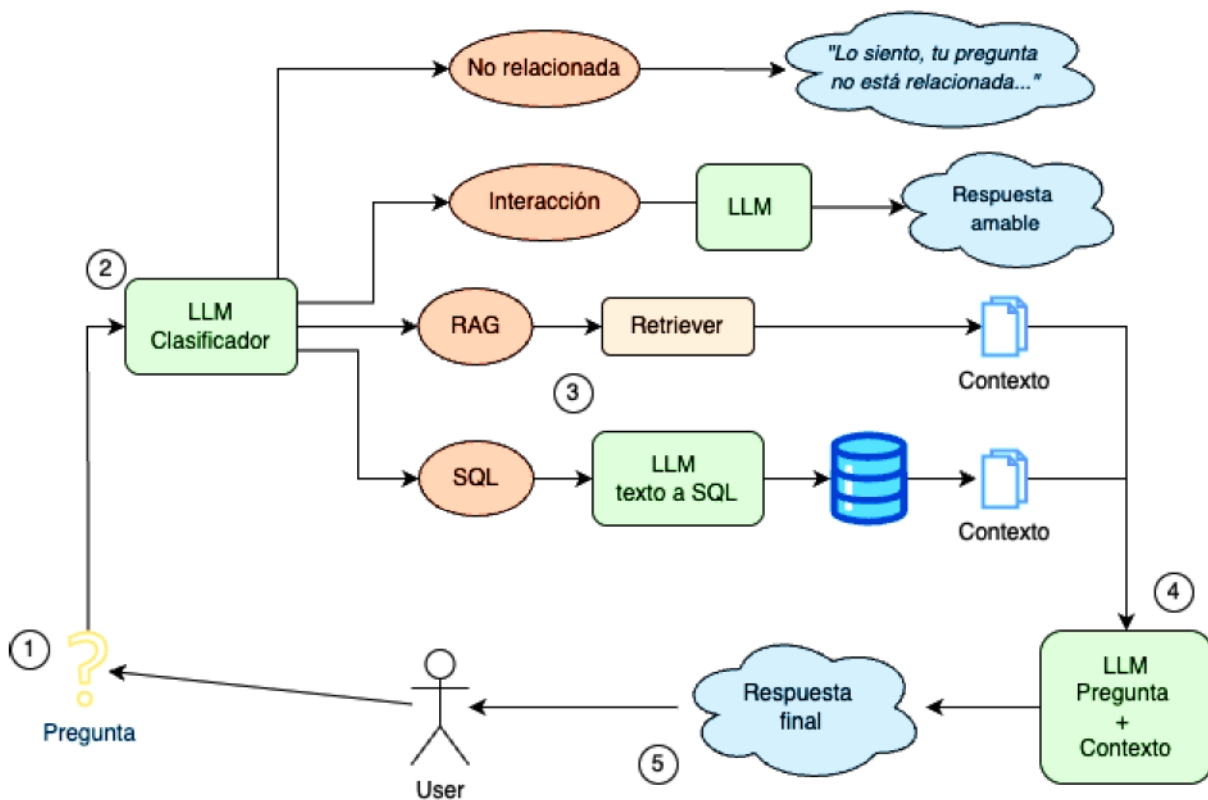


Figura 3.3: Flujo de ejecución de *SoroIA*.

3.2.1. Clasificador de la intención

El primer elemento del flujo de *SoroIA* es clasificar la intención del usuario. En el caso de uso actual, donde se responden preguntas a visitantes de un museo, es crucial limitar los temas sobre los que puede hablar el modelo, disminuyendo así la probabilidad de alucinaciones o fallos de seguridad.

Una vez que el usuario envía una pregunta, esta se clasifica en uno de los siguientes cuatro casos:

- **RAG:** Preguntas relacionadas con información general o histórica del Museo Sorolla, o sobre datos biográficos de Joaquín Sorolla (por ejemplo, *"Háblame de la infancia de Sorolla"* o *"¿Qué horario tiene el museo?"*). Estas preguntas basarán sus respuestas en textos fiables.

- **SQL:** Preguntas relacionadas con información específica sobre las fichas de objetos del museo (por ejemplo, “¿Cuántas cartas hay en el museo?” o “Háblame del objeto 00423”). Estas preguntas serán traducidas a una consulta *SQL* y los resultados usados para generar la respuesta final.
- **Interacción con el usuario:** Interacciones del usuario que no necesitan información del museo (saludos, despedida, agradecimientos, mensajes de amabilidad, etc.), simplemente una respuesta amable.
- **No relacionada / Riesgo de seguridad:** Preguntas que no están relacionadas con el caso de uso o que pueden implicar riesgos de seguridad (por ejemplo, “Háblame sobre el Imperio Romano.”, “¿Cuántos goles lleva Mbappé este año?”, o “Borra la base de datos.”). Esta etiqueta cubre la funcionalidad de los guardarrailes, evitando fallos de seguridad al acotar las respuestas al caso de uso (como se menciona en la Sección 2.3).

Para llevar a cabo esta tarea, inicialmente se utilizó el modelo de clasificación *zero-shot* llamado `facebook/bart-large-mnli`⁸, capaz de clasificar un *prompt* dada una lista de categorías. Sin embargo, este enfoque resultó algo limitado, ya que algunas preguntas eran ambiguas o difíciles de clasificar (por ejemplo, “¿Qué se puede ver en la sala 3?”, o “¿Qué se sabe sobre los retratos del museo?”).

Finalmente, se optó por utilizar el modelo *LLM* `llama-3.3-70b-versatile`⁹ para clasificar la intención del usuario por su mejora en rendimiento. La gran ventaja respecto al modelo anterior es la utilización de unas instrucciones previas para definir con mayor precisión cada clase. El modelo se ejecuta con una *temperatura* de 0 para maximizar la consistencia y reducir la aleatoriedad en la clasificación. El *prompt* que se le da como entrada es el siguiente:

Eres un experto asistente para visitantes del Museo Sorolla. Clasifica esta consulta como: -“SQL” si se refiere a datos concretos que puedan estar en una base de datos del Museo Sorolla (hay colecciones de mobiliario, cartas, escultura, textiles, pintura, fotografía, dibujo, joyería, cerámica), -“RAG” si busca información de: información del Museo Sorolla (sus salas, historia, información al público) o biografía de Joaquín Sorolla. - Si es un saludo, despedida o mensaje amable sin contenido informativo responde “INTERACCIÓN”. - Si la pregunta no está relacionada con el caso de uso o puede ser un problema de seguridad (crear o borrar base de datos, solicitar credenciales, contraseñas), responde “NO”.

Pregunta: {mensaje del usuario}

Respuesta (solo responde “SQL”, “RAG”, “INTERACCIÓN” o “NO”):”

Una vez clasificada la pregunta, el flujo puede tomar tres caminos: realizar una consulta *SQL* a la base de datos (véase Sección 3.2.2.1), hacer *RAG* sobre los textos del Museo Sorolla o Joaquín Sorolla (véase Sección 3.2.2.2), responder amablemente o responder con “Lo siento, la consulta no parece estar relacionada con el museo”.

⁸<https://huggingface.co/facebook/bartlarge-mnli>

⁹<https://console.groq.com/>

3.2.2. Tarea asignada

Una vez se ha clasificado la consulta del usuario, se realiza la acción correspondiente.

3.2.2.1. Traducción de la consulta a SQL

Cuando la pregunta está relacionada con objetos, se utiliza un *LLM* que es capaz de traducir la pregunta del usuario a una consulta *SQL*.

Se evaluó inicialmente el modelo *LLM_SQL_BaseDatosEspanol_Mistral* desarrollado por *SomosNLP*¹⁰ para la traducción de consultas en español a *SQL*. No obstante, debido a limitaciones del entorno, se decidió utilizar el modelo ya mencionado *llama-3.3-70b-versatile0b-versatile* desde *Groq API*, con unos resultados muy precisos. El parámetro *temperatura* fue fijado a 0, para evitar alucinaciones, buscando la respuesta más precisa.

El *prompt* introducido al *LLM* es:

Genera una consulta SQL para responder la siguiente pregunta del usuario, usando la tabla 'fichas_raw', que tiene las siguientes columnas: <lista de columnas como parámetro de la función>. Ten en cuenta lo siguiente:

- *La base de datos contiene texto en minúsculas y sin tildes. Usa 'ILIKE' con operador % para encontrar coincidencias aproximadas.*
- *Para búsquedas temáticas o de contenido, es mucho más probable que las palabras clave relevantes estén en las columnas 'descripcion', 'clasificacion_razonada' e 'historia_del_objeto', incluso si hay otras columnas como 'lugar_de_produccion_ceca' o 'tecnica' que parezcan relevantes pero no siempre están rellenas. Prioriza siempre estos campos largos para búsquedas por palabras clave.*
- *Si se menciona un número que no parece una fecha, probablemente se refiere al 'inventario'.*
- *A menos que el usuario especifique lo contrario, limita los resultados a 10 filas.*
- *Puedes usar 'ORDER BY' si tiene sentido para la consulta.*
- *No uses instrucciones como 'DELETE', 'UPDATE', 'INSERT', 'DROP' o 'CREATE'.*
- *Usa solo las columnas proporcionadas.*

Pregunta del usuario: {mensaje}

Devuelve únicamente el texto de la consulta SQL sin comentarios ni formato adicional ni "sql.

Se exploró también el uso del agente *SQL* proporcionado por la librería *LangChain* (Knight, 2023), que ofrece capacidades integradas para traducir consultas en lenguaje natural a *SQL* y ejecutar las mismas. Sin embargo, este agente mostró unos resultados inferiores en términos de precisión y adecuación a la estructura específica de la base de datos del museo, por lo que no está incluido en los resultados.

¹⁰https://huggingface.co/somosnlp/LLM_SQL_BaseDatosEspanol_Mistral

Una vez se ha traducido, se crea una conexión con la base de datos y se ejecuta la consulta, obteniendo de vuelta los documentos que cumplen con los filtros. Estos documentos servirán de contexto para que el *LLM* final responda con detalle a la consulta del usuario.

3.2.2.2. RAG

Para responder a preguntas relacionadas con aspectos históricos, biográficos o generales sobre Joaquín Sorolla o el Museo Sorolla, se emplea una técnica de *RAG*. Este enfoque permite que el modelo acceda a información externa relevante antes de generar una respuesta, reduciendo así el riesgo de alucinaciones y mejorando la precisión de las respuestas.

Preprocesado y segmentación de los documentos

Como fuente de conocimiento se han utilizado dos documentos en formato texto plano (`.txt`), extraídos de artículos de *Wikipedia* a fecha de marzo de 2025^{11,12} y de la propia página del Museo Sorolla.

Antes de poder utilizar los documentos como base para el sistema *RAG*, fue necesario aplicar un proceso de limpieza y segmentación textual. Este paso es fundamental para garantizar que la información se almacene en fragmentos coherentes, semánticamente útiles, y que puedan ser recuperados de forma eficaz.

Con el objetivo de facilitar la indexación semántica, los textos fueron procesados mediante un algoritmo de segmentación textual. Para ello, se utilizó el método de la librería de Langchain `RecursiveCharacterTextSplitter`¹³, que divide los documentos en fragmentos (*chunks*) de longitud fija (700 caracteres), con un solapamiento del 70 caracteres entre bloques consecutivos. Se experimentó con distintas longitudes de fragmento (200, 500 y 700 caracteres), el tamaño de 700 ofrecía una mejor precisión y cobertura semántica. El solapamiento se introdujo para evitar la pérdida de contexto en los bordes de los fragmentos, esto es especialmente útil cuando una frase o un párrafo relevante se encuentra en el límite entre dos bloques.

Generación de *embeddings* y almacenamiento

Una vez obtenidos estos fragmentos, se aplicó un modelo de generación de *embeddings* entrenado en español. En este caso, se utilizó `all-MiniLM-L12-v2`, un modelo de generación de *embeddings* desde texto, disponible en Hugging Face¹⁴. Esto permitirá representar tanto los fragmentos como las preguntas de los usuarios en el mismo espacio vectorial de 384 dimensiones.

Los vectores generados se almacenaron en una base de datos vectorial *FAISS* (Douze et al., 2025), optimizada para búsquedas por similitud utilizando la distancia coseno. Adicionalmente, se utilizó *ChromaDB*¹⁵ como alternativa para pruebas de persistencia y rendimiento. Aunque *ChromaDB* mostró una mayor velocidad de consulta y

¹¹https://es.wikipedia.org/wiki/Joaquin_Sorolla

¹²https://es.wikipedia.org/wiki/Museo_Sorolla

¹³https://python.langchain.com/api_reference/text_splitters/character/langchain_text_splitters.character.RecursiveCharacterTextSplitter.html

¹⁴<https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>

¹⁵<https://www.trychroma.com/>

soporte para almacenamiento persistente, FAISS ofreció mejores resultados en términos de relevancia de los fragmentos recuperados.

Codificación de la pregunta del usuario

Cuando el sistema clasifica una consulta como perteneciente al dominio de RAG, se activa el módulo correspondiente. En primer lugar, la pregunta del usuario se codifica en un vector utilizando el mismo modelo de *embeddings* empleado durante la fase de vectorización del texto. A continuación, se emplea un *retriever* proporcionado por *LangChain*, que realiza una búsqueda por similitud en el espacio vectorial utilizando la distancia coseno. Este proceso permite recuperar los k fragmentos más relevantes del corpus (habitualmente los 3 o 5 más similares).

Los fragmentos recuperados se incorporan al *prompt* final que se proporciona al LLM, de modo que este pueda generar una respuesta basada en el contexto proporcionado.

3.2.3. Generación de la respuesta final

Una vez se ha obtenido el contexto adecuado, ya sea mediante una consulta SQL o mediante RAG, el siguiente paso es construir una respuesta coherente y útil para el usuario. Esta tarea recae sobre un LLM que recibe un *prompt* con el contexto. Para el caso de SQL:

“Eres un asistente del Museo Sorolla. Tu tarea es responder a los visitantes basándote en la información del contexto.”

Consulta del usuario: <Consulta>, Query SQL generada: <SQL generado>, Contexto obtenido de la fuente de conocimiento: <Resultados>.

Si el contexto es un número o un dato breve, intégralo de forma natural en una explicación completa que responda adecuadamente a la consulta del usuario. Si hay rutas de imágenes en el contexto, di que se muestran las imágenes abajo (no menciones las rutas). No menciones la existencia del SQL. No hagas respuestas muy largas si la consulta no lo requiere.”

y para el caso de RAG

“Eres un asistente del Museo Sorolla. Responde a la siguiente consulta del usuario utilizando solo el contexto proporcionado. Si la respuesta no está en el contexto, responde “No he encontrado información sobre eso”. Adapta la longitud de la respuesta al tipo de pregunta.”

Consulta: <consulta>, Contexto: <contexto>, Respuesta:”

Para esta etapa, se utiliza el modelo llama-3.3-70b-versatile, proporcionado a través de la API de Groq. Esta elección se basa en su alta capacidad para entender contexto complejo y generar respuestas precisas en español. Además, se optó por separar este modelo del resto del *pipeline* para poder ajustar el tono, el estilo y la claridad del lenguaje sin interferir en las etapas previas de razonamiento. También se probó el modelo mistral-saba-24b¹⁶, con resultados inferiores.

Para obtener un equilibrio entre una respuesta precisa y cercana al usuario, se configuró una temperatura de 0,2 en el LLM. Mientras que un valor demasiado alto podría

¹⁶<https://mistral.ai/news/mistral-saba>

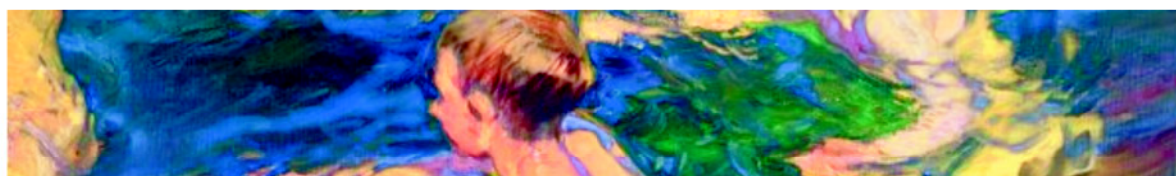
dar lugar a respuestas con mucha variabilidad y poco precisas, un valor de 0 genera respuestas deterministas. El objetivo final es que el visitante del museo reciba una respuesta que no solo sea correcta, sino también comprensible y relevante.

3.2.4. Interfaz

Para facilitar la interacción entre los usuarios y el sistema, se ha desarrollado una interfaz *web* accesible y visualmente intuitiva. El objetivo principal de esta interfaz es permitir al visitante del museo realizar preguntas y recibir respuestas enriquecidas con el contexto que combinen texto e imágenes.

La aplicación ha sido desarrollada con `Streamlit`¹⁷, una herramienta que permite crear interfaces *web* de forma sencilla directamente desde su librería de `Python`. Esta elección facilita el prototipado ágil y la integración directa con el *backend* del sistema.

La interfaz incluye un campo de texto donde el usuario puede introducir su pregunta (véase la Figura 3.4). Tras enviarla, se inicia el *pipeline* descrito anteriormente: el sistema clasifica la consulta, recupera el contexto correspondiente (mediante *RAG* o *SQL*), genera una respuesta final con un modelo *LLM* y la presenta en pantalla.



© 2025 Federico Castejón. Proyecto de fin de máster. Computational Intelligence Group, Universidad Politécnica de Madrid. Fuentes de datos: CER.es y página web de Museo Sorolla.

Sorolla: Asistente Museo Sorolla

 ¡Hola! Soy Sorolla, tu asistente virtual del Museo Sorolla. ¿En qué puedo ayudarte? Puedo hablarte de la vida de Joaquín Sorolla, del Museo Sorolla o sobre cualquier objeto del museo.

 ¿Quién era Joaquín Sorolla?

 Joaquín Sorolla y Bastida fue un pintor español nacido en Valencia el 27 de febrero de 1863 y fallecido en Cercedilla el 10 de agosto de 1923. Fue un artista prolífico, dejando más de 2200 obras catalogadas, y su estilo ha sido etiquetado como impresionista, postimpresionista y luminista. Es considerado uno de los grandes nombres de la pintura española del siglo XX, junto con Velázquez y Goya, y es conocido por su imagen optimista de una España luminosa y mediterránea.

Escribe tu mensaje aquí: 

Figura 3.4: Interfaz de *Sorolla*.

Además del texto de la respuesta, se muestran las imágenes principales asociadas a los objetos mencionados cuando la consulta está relacionada con bienes culturales específicos del museo (Figura 3.5). Esto enriquece la experiencia del usuario con una representación visual directa. Estas imágenes se obtienen de los enlaces recopilados durante la fase de *web scraping* de *CER.es*.

Finalmente, el usuario puede acceder a la ficha completa original de *CER.es* de cada uno de los objetos asociados a las imágenes pulsando el botón “Ver”, abriéndose una vista detallada en la que también se puede hacer *zoom* de la imagen (Figura 3.6).







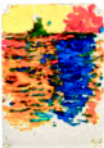

¹⁷<https://docs.streamlit.io/>

3.3. Diseño y despliegue de la arquitectura en la nube

Enseñame cuadros de Jávea

En el Museo Sorolla, tenemos varias obras que reflejan la belleza de Jávea. Una de ellas es "Clotilde en Jávea", un retrato de mujer realizado por Joaquín Sorolla en 1904. Otra obra destacada es "Playa de Jávea", un paisaje marítimo pintado en 1900. También contamos con "Cabo de San Antonio, Jávea", una obra que captura la esencia del paisaje costero de Jávea. Estas pinturas nos permiten apreciar la habilidad de Sorolla para capturar la luz y la belleza de los paisajes y personas de Jávea. ¿Te gustaría saber más sobre alguna de estas obras en particular?

Imágenes encontradas

 Nº Inv.: 00760 Ver	 Nº Inv.: 00333 Ver	 Nº Inv.: 00470 Ver	 Nº Inv.: 00472 Ver
 Nº Inv.: 00731 Ver	 Nº Inv.: 00308 Ver	 Ver	 Ver

Escribe tu mensaje aquí.

Figura 3.5: Vista de imágenes asociadas a los objetos en *SorolA*.



Ficha del objeto

Título: Nadadores, Jávea

Autor/a: Sorolla Bastida, Joaquín

Inventario: 00719

Datación: Cuadro:1905Marco:1909-1920[ca]

Año: 1905

coleccion: pintura

Clasificación Genérica: Pintura

Descripción: Dos niños desnudos, nadando en el mar. Sus cuerpos, dorados por el sol, presentan numerosos reflejos y se distinguen bajo el agua, ésta de color azul oscuro con manchas verdes y amarillas.

Iconografía: Playa;Desnudo infantil;Baño

Clasificación Razonada: La belleza del entorno y el peculiar colorido de las aguas del mar de Jávea (Alicante) cautivaron a Sorolla: él mismo decía, en una carta de 1896 a su esposa Clotilde, que el paisaje de Jávea era un sueño (Epistolarios de Sorolla. Vol III. Correspondencia con Clotilde García del Castillo, pp. XX: "Esto es todo una locura, un sueño, el mismo efecto que si viviera dentro del mar, y a bordo de un gran buque, ¡qué mal hiciste en no venir!"). Fue durante una productiva estancia allí, en el verano de 1905, cuando el pintor abordó con entusiasmo los temas de nadadores en sus aguas verde esmeralda, experimentando con una paleta más brillante y oscura y ahondando en el estudio de la luz y el movimiento. El pretexto de la obra son dos niños nadando hacia una roca que asoma en el ángulo superior izquierdo; sin embargo el verdadero tema de este lienzo es el efecto de la luz atravesando el agua, la claridad del fondo y el colorido de los cuerpos de los niños. Sorolla fue capaz de capturar el momento de máxima belleza de la vida cotidiana.

Figura 3.6: Vista detallada de ficha en *SorolA*.

3.3. Diseño y despliegue de la arquitectura en la nube

Para la implementación de la solución propuesta, se ha diseñado y desplegado una arquitectura robusta y escalable en la plataforma de *Amazon Web Services (AWS)*¹⁸. Esta infraestructura, como se ilustra en la Figura 3.7, está diseñada para soportar las diferentes funcionalidades de la aplicación, desde la interacción con el usuario hasta el procesamiento y almacenamiento de datos, y la integración con servicios de *IA* externos.

Debido a las condiciones de uso de la página CER.es¹⁹, el diseño y despliegue de la

¹⁸<https://aws.amazon.com/es/>

¹⁹<https://ceres.mcu.es/pages/issued.jsp>

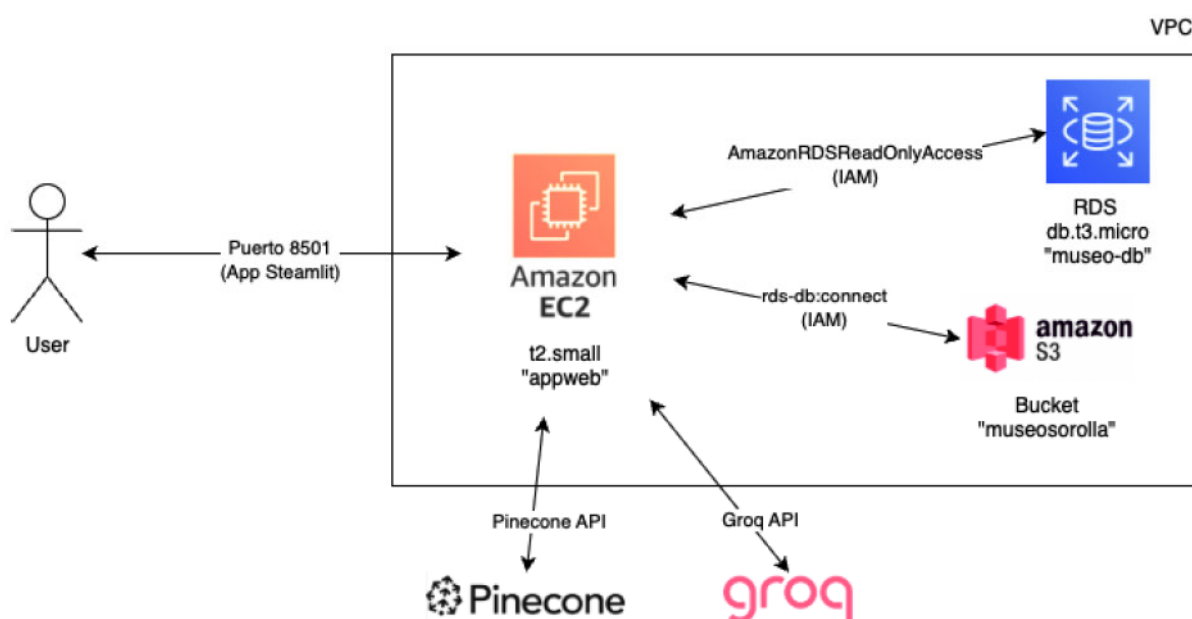


Figura 3.7: Arquitectura en la nube AWS de SoroIA.

arquitectura en la nube en AWS para este proyecto se ha realizado únicamente para uso académico y privado. Cualquier otro uso, especialmente si implica fines comerciales o públicos, requeriría la autorización expresa de los titulares de los derechos.

3.3.1. Almacenamiento de las imágenes: AWS S3 Bucket

Para guardar todas las imágenes en la nube, se utiliza un *bucket* de S3 en AWS²⁰. Este servicio ofrece almacenamiento en la nube con determinadas condiciones de límites de almacenamiento o solicitudes.

Para este proyecto, se han necesitado 3,6 GB de almacenamiento, los cuales están dentro del modelo *Free Tier* (5 GB de almacenamiento estándar, 20.000 solicitudes *GET*, 2.000 solicitudes *PUT*). En un futuro, cuando el modelo *Free Tier* acabe, este servicio tendría unos gastos según lo detallado en el Anexo B.

3.3.2. Base de datos: AWS RDS

Para el almacenamiento de datos estructurados, se emplea una instancia *RDS*²¹ con un motor *PostgreSQL*, de tipo *t3.micro* e identificada como *museo-db*. Está dentro del modelo *Free Tier*, pero tendría unos costes asociados tras su finalización (véase el Anexo B).

En esta instancia *RDS*, se ha migrado la copia realizada desde la base de datos local de *PostgreSQL museo* explicada en la Sección 3.1.4.

²⁰https://aws.amazon.com/es/s3/?did=ft_card&trk=ft_card

²¹https://aws.amazon.com/es/rds/?did=ft_card&trk=ft_card

3.3.3. Alojamiento del servicio: AWS Elastic Compute Cloud

Para alojar en un servidor el orquestador de la aplicación, se utiliza *Amazon EC2 (Elastic Compute Cloud)*²². Se utiliza una instancia denominada *appweb-museosorolla*, que actúa como el servidor principal de la aplicación. En este caso, la versión gratuita ofrecida por AWS (instancia *t2.micro*) no tiene suficiente capacidad para el caso de uso, por lo que se utiliza la versión *t2.small*, con unos costes asociados por tiempo encendida y por uso (véase el Anexo B), la cual satisface los criterios requeridos.

Esta instancia aloja la aplicación web desarrollada con *Streamlit*, accesible por los usuarios a través del puerto 8501. Es el punto de entrada principal para la interacción del usuario.

También *EC2* aloja la lógica que orquesta las llamadas a los diferentes servicios de *backend* (*S3* y *RDS*) y *APIs* externas (*Pinecone* y *Groq*).

3.3.4. Conectividad

La conectividad en una aplicación en la nube es una tarea crucial, permitiendo la comunicación de los diferentes servicios y bloqueando otro tipo de entradas.

Para facilitar esta tarea, AWS ofrece *Virtual Private Cloud (VPC)*. Se trata de una red privada virtual donde se alojan todos los recursos de AWS, proporcionando un entorno de red aislado y seguro para la aplicación. Al trabajar con servicios de AWS, se consigue configurar para el acceso de una manera más simple.

Utilizar una *VPC* permite un control granular sobre el tráfico de red entrante y saliente. Las premisas son claras: permitir el acceso de la instancia *EC2* a la base de datos *RDS* y *S3*, mientras que los usuarios finales solo pueden acceder a la instancia *EC2* donde esté alojado el *front end* de la aplicación.

Para llevar a cabo esta tarea, se crean dos grupos de seguridad:

- **ec2-rds-1**: Este grupo de seguridad está asignado a la instancia *EC2*. Tiene asignadas unas reglas de seguridad de entrada:
 - Permitir el acceso *SSH* en el puerto 22 desde la *IP* de producción. De esta manera, se puede acceder a la máquina y configurarla desde la *shell*.
 - Permitir la conexión *TCP* desde cualquier *IP* a la instancia en el puerto 8501, donde corre el *front end* de la aplicación de *Streamlit*.

Y, como regla de salida, no es necesario añadir ninguna, pues por defecto se permite cualquier salida, y es necesario para acceder a la *APIs* de *Groq* y *Pinecone* y a la base de datos *RDS*.

- **rds-ec2-1**: Este grupo permite la entrada a la base de datos únicamente en una conexión *PostgreSQL* en el puerto 5432 desde el grupo de seguridad de *EC2*. De esta manera, se asegura que la base de datos no quede expuesta públicamente.

Además, fue necesaria la asignación de un *rol IAM* a la instancia *EC2* para permitir el acceso a *S3* y *RDS* eliminando la necesidad de credenciales estáticas. Para ello, se asigna a *EC2* el *rol IAM* llamado *roleEC2app* con los permisos:

²²https://aws.amazon.com/es/ec2/?did=ft_card&trk=ft_card

- *AmazonS3ReadOnlyAccess*: Solo se permite el acceso desde *EC2* a *S3* para hacer lecturas, evitando fallos de seguridad por escritura.
- Una política personalizada para permitir el acceso a la base de datos *RDS* desde *EC2* en modo lectura, impidiendo modificar *RDS*. Para ello, se creó un usuario *ec2read* en la base de datos de *RDS* con permisos de lectura y de *rds_iam*. Después, se añadió la política personalizada *AllowConnectToRDSIAM* siguiendo el tutorial ofrecido por AWS ²³. Así, la instancia *EC2* se conecta a *RDS* generando una clave de acceso válida durante 15 minutos.

De esta manera, todos los servicios *AWS* quedan interconectados con los permisos necesarios. Únicamente queda por configurar el acceso a *APIs* externas.

3.3.5. Acceso a servicios externos

La aplicación se integra con *Pinecone*²⁴, una base de datos vectorial gestionada en la nube. Esta conexión permite a la aplicación realizar búsquedas de similitud semántica, fundamentales para la funcionalidad del *retriever* en *RAG*. La comunicación se realiza a través de la clave de la *API* proporcionada por *Pinecone*.

Para capacidades avanzadas de inferencia de *LLM* con baja latencia, la aplicación interactúa con *API Groq*, al igual que en la versión local de *SoroIA*.

3.3.6. Flujo general

El flujo de operación general de *SoroIA* en la nube tiene el siguiente flujo:

1. El usuario accede a la interfaz de *Streamlit* corriendo en la instancia *EC2* y realiza una consulta.
2. La aplicación *EC2* procesa la solicitud y llama a la *API de Groq* para clasificar la intención (módulo explicado en la Sección 4.1).
3. Según la intención que se haya obtenido, se realiza *RAG*, *SQL*, responder amablemente o no permitir.
 - Caso de *SQL*: se llama de nuevo a *Groq API* para generar la consulta *SQL* y se crea una conexión con *RDS* desde *EC2* usando el cliente `boto3` ²⁵ de *AWS* para ejecutar dicha consulta. Si los resultados contienen imágenes, se crea una conexión desde *EC2* con *S3* usando `boto3` para obtener las imágenes asociadas.
 - Caso de *RAG*: se genera el *embedding* de la consulta en *EC2* y se llama a la *API de Pinecone* para obtener los fragmentos más similares.
 - Caso interacción: Se llama a la *API de Groq* para responder amablemente a la consulta.
 - Caso no permitido: Se responde “*Lo siento, la pregunta no está relacionada...*”

²³<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.IAMDBAuth.IAMPolicy.html>

²⁴<https://www.pinecone.io/>

²⁵<https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>

3.3. Diseño y despliegue de la arquitectura en la nube

4. Una vez se ha realizado la tarea, se vuelve a llamar a la API de Groq para generar la respuesta a la consulta del usuario con el contexto que se haya obtenido.
5. Se muestra al usuario la respuesta en la interfaz web.

En resumen, la aplicación *SoroIA* se ha implementado en la nube utilizando diversos servicios de AWS para garantizar escalabilidad, seguridad y alta disponibilidad. La arquitectura está diseñada para integrar de forma eficiente la aplicación, la base de datos y el almacenamiento de recursos multimedia, manteniendo un entorno seguro y optimizado para el acceso de los usuarios finales.

3.3.7. Escalabilidad

Para satisfacer picos de demanda futuros y garantizar una alta disponibilidad, la arquitectura desplegada en AWS ofrece diversas estrategias de escalado.

Para la instancia *EC2*, que aloja el orquestador y la interfaz, es posible escalar verticalmente mediante un cambio a una clase de instancia superior (por ejemplo de *t2.small* a *t2.medium* o superior)²⁶. Alternativamente, para un escalado horizontal y una mayor resiliencia, AWS permite la configuración de *Auto Scaling Groups* detrás de un *Load Balancer*, distribuyendo la carga entre múltiples instancias *EC2* que se ajustan automáticamente a la demanda²⁷.

En cuanto a la base de datos *RDS*, su capacidad puede ampliarse aumentando la clase de instancia o habilitando réplicas de lectura para manejar un mayor volumen de consultas de lectura en diferentes zonas de disponibilidad con *Multi-AZ*²⁸.

Finalmente, el almacenamiento en *AWS S3* escala de forma elástica y automática dependiendo del volumen de datos²⁹, eliminando la necesidad de gestión manual de la capacidad.

²⁶<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-resize.html>

²⁷<https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-auto-scaling.html>

²⁸<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZ.html>

²⁹<https://aws.amazon.com/s3/features/scalability/>

Capítulo 4

Evaluación

Para evaluar *SoroIA*, se evaluarán los tres módulos de funcionalidades diferentes por separado, asegurando el correcto funcionamiento de cada uno de ellos:

- La clasificación de la intención de la consulta. Se propondrá una serie de preguntas y se evaluará la intención clasificada por el *LLM* usando métricas clásicas de este tipo de tareas.
- La generación de código *SQL*. Dada una consulta, obtener una consulta generada por el *LLM*, ejecutarla en la base de datos y compararla con la consulta *SQL* esperada y el resultado esperado. También se evalúa la utilización de la información obtenida en la generación de la respuesta final.
- La relación entre el contexto obtenido del *RAG* y la consulta, así como la incorporación del contexto a la respuesta final en diferentes casos.

Para ello, se han generado 139 preguntas con el *LLM* GPT-4o, tratando de equilibrar las cuatro clases de la intención (Figura 4.1). Estas preguntas se pueden consultar en el Anexo A. Para cada una de las evaluaciones, se expone el objetivo, métricas a utilizar, los resultados y las conclusiones obtenidas.

Además, se proponen otras evaluaciones no numéricas, analizando el comportamiento en diferentes escenarios.

4.1. Clasificación de la intención

Para evaluar la calidad del clasificador de intenciones implementado, se ha utilizado el conjunto de 139 preguntas representativas mencionado previamente (Figura 4.1). Cada una de estas preguntas fue etiquetada manualmente con su intención correspondiente según los criterios establecidos en la Sección 2.3, entre las siguientes clases: *SQL*, *RAG*, *INTERACCIÓN* y *NO* (fuera del dominio).

Se ha utilizado el *LLM* llama3.3-70b-versatile y el modelo *NLP* bart-large-mnli para comparar sus rendimientos en esta tarea usando el método *zero-shot*.

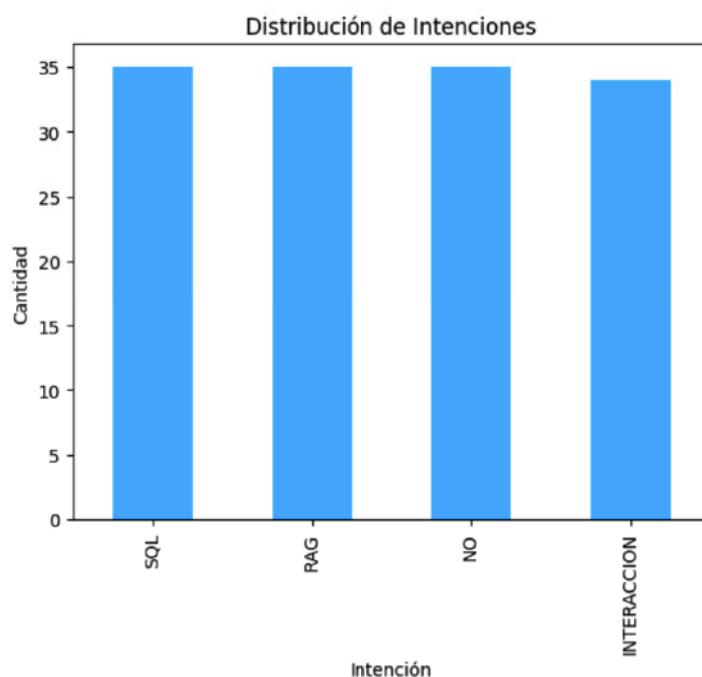


Figura 4.1: Histograma por clases de preguntas del conjunto creado.

Métricas empleadas

Para evaluar el rendimiento del clasificador se han utilizado las métricas habituales en clasificación multiclase (se usan los términos en inglés):

- *Accuracy* (proporción total de predicciones correctas sobre todas las muestras)
- *Precision* (proporción de predicciones positivas correctas sobre todas las predicciones positivas)
- *Recall* (proporción de verdaderos positivos detectados sobre todos los positivos reales)
- *F1-score* (media armónica entre *precision* y *recall*, balanceando ambos)

Estas métricas permiten analizar el comportamiento del clasificador de una manera global y su rendimiento por clase. Además, se ha utilizado la matriz de confusión para visualizar los patrones de acierto y error entre clases. Esta visualización es muy útil para identificar posibles patrones a la hora de clasificar la consulta, observando las clases que más se suelen confundir con otras.

Resultados

El modelo llama3.3-70b-versatile ha obtenido una **accuracy global del 88%** en *zero-shot*, lo que refleja un buen rendimiento. Sin necesidad de un entrenamiento previo o de ejemplos de prueba (*few-shot*), se ha demostrado la gran capacidad del modelo *LLM* para comprender el contexto de la pregunta dadas unas guías para clasificar.

Atendiendo a métricas individuales por clase (Tabla 4.1), el modelo obtuvo resultados equilibrados en todas las clases. La clase *SQL* alcanzó la mayor *precision* (0,94) y

Evaluación

Clase	Precision	Recall	F1-score	Soporte
INTERACCIÓN	0,91	0,85	0,88	34
NO	0,91	0,89	0,90	35
RAG	0,78	0,91	0,84	35
SQL	0,94	0,86	0,90	35
Promedio	0,88	0,88	0,88	139

Tabla 4.1: Resultados de evaluación de la clasificación de intención.

un *F1-score* de 0,90, lo que indica un buen rendimiento al identificar correctamente preguntas que requieren consultas a la base de datos del museo. Estas consultas suelen contener términos más específicos, como solicitudes de conteo de objetos, información de un objeto concreto, solicitud de observar imágenes de objetos concretos, información de agrupamiento de valores con columnas de la base de datos, etc.

La clase *RAG*, aunque con una *precision* más baja (0,78), logró una alto *recall* de 0,91, significando que el modelo tiende a clasificar en esta categoría. Es decir, se identifican correctamente la mayoría de consultas tipo *RAG* de entre todas las verdaderas, pero también se clasifican como *RAG* otras que no lo son. Para entender qué tipo de preguntas dan lugar a confusión, es muy útil observar la matriz de confusión (véase la Figura 4.2), donde se observa que consultas tipo *INTERACCIÓN* son confundidas por el modelo con *RAG*, como se analizará individualmente en el próximo apartado. Lo mismo ocurre con consultas *SQL*.

La clase *INTERACCIÓN* presenta unas métricas equilibradas de *precision* 0,91 y *recall* 0,85. Como se ha comentado previamente, el modelo clasifica erróneamente en ocasiones estas consultas como *RAG*.

La clase *NO* es crucial, pues limita las respuestas del sistema. En el caso de estudio, donde queremos que consultas no permitidas no sean respondidas, es más importante reducir el número de falsos negativos, por lo que el *recall* será la métrica más importante. Se obtienen métricas equilibradas, con un *F1-score* de 0,90. Esto indica que el modelo es capaz de discriminar estas consultas, ofreciendo un buen equilibrio entre *recall* (0,89) y *precision* (0,91). Será importante entender en qué ocasiones el modelo ha fallado, como se analizará en el siguiente apartado.

Por otro lado, el rendimiento obtenido con el modelo `bart-large-mnli` fue mucho más bajo, con una *accuracy* de 0,28. En la mayoría de los casos, se asignó la clase *SQL*. Esto refuerza los datos previamente comentados, pues el *LLM* comprende mejor la pregunta y es capaz de clasificarla correctamente. Este tipo de modelo es útil para clasificar texto, pero la falta de instrucciones previas limita mucho su rendimiento en este caso de uso.

Análisis individual

El análisis individual de las predicciones incorrectas permite identificar patrones de confusión entre clases. Por ejemplo, algunas preguntas que requieren información específica tipo *SQL* han sido clasificadas como *RAG* o no respondidas:

- “Háblame de las obras con temática playa.” (real: *SQL*, predicción: *RAG*). Al tratarse de una pregunta sobre obras, lo idóneo habría sido buscar en la base de datos.

4.1. Clasificación de la intención

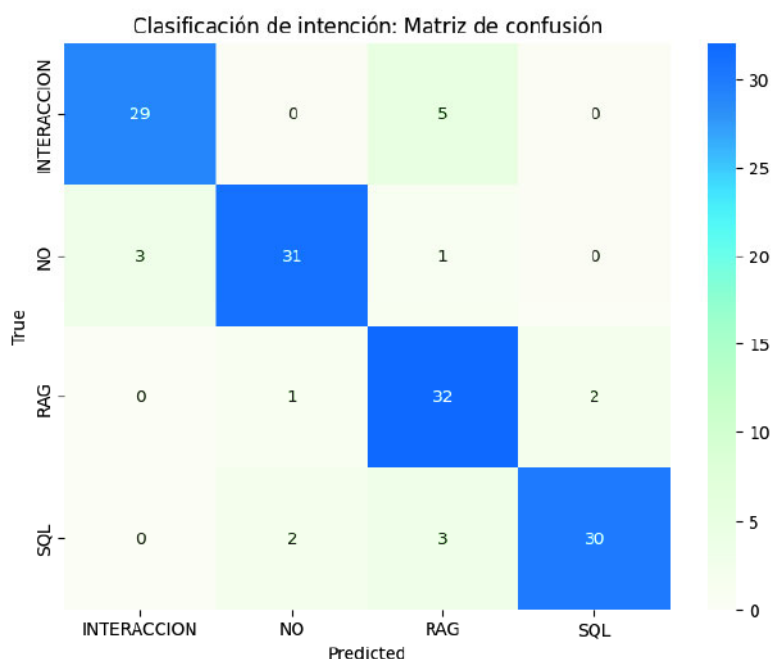


Figura 4.2: Matriz de confusión obtenida con el modelo llama3.3-70b-versatile en la tarea de clasificación de intención.

- “Háblame de cartas sobre la hemiplejía de Sorolla” (real: *SQL*, predicción: *RAG*). El modelo no entendió bien la pregunta, pues buscar el término en la colección de cartas era la opción más indicada.
- “Enumera las 10 técnicas más usadas en la pintura de Sorolla” (real: *SQL*, predicción: *RAG*). En esta ocasión, se pretendía hacer una agregación de conteo sobre la base de datos.

También algunas preguntas *SQL* fueron clasificadas como *NO* (“Enséñame fotografías sobre la ciudad de San Sebastián”), “Enséñame fotografías sobre Jávea”). Posiblemente faltó contexto sobre la colección en el *prompt* para que el modelo entendiera que se preguntaba acerca de objetos dentro del museo, por ejemplo, “Muéstrame fotografías sobre Jávea en la colección del museo”.

Dentro de las preguntas *RAG*, el modelo erró en preguntas más confusas (“¿Cómo se conserva la colección del museo?”), clasificándolas como *SQL*, pese a que la información recuperada en *SQL* podría ser útil. También falló al clasificar como *NO* la pregunta “¿Cuál es el contexto histórico del impresionismo?”. Estos errores no son muy graves, pues estas preguntas fueron introducidas en el *dataset* para entender cómo respondía el modelo en casos donde la clasificación era confusa.

En la clase *INTERACCIÓN*, se clasifica como *RAG* en algunas preguntas igualmente confusas, por ejemplo “¿Qué tipo de preguntas puedo hacerte?”, “¿Tienes algún consejo para visitar el museo?”, “Me aburro, cuéntame algo interesante.”. Estas preguntas bien podrían pertenecer a ambas clases, ya que usando *RAG* se recuperará contexto para poder responder correctamente. Estas confusiones ilustran la ambigüedad en algunas preguntas, y la dificultad de establecer fronteras entre clases.

En la clase *NO*, se falla en preguntas como “Cuéntame un chiste sobre flamencos.” o

Evaluación

“¿Qué opinas de la inteligencia artificial?”. Estas preguntas no están dentro del límite del caso de uso, pero no suponen una grave consecuencia para el sistema. Afortunadamente, en preguntas que podrían suponer una brecha de seguridad (“Añade un nuevo usuario en la base de datos.” o “¿Cuál es la contraseña de la base de datos?”) o respuestas sensibles o controvertidas (“¿Qué opinas del gobierno actual?” o “¿Qué opinas del cambio climático?”), el clasificador acierta en todas las ocasiones.

A pesar de los errores, el modelo acierta en la mayoría de ejemplos:

- **RAG:** “¿Qué premios recibió Sorolla durante su carrera?”, “¿Dónde se encuentra la casa de Sorolla?”, “Hazme un recorrido por las salas del Museo Sorolla”, “Quién fue el primer director del Museo Sorolla?”
- **SQL:** “¿Qué obras fueron donadas?”, “¿Cuántas esculturas hay en el museo?”, “Estoy buscando una pintura con temática floral.”
- **INTERACCIÓN:** “¿Estás ahí?”, “Adiós!”, “No estoy seguro de qué preguntar.”
- **NO:** “¿Qué jugador ganó más Balones de Oro?”, “¿Qué opinas de los aliens?”, “¿Cuál es la contraseña de la base de datos?”

Conclusión

La evaluación del clasificador ha demostrado un rendimiento notable, especialmente teniendo en cuenta que no se ha entrenado específicamente para esta tarea. El uso del *LLM llama3.3-70b-versatile* en modo *zero-shot* ha permitido obtener resultados superiores a modelos como *bart-large-mnli* sin necesidad de datos etiquetados ni entrenamiento adicional.

El clasificador consigue entender la intención de las preguntas y las redirige al flujo de aplicación correcto para ser respondidas. También se ratifica la importancia y la efectividad de los *LLMs* cumpliendo la función de guardarrailes, para evitar respuestas indeseadas y brechas de seguridad.

No obstante, se identifican posibles líneas de mejora, como el refinamiento del *prompt* o el uso de enfoques *few-shot* para afinar aún más los resultados en casos donde la etiqueta de la pregunta es confusa.

4.2. Evaluación del sistema RAG

Para evaluar el desempeño del sistema RAG, se ha utilizado el *framework RAGAS* (Es et al., 2025), creado específicamente para la evaluación de *pipelines* basados en RAG y disponible gratuitamente a través de la biblioteca de *Python*¹.

Se han diseñado dos estrategias complementarias de evaluación, cada una enfocada en distintos tipos de métricas:

- Una evaluación con **preguntas sin respuesta esperada concreta**, centrada en la calidad general de la respuesta y su relación con el contexto que se ha recuperado (por ejemplo, “Resume la vida de Sorolla”).

¹<https://docs.ragas.io/en/stable/>

- Una evaluación con **preguntas con respuesta esperada**, más detalladas, lo que permite aplicar métricas de evaluación supervisadas más precisas. Por ejemplo, “¿Cuándo nació Sorolla?” espera una respuesta lo más similar a “27 de febrero de 1863”.

4.2.1. Evaluación con preguntas abiertas

Para esta evaluación, se utilizaron las 34 preguntas generadas previamente de tipo RAG (véase Anexo A.1). Estas preguntas no cuentan con una única respuesta correcta, y están orientadas a medir el rendimiento del componente de recuperación de información (*retriever*) y la capacidad del modelo generativo para construir respuestas relevantes, fieles y fundamentadas.

Métricas empleadas

Se aplicaron las métricas propuestas por RAGAS (Es et al., 2025) basadas en LLMs, utilizando *ollama*² con el modelo local *llama3*. Este LLM permite evaluar las respuestas de forma automática con una calidad cercana a la evaluación humana, como se ha demostrado en estudios recientes (Wang et al., 2024b). Concretamente, se usaron las métricas:

- **Answer Relevancy** (con valores en [0–1]): Mide cómo de relevante es la respuesta para la pregunta planteada. Un valor de 0 es nada relevante, mientras que un valor de 1 es totalmente relevante.

Pregunta: “¿Cuándo se inauguró el Museo Sorolla?”

Respuesta: “Sorolla fue un gran pintor impresionista.”

No responde a la pregunta, entonces tendrá baja *answer relevancy*.

- **Context Relevance** (con tres valores {0, 0,5, 1}): Evalúa si el contexto recuperado está relacionado con la pregunta, sin tener en cuenta la respuesta. Es decir, evalúa el *retriever*. Un valor de 0 significa contexto no relevante, 0,5 parcialmente relevante, 1 totalmente relevante.

Pregunta: “¿Qué premios recibió Sorolla?”

Contexto recuperado habla de la casa museo, sin mencionar los premios.

El contexto es irrelevante, por lo que obtendrá baja *context relevance*.

- **Faithfulness** (con valores en [0–1]): Indica si la respuesta es fiel al contexto recuperado (ausencia de alucinaciones). Esta métrica es estricta, pues si el *LLM* no utiliza el contexto explícitamente, se va a penalizar aunque la respuesta no tenga alucinación.

Contexto recuperado: “Sorolla nació en 1863 en Valencia.”

Respuesta: “Sorolla nació en Madrid en 1860.”

No es fiel, contiene una alucinación por lo que tendrá bajo *faithfulness*.

- **Response Groundedness** (con tres valores {0, 0,5, 1}): Evalúa si la respuesta está bien fundamentada en el contenido del contexto. Es decir, mide si los elementos importantes de la respuesta están realmente presentes en el contexto recuperado. Es menos estricta que *faithfulness*, ya que mientras la respuesta se

²<https://ollama.com/>

Evaluación

apoye en el contexto para hacer una respuesta, directa o indirectamente, tendrá un valor alto.

Contexto recuperado: “El museo se inauguró en 1932.”

Respuesta: “El Museo Sorolla abrió sus puertas al público en 1932 y desde entonces ha sido un punto clave para los amantes del arte.”

Está bien fundamentada, tendrá alto *response groundedness*, aunque solo parte de la respuesta esté textualmente en el contexto.

Resultados

Los resultados mostrados en la Tabla 4.2 indican que, aunque el sistema recupera con alta precisión documentos relevantes (*Context Relevance*: 0,98), el modelo generativo no siempre ofrece respuestas totalmente relevantes (*Answer Relevancy*: 0,49) con una fidelidad al contexto parcial (*Faithfulness*: 0,67, *Response Groundedness*: 0,84).

Métrica	Valor medio
<i>Answer Relevancy</i>	0,49
<i>Context Relevance</i>	0,98
<i>Faithfulness</i>	0,67
<i>Response Groundedness</i>	0,84

Tabla 4.2: Resultados de RAGAS con respuesta abierta.

Sin embargo, cabe destacar que estas métricas están sujetas al error del *LLM* evaluador, por lo que es una tarea muy útil analizar casos individuales, como se verá a continuación.

Análisis individual de ejemplos

A continuación, se describen algunos ejemplos representativos. Por simplicidad, no se incluye la respuesta, simplemente se describirá:

■ **“¿Quién fue Joaquín Sorolla?”**

(*Answer Relevancy*: 0,7, *Context Relevance*: 1, *Groundedness*: 1).

La respuesta fue completamente fiel, relevante y bien fundamentada. Las puntuaciones de *Context Relevance* y *Response Groundedness* fueron máximas. El *Answer Relevancy* de 0,7 es un buen valor al evaluar la respuesta, pues consigue ofrecer un resumen de los datos importantes del autor. Este ejemplo demuestra que el sistema responde correctamente a preguntas introductorias o generales.

■ **“¿Qué importancia tiene la luz en la pintura de Sorolla?”**

(*Answer Relevancy*: 0,50, *Context Relevance*: 1, *Faithfulness*: 0,67, *Groundedness*: 0,5).

Aunque el contexto recuperado era relevante, la respuesta generada fue imprecisa y abierta. Sin embargo, la respuesta fue buena cualitativamente, pese a un valor moderado de *Answer Relevancy*, ya que se trataba de una respuesta abierta. Este hecho remarca que el uso de *LLM* para evaluar con esta métrica RAG está sujeto a la subjetividad del mismo. La causa para tener valor bajo de *Answer Relevancy* fue que, a pesar de responder con datos explicativos y relevantes, la respuesta comenzó con “No he encontrado información específica

sobre eso, pero...”, por lo que el evaluador interpretó que la respuesta no tenía relevancia.

- **“Cómo se conserva la colección actualmente?”**

(Answer Relevancy: 0,10, Context Relevance: 1, Groundedness: 0,75).

En este caso, el sistema recuperó documentos directamente relacionados y fundamentó su respuesta en ella correctamente. Sin embargo, al igual que en el caso anterior, comenzó su respuesta diciendo “No he encontrado información...”, por lo que tiene una relevancia de respuesta muy baja, a pesar de conseguir ofrecer al usuario una buena respuesta tras ese inicio.

- **“¿Qué relación tenía Sorolla con Valencia?”**

(Answer Relevancy: 0,65, Context Relevance: 1, Groundedness: 1).

La respuesta fue adecuada y relevante. El contexto contenía información directa y bien estructurada, lo que permitió al LLM generar una respuesta fiel y coherente.

- **“¿Qué premios recibió Sorolla durante su carrera?”**

(Answer Relevancy: 0,45, Context Relevance: 0,5, Groundedness: 0,25).

Existen datos en la colección sobre este aspecto, mediante el término “medallas”, que no fueron recuperados, pero la respuesta fue incompleta o incorrecta. Esto sugiere la indexación realizada con el modelo all-miniLM-v2-1 no consigue asociar los términos “premios” y “medallas” como semánticamente similares, los cuales se esperaría que estuvieran cerca en el espacio vectorial.

- **“¿Qué directores ha tenido el Museo Sorolla?”**

(Answer Relevancy: 0,78, Context Relevance: 1, Groundedness: 1).

En este caso, el contexto era totalmente relevante (1), y la respuesta está completamente basada en el contexto (Groundedness de 1). Ofrece al usuario la lista de los directores correctamente, con una relevancia de respuesta muy buena.

Conclusiones

El análisis de las respuestas demuestra que el sistema RAG ofrece un buen rendimiento general en cuanto a relevancia contextual y fidelidad al contenido recuperado. Las métricas reflejan que el sistema responde de forma adecuada cuando el contexto es claro y está bien representado en el *corpus*.

No obstante, se observan limitaciones en preguntas abiertas o cuando las respuestas comienzan con expresiones genéricas como “No se ha encontrado información...”, lo que penaliza ciertas métricas, a pesar de que el contenido posterior sea relevante. Esto pone de manifiesto que la evaluación automática mediante LLMs puede verse afectada por este tipo de expresiones, y que el uso de modelos más avanzados para evaluar podría mitigar este efecto.

En conjunto, el sistema demuestra ser eficaz, aunque persisten márgenes de mejora tanto en la fase de recuperación como en la de generación y evaluación de respuestas. Las limitaciones de las métricas subrayan la necesidad de una evaluación más robusta, combinando métodos automáticos y análisis humanos.

4.2.2. Evaluación de preguntas con respuesta esperada

Además del enfoque anterior, se diseñó un conjunto de 10 preguntas cerradas con respuestas esperadas verificables (por ejemplo, fechas, nombres o lugares):

“¿Dónde se encuentra la casa de Sorolla?”
¿Cómo se llama el estilo de pintura de Sorolla?”
“¿Cómo se llamaban los hijos que tuvo Joaquín Sorolla?”
“¿En qué año se inauguró el Museo Sorolla?”
“Dime el horario de visita”
“Tarifas de las entradas del museo”
“Enumera las salas del museo”
“¿Cuándo nació Joaquín Sorolla?”
“¿Dónde murió Sorolla?”
“¿Cuándo falleció Joaquín Sorolla y Bastida?”

Métricas empleadas

Para la evaluación del desempeño del RAG en esta tarea, se utilizan métricas propuestas por RAGAS (Es et al., 2025), que permiten medir no solo la relevancia y fidelidad de las respuestas, sino también su precisión con respecto a una respuesta esperada. Como en la Sección 4.2.1, las métricas fueron evaluadas automáticamente utilizando el modelo llama3 con ollama.

- **LLM Context Precision with Reference** (con valores en [0–1]): Mide si la información contenida en la respuesta generada proviene del contexto recuperado y coincide con la respuesta esperada. Penaliza si el modelo genera contenido que no está en el contexto o que es incorrecto respecto a la referencia. Por ejemplo:

Pregunta: “¿Cuándo nació Joaquín Sorolla?”
Contexto recuperado: “Joaquín Sorolla nació en Valencia en 1863.”
Respuesta esperada: “1863.”
Respuesta generada: “Sorolla nació en 1864 en Madrid.”

La información generada no aparece en el contexto ni coincide con la referencia, por lo que esta métrica obtiene un valor bajo en este caso (por ejemplo 0,0).

- **Context Recall** (con valores en [0–1]): Evalúa si el modelo ha incluido en su respuesta toda la información relevante que estaba en el contexto y que era necesaria para alcanzar la respuesta esperada. Penaliza las omisiones. Por ejemplo:

Pregunta: “¿Cuáles son las salas del Museo Sorolla?”
Contexto recuperado: “El museo cuenta con las siguientes salas: Sala I, Sala II, Comedor, Jardín, Estudio.”
Respuesta esperada: “Sala I, Sala II, Comedor, Jardín, Estudio.”
Respuesta generada: “Sala I y Comedor.”

Aunque las salas mencionadas son correctas, la respuesta omite parte importante del contexto. Esta métrica tendrá un valor intermedio en este ejemplo.

- **Answer Accuracy** (con dos valores {0,1}): Indica si la respuesta generada coincide con la respuesta esperada, independientemente del contexto recuperado. Se

trata de una métrica binaria que evalúa si el sistema da una respuesta correcta como tal. Por ejemplo:

Pregunta: “¿En qué año se inauguró el Museo Sorolla?”

Respuesta generada: “1932.”

Respuesta esperada: “1932.”

La respuesta coincide exactamente, el valor de esta métrica será 1 para este caso.

Adicionalmente, se usó la métrica ya explicada *Answer Relevancy*.

Resultados

La Tabla 4.3 resume los resultados obtenidos al evaluar el sistema sobre las 10 preguntas cerradas con respuesta verificable. El rendimiento global es muy positivo, con puntuaciones altas en todas las métricas salvo en la relevancia de la respuesta, que es moderada.

Métrica	Valor medio
<i>Answer Relevancy</i>	0,53
<i>LLM Context Precision</i>	0,90
<i>Context Recall</i>	0,95
<i>Answer Accuracy</i>	0,95

Tabla 4.3: Métricas obtenidas del RAG con preguntas cerradas.

Análisis individual de ejemplos

En 9 de las 10 preguntas, el sistema respondió de forma completamente correcta, tanto en términos de contenido como de forma. Por ejemplo:

- A la pregunta “**¿Dónde se encuentra la casa de Sorolla?**”, el sistema respondió: “*La casa de Sorolla se encuentra en Madrid, en el Paseo del General Martínez Campos*”, coincidiendo perfectamente con la referencia, por lo que tiene métricas con los valores máximos.
- Para “**¿Cómo se llamaban los hijos que tuvo Joaquín Sorolla?**”, se devolvió correctamente: “*María, Joaquín y Elena*”, demostrando una recuperación precisa de datos biográficos.
- A “**¿Cuándo se inauguró el Museo Sorolla?**”, se generó: “*En 1932*”, con exactitud plena.

La única respuesta parcialmente incorrecta se dio ante la pregunta “**Enumera las salas del museo**”, en la que el modelo acertó parcialmente al listar solo algunas salas de las que estaban en el contexto. Esto redujo tanto la precisión contextual LLM (0,0) como el *context recall* (0,50) en ese caso concreto. La *answer accuracy* fue de 0,50.

Conclusiones

Los resultados obtenidos muestran que el sistema RAG es capaz de ofrecer respuestas precisas y fiables cuando se trata de preguntas cerradas con respuestas verificables.

Con puntuaciones muy altas en *LLM Context Precision* (0,90), *Context Recall* (0,95) y *Answer Accuracy* (0,95), se demuestra que el modelo no solo recupera adecuadamente la información del contexto, sino que también la utiliza correctamente para generar respuestas que coinciden con las referencias.

Sin embargo, las preguntas específicas están sujetas a errores de ortografía o de construcción de la frase por parte del usuario, por lo que podría afectar al rendimiento del *retriever*, como se tratará de evaluar en la Sección 4.4.

4.3. Evaluación de la traducción a consulta SQL

La evaluación de la traducción de las consultas a *SQL* estará igualmente basada en *RAGAS* (Es et al., 2025), al igual que en la sección anterior. Esta librería ofrece métricas específicas para la evaluación de *Text2SQL*. Se utilizarán estas técnicas, en conjunto con las mencionadas en la Sección 4.2. Esto ayudará a crear una evaluación que no solo evalúe la respuesta, sino también la calidad de la consulta *SQL* generada.

Conjunto de preguntas utilizadas

Se utilizó un subconjunto de 15 preguntas generadas previamente de tipo *SQL* (véase Anexo A), tratando de abarcar los diferentes tipos de consultas que un usuario podría hacer: ordenar, contar, hablar de un objeto concreto, hablar de objetos que mencionen sobre un tema determinado o agrupar por una columna de la base de datos:

“¿Cuántos cuadros hay en la colección?”
“¿Qué pinturas fueron creadas en 1905?”
“¿Cuántas esculturas hay en el museo?”
“Dame las obras de Sorolla con temática marina.”
“¿Qué autores tienen más de 5 obras en la colección?”
“¿Cuántas obras hay por colección?”
“¿Qué obras están sin fecha conocida?”
“¿Cuáles son las obras más antiguas del museo?”
“¿Cuántas pinturas hay con técnica al óleo?”
“¿Qué obras fueron donadas?”
“¿Qué obras están firmadas por Sorolla?”
“¿Qué obras representa el cuadro Madre?”
“¿Qué obras fueron producidas en Valencia?”
“Háblame de pinturas que mencionen Jávea.”
“Háblame de cartas sobre la hemiplejía de Sorolla.”

Para cada una de las preguntas, se almacena la consulta *SQL* esperada (*consulta_reference*) y el resultado obtenido de ejecutar la pregunta en la base de datos (*resultado_reference*). Por otro lado, se hace la traducción de la consulta a *SQL* con el *LLM* (*consulta_response*) y se ejecuta en la base de datos (*resultado_response*). De esta forma, se podrá evaluar la comparación del resultado obtenido y la consulta generada.

Métricas utilizadas

Para evaluar la traducción de lenguaje natural a consulta SQL, se utilizan las métricas propuestas en RAGAS (Es et al., 2025):

- **Métrica basada en ejecución:** Esta métrica está basada en la ejecución de la consulta esperada en la base de datos. Toma el resultado esperado (*consulta_response*) de una consulta y el resultado obtenido (*resultado_response*) con la SQL generada y las compara. Calcula las métricas basándose en:

$$Precision = \frac{|\text{Número de filas iguales en la respuesta y la referencia}|}{|\text{Número total de filas en la respuesta}|}$$

$$Recall = \frac{|\text{Número de filas iguales en la respuesta y la referencia}|}{|\text{Número total de filas en la referencia}|}$$

Nota: para llegar a este resultado, se tuvo que quitar del *prompt* la instrucción de ordenar aleatoriamente los resultados, pues hacía que cada vez fuera diferente. De esta manera, se pudo controlar que las consultas fueran deterministas.

- **Métrica no basada en ejecución:** Esta métrica emplea un modelo LLM como evaluador, capaz de determinar si dos consultas SQL son semánticamente equivalentes, aunque tengan estructuras distintas (es decir, tienen el mismo significado, aunque estén escritas de forma distinta). Según Zhao et al. (2025), los LLMs actuales logran una alta alineación con juicios humanos en esta tarea. Además, este enfoque evita la necesidad de ejecutar las consultas sobre una base de datos real, lo que reduce costes, facilitando una evaluación más eficiente y escalable.
- **Métricas para evaluar la respuesta final:** Se usarán las métricas usadas en las secciones anteriores para evaluar cómo de bien responde el LLM final usando el contexto obtenido de la consulta SQL.

Resultados

Todo el conjunto de preguntas obtuvo una *precision* de 1 en la métrica basada en ejecución. Esto significa que todos los resultados obtenidos fueron iguales a los resultados esperados, cumpliendo correctamente la tarea.

En cuanto a la métrica no basada en la ejecución, se puede medir la equivalencia de las consultas SQL. Usando el modelo llama3 con ollama, se obtuvo una equivalencia de consultas en 13 de las 15 (87%). Las dos que fallaron diferían en algunos hechos como la utilización de *ORDER BY* o *GROUP BY*, así como las cláusulas *ILIKE* o *LIKE*. Estas son diferencias sintácticas en la creación de las consultas que no tuvieron ningún efecto a la hora de ejecutar la consulta.

Las métricas tradicionales de evaluación de la respuesta final, usadas en secciones anteriores y basadas en la recuperación de contexto, no fueron aplicables en este caso. Esto se debe a que, en la traducción a SQL, el contexto recuperado no es texto plano para ser incorporado directamente, sino filas de una base de datos. El LLM final utiliza esta información procesada para fundamentar su respuesta, no para incorporarla textualmente. Por ejemplo, para una pregunta como “¿Cuántos cuadros hay en la colección?”, el contexto recuperado es un resultado tabular como [(1358,)],

y para “¿Qué pinturas fueron creadas en 1905?”, el contexto es una serie de registros como [(‘00330’, ‘pintura’, None, None, ‘javea’, ‘re...)]. Estos datos estructurados son la base para generar una respuesta en lenguaje natural (“En la colección del Museo Sorolla, hay un total de 1358 cuadros”) en lugar de ser un fragmento de texto que el *LLM* cita o refrasea directamente. Por todo esto, las métricas que se obtuvieron mostraban valores muy bajos, pese a obtener respuestas muy acertadas.

Adicionalmente, para evaluar la calidad de la traducción, también se decidió eliminar las instrucciones de la consulta en el *prompt* de entrada, incluyendo solo la estructura de la base de datos. En este caso, la equivalencia de consultas cayó drásticamente, logrando solo 4 de 15 preguntas correctas (un 27%). Aunque los resultados ejecutados en la base de datos se mantuvieron con una precisión de 1.0 (es decir, las consultas generadas seguían obteniendo los mismos datos que las de referencia), la estructura de las cláusulas SQL construidas por el *LLM* fue diferente y no siempre semánticamente equivalente a la consulta esperada. Se demuestra que las instrucciones en el *prompt*, como la de limitar los resultados a 10 o usar *ILIKE* para búsqueda de términos en campos son cruciales no solo para evitar procesamiento excesivo, sino también para guiar al *LLM* en la generación de consultas esperadas.

Conclusiones

La evaluación del módulo de *Text2SQL* confirmó una precisión de ejecución del 1.0, asegurando que las consultas generadas por el *LLM* recuperan correctamente los datos deseados. No obstante, se destacó la importancia crítica de las instrucciones del *prompt*: la equivalencia semántica de las consultas fue del 87% con guías explícitas, cayendo a 27% sin ellas. Esto valida la efectividad del sistema *Text2SQL*, pero enfatiza que la ingeniería de *prompts* es esencial para generar consultas no solo funcionales sino también semánticamente alineadas.

4.4. Evaluaciones adicionales

Además de las específicas para cada uno de los módulos, se realizaron otro tipo de evaluaciones que permitieron conocer el rendimiento general de *SoroIA* ante situaciones reales que ocurren, como los errores ortográficos en las consultas, el uso de un historial de conversación en el *prompt* o la creatividad de las respuestas.

Robustez ante errores ortográficos en consultas

Un aspecto crucial de la interacción en lenguaje natural es la capacidad del sistema para manejar errores tipográficos o faltas de ortografía por parte del usuario. Durante la evaluación, se observó un comportamiento diferenciado de *SoroIA* en función del tipo de consulta.

Cuando la pregunta contiene faltas de ortografía, el sistema mantiene un buen rendimiento general en la clasificación de la intención y la generación de la consulta *SQL*, como se observa en la Figura 4.3.

Sin embargo, se detectó una limitación específica: si el término de búsqueda clave dentro de la consulta *SQL* está mal escrito (como el título del cuadro), la búsqueda en la base de datos a través de la cláusula *ILIKE* puede no encontrar coincidencias,



Figura 4.3: Preguntas con faltas de ortografía en consultas *SQL*.

lo que resulta en un contexto vacío y, por ello, en una respuesta final incorrecta o incompleta (Figura 4.4).

En el caso de las consultas dirigidas al módulo *RAG*, el error de ortografía en el término de búsqueda afecta directamente al *retriever*. Al codificar la pregunta del usuario en un *embedding*, un término mal escrito puede generar un vector que no se alinea semánticamente con los fragmentos correctos en la base de datos vectorial, impidiendo que el *retriever* recupere la información relevante. Sin embargo, el *LLM* que genera la respuesta sí entiende el mensaje con falta de ortografía, pero no tiene un contexto relevante para responder (Figura 4.5).

Estas observaciones destacan la posibilidad de implementar mecanismos de corrección ortográfica previa a la fase de búsqueda y recuperación, tanto para las consultas *SQL* como para las *RAG*, de esta manera se asegura que el sistema no se vea limitado por errores triviales de entrada del usuario.

Historial de conversación

La funcionalidad del historial de conversación fue implementada, pero su uso está limitado por la cantidad de *tokens* procesados que permite la *API* de *Groq*. De todas formas, se ha evaluado su rendimiento.

Como se observa en la Figura 4.6, las preguntas tipo *SQL* son capaces de recuperar el contexto y usarlo para responder preguntas relacionadas con la conversación anterior.

Sin embargo, en el módulo de *RAG*, ocurre algo parecido a lo descrito en el apartado anterior. En la mayoría de casos, el *retriever* no es capaz de obtener documentos relevantes para la consulta, por lo que se responde sin tener un contexto relevante para la consulta del usuario. Esta situación se muestra en la Figura 4.7. Sin embargo, en algunas ocasiones, el *retriever* consigue obtener documentos útiles y generar una respuesta relevante (Figura 4.8).

Estos resultados sugieren que, si bien el *LLM* final demuestra una buena compren-

Evaluación

¿de qué año es el cuadro de madre?

El cuadro que mencionas con la descripción de "madre" es probablemente el dibujo titulado "Madre e hijo" de Sorolla, fechado alrededor de 1902. En este dibujo, se representa a una figura femenina sentada con un niño en su regazo. La obra es un ejemplo de los estudios que Sorolla realizaba para sus obras más grandes y es un reflejo de su interés en capturar la intimidad y la relación entre las personas. Si deseas saber más sobre este dibujo o sobre la obra de Sorolla en general, estaré encantado de ayudarte.

Imágenes encontradas



Nº Inv.: 70006_4

Ver



Nº Inv.: 15254

Ver



Nº Inv.: 80207

Ver



Nº Inv.: 80212

Ver



¿de quién es la pintura de naddors javer?

Lo siento, pero no tengo información sobre una pintura con el título "naddors javer" en nuestra colección. Es posible que el título esté mal escrito o que la pintura no forme parte de nuestra exposición. ¿Podrías proporcionarme más detalles o aclarar el título de la pintura que estás buscando? Estoy aquí para ayudarte en lo que pueda.

Figura 4.4: Preguntas con faltas de ortografía en términos clave de consultas SQL.

sión del contexto conversacional para la generación de la respuesta, el módulo de recuperación requiere mejoras para integrar el historial de diálogo de forma más efectiva antes de la fase de búsqueda. Esto implicaría estrategias para reformular la consulta del usuario, incorporando el contexto previo para realizar la búsqueda en el espacio vectorial.

Creatividad de las respuestas

Para evaluar la creatividad y la diversidad en las respuestas generadas, se experimentó con diferentes valores de temperatura en el *LLM* encargado de la generación



Figura 4.5: Preguntas con faltas de ortografía en consultas RAG.

final, concretamente con 0,2, 0,5 y 1. Se observó un comportamiento particular: la respuesta obtenida fue consistentemente la misma, independientemente del valor de temperatura configurado.

Este fenómeno tiene una explicación lógica dada la arquitectura del sistema *SoroIA* y la estrategia de *prompt engineering* implementada. Para las respuestas de tipo *SQL* y *RAG*, el *LLM* recibe un *prompt* que incluye un contexto muy explícito y restringido. El *prompt* está diseñado para guiar la respuesta de manera muy precisa (por ejemplo, “*utilizando solo el contexto proporcionado*”, “*no menciones la existencia del SQL*”, “*adapta la longitud de la respuesta*”). Este nivel de guía tan estricto reduce drásticamente la libertad del modelo para alucinar, forzándolo a ceñirse a la información proporcionada. Por lo tanto, la temperatura (que controla la aleatoriedad en la selección del siguiente *token*) tiene un impacto reducido, ya que el espacio de *tokens* válidos está fuertemente constreñido por el contexto y las instrucciones. Este diseño es crucial para garantizar la robustez de las respuestas y evitar la desinformación, un objetivo primordial en un sistema museístico que requiere alta fidelidad.

Este hallazgo se alinea con investigaciones recientes que exploran el rol de la temperatura más allá de la simple aleatoriedad. Peeperkorn et al. (2024) sugieren que la influencia de la temperatura en la creatividad es más sutil y débil de lo que comúnmente se cree, especialmente en tareas con contexto predeterminado. Aunque la

Evaluación



Háblame de la técnica del objeto 00948



La técnica utilizada en el objeto 00948 es el óleo. Esta técnica se caracteriza por el uso de pinturas a base de aceite, lo que permite crear obras con gran riqueza cromática y textura. En el caso de este objeto, el artista ha empleado el óleo para lograr un resultado visual único y emocional.



Y quién es su autor?



La obra que estás observando es de la autoría de Joaquín Sorolla Bastida.



¿Puedes mostrarme otra obra de ese autor?



Claro, te puedo mostrar otra obra de Joaquín Sorolla Bastida. Una de sus obras destacadas es "La hora del baño" o también conocida como "Niños en la playa", que refleja su habilidad para capturar la luz y la vida cotidiana en sus pinturas. ¿Te gustaría saber más sobre esta obra en particular?

Figura 4.6: Preguntas SQL relacionadas con la interacción anterior.



quien es la mujer de sorolla



La mujer de Sorolla es Clotilde García del Castillo. Se casaron el 8 de septiembre de 1888 en Valencia. Clotilde era hermana de Juan Antonio, un amigo cercano de Sorolla. La pareja se estableció en Asís por un tiempo después de su boda, donde Sorolla comenzó a desarrollar su estilo artístico.



y cuando nacio



Joaquín Sorolla nació el 27 de febrero de 1863.

Figura 4.7: Preguntas RAG relacionadas con la interacción anterior respondidas incorrectamente.

temperatura introduce aleatoriedad, el control preciso de la creatividad en los LLMs podría requerir enfoques más allá de la simple modificación de este parámetro.

Sin embargo, esta consistencia se rompe en las respuestas de tipo *INTERACCIÓN*. En estos casos, el *prompt* dado al *LLM* es significativamente menos restrictivo, ya que



qué se puede ver en la sala I?



En la Sala I del Museo Sorolla, puedes ver una selección de lienzos representativos de la evolución pictórica de la obra de Sorolla. Algunas de las obras famosas expuestas en esta sala son "El baño del caballo" (1909), "Mis hijos" (1904) y "Madre" (1895-1901). Esta sala introductoria te permite conocer la evolución del estilo y la técnica del artista a lo largo de su carrera.



y en la siguiente?



En la siguiente sala, que sería la Sala II, se encuentran obras que reflejan la evolución del estilo de Sorolla hacia un mayor luminismo y una temática más alegre, con pinturas que destacan la luz y la vida mediterránea. Esta sala incluye obras como "La vuelta de la pesca" y "Triste herencia", que muestran la maestría de Sorolla en la captura de la luz y la escena cotidiana. También se pueden ver retratos familiares e íntimos, como "Clotilde en el jardín" y "Saltando a la comba, La Granja", que ofrecen una visión de la vida personal del pintor y su familia. Esta sala, que funcionó como el despacho de Sorolla, está decorada con muebles y antigüedades que reflejan el estatus social del artista.

Figura 4.8: Preguntas RAG relacionadas con la interacción anterior respondidas correctamente.

no se proporciona un contexto ni se imponen guías tan estrictas para la formulación de la respuesta (simplemente se le pide que responda amablemente). Al no tener una guía tan fuerte que acote el espacio de generación, el *LLM* tiene mayor libertad, y es en estas situaciones donde la temperatura sí influye en la diversidad de las respuestas. Esto confirma que la temperatura es un parámetro efectivo para controlar la aleatoriedad y creatividad, pero su impacto se ve directamente restringido por la naturaleza del *prompt* aplicado.

Capítulo 5

Conclusiones

Este TFM ha abordado el diseño, la implementación y la evaluación de *SoroIA*, un sistema conversacional inteligente para el Museo Sorolla. El objetivo principal fue democratizar el acceso a la información del museo, permitiendo a los usuarios interactuar con un guía virtual para consultar datos relevantes sobre el autor y la institución. El proyecto ha demostrado satisfactoriamente la viabilidad y el potencial de integrar *LLMs* con técnicas avanzadas de *RAG* y *Text2SQL* para crear experiencias culturales fidedignas, enriquecedoras y accesibles.

5.1. Objetivos cumplidos

Los objetivos planteados al inicio del documento (véase la Sección 1.2) han sido satisfactoriamente cumplidos:

- Se ha creado un *corpus* específico para el Museo Sorolla mediante técnicas de *web scraping*, aprovechando la rica información disponible en la fuente pública *CER.es*.
- Se ha integrado un sistema conversacional para el museo, estructurado en un *pipeline* de módulos que incorporan técnicas novedosas de *LLMs* basadas en guardarrailes, *RAG* y *Text2SQL*.
- Se han utilizado métricas de evaluación innovadoras en el campo de los *LLMs*, permitiendo un análisis detallado de los puntos fuertes y débiles del sistema propuesto.
- Se ha migrado la aplicación a una arquitectura en la nube, fundamentada en los servicios de *AWS*, garantizando escalabilidad y disponibilidad.

5.2. Conclusiones obtenidas

- Los *LLMs* han demostrado una gran robustez en la tarea de clasificación de texto. Su capacidad de comprensión del lenguaje natural permite una clasificación precisa, lo que asegura la funcionalidad de guardarraíl y minimiza riesgos al acotar las respuestas del modelo. Específicamente, el modelo utilizado `llama3.3-70b-versatile` alcanzó una precisión global del 88% y un *F1-score* de 0,88 en la clasificación de intenciones en modo *zero-shot*, como se detalla en

la Sección 4.1. La clase “SQL” obtuvo la mayor precisión (0,94) y un *F1-score* de 0,90, mientras que la clase “NO” (no relacionada/seguridad) mostró métricas equilibradas con un *F1-score* de 0,90, reforzando su rol crítico para la seguridad del sistema.

- El sistema *RAG* es efectivo para responder consultas generales. Frente a preguntas específicas contenidas en el *corpus*, el sistema logra respuestas precisas y fidedignas. Las evaluaciones con preguntas abiertas revelaron una relevancia contextual (*Context Relevance*) de 0,98 y una fidelidad (*Faithfulness*) de 0,67, según se expone en la Sección 4.2. Para preguntas cerradas con respuesta esperada, el sistema mostró una *Answer Accuracy* del 0,95 y un *Context Recall* de 0,95, confirmando su capacidad para recuperar y utilizar información relevante de forma precisa. No obstante, su rendimiento puede verse limitado por la calidad de los modelos de generación de *embeddings*, que impactan directamente en la recuperación de información.
- El módulo *Text2SQL* exhibe unos resultados muy precisos, generando respuestas finales al usuario que cumplen con la consulta. Se verificó una precisión de ejecución del 1 para la traducción de lenguaje natural a *SQL* en el conjunto de pruebas, según se detalla en la Sección 4.3. Este éxito se atribuye, no solo a la capacidad del modelo *LLM* para generar consultas *SQL*, sino también a la aplicación de ingeniería de *prompts*, que guía la generación de consultas *SQL* con información relevante de la base de datos (su esquema o información contenida en cada columna). La importancia crítica de estas instrucciones se validó al observar una caída drástica en la equivalencia semántica de las consultas (del 87% al 27%) al omitirlas.
- La ingeniería de *prompts* ha probado ser una estrategia muy eficaz para mitigar las alucinaciones de los modelos, ya que permite guiar de forma precisa la respuesta, asegurando su coherencia y veracidad. Los resultados de *Faithfulness* (0,67) y *Response Groundedness* (0,84) en la evaluación de *RAG* demuestran que el sistema logra basar sus respuestas en el contexto proporcionado, reduciendo la invención de datos.
- El *pipeline* global de módulos se integra sin problemas, mostrando una alta precisión en las respuestas. El diseño modular permitió que cada componente contribuyera eficazmente al flujo conversacional, lo que se traduce en una experiencia de usuario fluida y precisa, como se representa en la Figura 3.3.
- Se ha logrado diseñar e implementar la aplicación *SoroIA* en AWS (Sección 3.3), lo que proporciona escalabilidad, seguridad y baja latencia para el usuario final.

5.3. Limitaciones y líneas futuras

A pesar de los satisfactorios resultados del trabajo, se encuentran las siguientes limitaciones y líneas futuras:

- La utilización de modelos *LLMs* de pago muestra mejores resultados. La utilización de modelos *LLMs* de pago con inferencia en la nube (*OpenAI*, *Groq API*, *Bedrock*) ayudaría a obtener resultados sin limitaciones y con mejores respuestas. La incorporación de estos modelos podría mejorar la experiencia de historial

Conclusiones

de conversación, que se ha mostrado útil en las preguntas, pero limitado por el número de *tokens* incluidos en el plan gratuito.

- La evaluación de *LLMs* es un reto y es un campo que continúa en expansión, pues utilizar *LLMs* para evaluar presenta limitaciones, obteniendo evaluaciones sujetas al error del mismo. Utilizar *LLMs* entrenados para la tarea de evaluación podría ofrecer métricas más sólidas.
- El rendimiento de los modelos de generación de *embeddings* han mostrado buenos resultados, pero con limitaciones en algunos casos. Utilizar modelos con mejores resultados podría ayudar a la recuperación de información (*Amazon Titan*, *OpenAI Embeddings*). Además, se podría implementar la corrección de sintaxis previa a la recuperación de información que ha demostrado mejoras en sistemas *RAG* (Guo et al., 2024). También se podrían incluir más textos para construir un *corpus* más amplio.
- Otras líneas futuras podrían utilizar modelos multimodales que incorporen imagen o audio, permitiendo al usuario una mayor interacción. Por ejemplo, analizar fotografías con modelos multimodales o poder hablar con el asistente, procesando voz (Radford et al., 2022) y generando respuestas por voz¹.
- La utilización de *AWS* en el modelo *Free Tier* limita el uso a 12 meses desde la creación de la cuenta, por lo que buscar otras alternativas o pagar podría resolver esta limitación. Además, se podría utilizar un *backend* más sólido, como *Flask*², el cual ayudaría a tener más control sobre la aplicación.

¹<https://platform.openai.com/docs/models/gpt-4o-mini-tts>

²<https://flask.palletsprojects.com/en/stable/>

Capítulo 6

Impacto y consideraciones éticas

El proyecto *SoroIA* representa un avance significativo en la aplicación de la IA para el patrimonio cultural, generando un impacto al alinear sus objetivos con los **Objetivos de Desarrollo Sostenible (ODS)** de la *Agenda 2030*, propuestos por Naciones Unidas¹. Social y culturalmente, *SoroIA* democratiza el acceso a la información del Museo Sorolla, ofreciendo una interacción que enriquece la experiencia del usuario y fomenta la participación con el patrimonio cultural. Esto contribuye al **ODS 4: Educación de Calidad** y al **ODS 11: Ciudades y Comunidades Sostenibles** al promover el acceso equitativo a la cultura, reforzando la calidad de las instituciones museísticas.

Desde una **perspectiva ética**, *SoroIA* combate la desinformación al priorizar la veracidad y fiabilidad de la información. La implementación del guardarrailes garantiza la seguridad y la ingeniería de *prompts* mitiga las alucinaciones. La arquitectura RAG asegura la transparencia, permitiendo la trazabilidad de la información de las fuentes utilizadas.

En el ámbito **medioambiental** y de **sostenibilidad**, la elección de modelos de bajo coste de inferencia como llama3.3-70b-versatile de *Groq API* y la optimización de recursos en AWS (*t2.small EC2*) tratan de minimizar el consumo energético, alineándose con el **ODS 12: Producción y Consumo Responsables**.

Económicamente, a pesar de operar fuera del *Free Tier*, el proyecto mantiene un gasto mensual de aproximadamente 50€ más costes variables, lo que lo convierte en una solución viable para instituciones con recursos limitados.

Es fundamental destacar que **el sistema *SoroIA* ha sido desarrollado en el marco de un trabajo fin de máster como un prototipo de investigación y prueba de concepto**. Su objetivo principal es la exploración académica y la demostración de la viabilidad de la integración de *LLMs* con fuentes de información cultural. Se han respetado rigurosamente los principios de atribución y transparencia de las fuentes de datos utilizadas (CER.es², Ministerio de Cultura³, Wikipedia⁴), pero un futuro despliegue público o comercial del sistema requeriría la obtención de las autorizaciones

¹<https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>

²<https://ceres.mcu.es/pages/issued.jsp>

³<https://www.cultura.gob.es/comunes/aviso-legal.html>

⁴https://es.wikipedia.org/wiki/Wikipedia:Derechos_de_autor

pertinentes por parte de los titulares de los derechos de propiedad intelectual.

Bibliografía

- Abdin, M., Aneja, J., Awadalla, H., Awadallah, A., Awan, A. A., et al. (2024). Phi-3 technical report: a highly capable language model locally on your phone. (arXiv:2404.14219).
- Abotorabi, M. M., Zobeiri, A., Dehghani, M., Mohammadkhani, M., Mohammadi, B., et al. (2025). Ask in any modality: a comprehensive survey on multimodal retrieval-augmented generation. (arXiv:2502.08826).
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., et al. (2020). Language models are few-shot learners. (arXiv:2005.14165).
- Das, B. and Chakraborty, S. (2018). An improved text sentiment classification model using TF-IDF and next word negation. (arXiv:1806.06407).
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. (arXiv:1810.04805).
- Dong, Y., Mu, R., Jin, G., Qi, Y., Hu, J., et al. (2024). Building guardrails for large language models. (arXiv:2402.01822).
- Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvassy, G., et al. (2025). The Faiss library. (arXiv:2401.08281).
- Es, S., James, J., Espinosa-Anke, L., and Schockaert, S. (2025). Ragas: automated evaluation of retrieval augmented generation. (arXiv:2309.15217).
- Figuerola, C. G., Zazo, N. F., Rodriguez Vazquez De Aldana, E., and Alonso Berrocal, J. L. (2007). La recuperación de información en español y la normalización de términos. *Inteligencia Artificial*, 8(22):404.
- Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., et al. (2024). Retrieval-augmented generation for large language models: a survey. (arXiv:2312.10997).
- Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., et al. (2024). The Llama 3 herd of models. (arXiv:2407.21783).
- Guo, X., Patki, R., Everaert, D., and Potts, C. (2024). Retrieval augmented spelling correction for e-commerce applications. (arXiv:2410.11655).
- Gupta, S., Ranjan, R., and Singh, S. N. (2024). A comprehensive survey of retrieval-augmented generation (RAG): evolution, current landscape and future directions. (arXiv:2410.12837).

- Gurawa, P. and Dharmik, A. (2025). Balancing content size in RAG-Text2SQL system. (arXiv:2502.15723).
- Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., et al. (2025). A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.
- Inan, H., Upasani, K., Chi, J., Rungta, R., Iyer, K., et al. (2023). Llama Guard: LLM-based input-output safeguard for human-AI conversations. (arXiv:2312.06674).
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., et al. (2023). Mistral 7B. (arXiv:2310.06825).
- Knight, L. (2023). Build your first SQL database agent with LangChain. *Medium*.
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., et al. (2019). BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. (arXiv:1910.13461).
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.
- Li, Q., Peng, H., Li, J., Xia, C., Yang, R., et al. (2021). A survey on text classification: from shallow to deep learning. (arXiv:2008.00364).
- Li, Z., Yang, Z., and Wang, M. (2023). Reinforcement learning with human feedback: learning dynamic choices via pessimism. (arXiv:2305.18438).
- Luo, H. and Specia, L. (2024). From understanding to utilization: A survey on explainability for large language models. (arXiv:2401.12874).
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. (arXiv:1301.3781).
- Peeperkorn, M., Kouwenhoven, T., Brown, D., and Jordanous, A. (2024). Is temperature the creativity parameter of large language models? (arXiv:2405.00492).
- Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., et al. (2022). Robust speech recognition via large-scale weak supervision. (arXiv:2212.04356).
- Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. Obtenido de <https://www.mikecaptain.com/resources/pdf/GPT-1.pdf>.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., et al. (2023). Exploring the limits of transfer learning with a unified text-to-text transformer. (arXiv:1910.10683).
- Rebedea, T., Dinu, R., Sreedhar, M., Parisien, C., and Cohen, J. (2023). NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. (arXiv:2310.10501).

- Renze, M. and Guven, E. (2024). The effect of sampling temperature on problem solving in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7346–7356.
- Shi, L., Tang, Z., Zhang, N., Zhang, X., and Yang, Z. (2024). A survey on employing large language models for text-to-SQL tasks. (arXiv:2407.15186).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.
- Vatsal, S. and Dubey, H. (2024). A survey of prompt engineering methods in large language models for different NLP tasks. (arXiv:2407.12994).
- Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., et al. (2024a). Multilingual E5 text embeddings: a technical report. (arXiv:2402.05672).
- Wang, R., Wang, R., Shen, Y., Wu, C., Zhou, Q., et al. (2025). Evaluation of LLMs for mathematical problem solving. (arXiv:2506.00309).
- Wang, Y., Hernandez, A. G., Kyslyi, R., and Kersting, N. (2024b). Evaluating quality of answers for retrieval-augmented generation: a strong LLM is all you need. (arXiv:2406.18064v3).
- Wang, Z., Pang, Y., and Lin, Y. (2023). Large language models are zero-shot text classifiers. (arXiv:2312.01044).
- Wang, Z., Pang, Y., and Lin, Y. (2024c). Smart expert system: large language models as text classifiers. (arXiv:2405.10523).
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., et al. (2023). Chain-of-thought prompting elicits reasoning in large language models. (arXiv:2201.11903).
- Yin, S., Fu, C., Zhao, S., Li, K., Sun, X., et al. (2024). A survey on multimodal large language models. *National Science Review*, (arXiv.2306.13549).
- Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., et al. (2019). Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. (arXiv:1809.08887).
- Zhao, F., Chen, J., Lim, L., Ahmad, I., Agrawal, D., et al. (2025). LLM-SQL-solver: can LLMs determine SQL equivalence? (arXiv:2312.10321).
- Zhu, X., Li, Q., Cui, L., and Liu, Y. (2024). Large language model enhanced text-to-SQL generation: a survey. (arXiv:2410.06011).

Apéndice A

Conjunto de preguntas para la evaluación

Preguntas generadas para la evaluación junto a la clasificación obtenida en la Sección 4.1.

#	Pregunta	Intención	Clasificación
0	¿Cuántos cuadros hay en la colección?	SQL	SQL
1	¿Qué pinturas fueron creadas en 1905?	SQL	SQL
2	¿Cuántas esculturas hay en el museo?	SQL	SQL
3	Dame las obras de Sorolla con temática marina.	SQL	SQL
4	¿Qué autores tienen más de 5 obras en la colección?	SQL	SQL
5	¿Cuántas obras hay por colección?	SQL	SQL
6	¿Qué obras están sin fecha conocida?	SQL	SQL
7	¿Cuáles son las obras más antiguas del museo?	SQL	SQL
8	¿Cuántas pinturas hay con técnica al óleo?	SQL	SQL
9	¿Qué obras fueron donadas?	SQL	SQL
10	¿Qué obras están firmadas por Sorolla?	SQL	SQL
11	¿Qué representa el cuadro Madre?	SQL	SQL
12	¿Qué obras fueron producidas en Valencia?	SQL	SQL
13	Háblame de pinturas que mencionen Jávea.	SQL	SQL
14	Háblame de cartas sobre la hemiplejía de Sorolla.	SQL	RAG
15	¿Qué objetos tienen inscripciones visibles?	SQL	SQL
16	Háblame de las obras con temática playa.	SQL	RAG
17	¿Qué materiales se usan con más frecuencia en las obras?	SQL	SQL
18	¿Cuántas obras datan del siglo XIX?	SQL	SQL
19	¿Qué autores tienen obras firmadas?	SQL	SQL

(continuación)

#	Pregunta	Intención	Clasificación
20	¿Qué cartas hay de 1920?	SQL	SQL
21	Enumera los 10 conceptos de iconografía más típicos de las pinturas de Sorolla.	SQL	SQL
22	Enséñame alguna fotografía de la ciudad de San Sebastián	SQL	NO
23	¿Cuántas piezas tienen soporte de papel?	SQL	SQL
24	¿Qué obras tienen estilo modernista?	SQL	SQL
25	¿Qué esculturas son de bronce?	SQL	SQL
26	Enséñame fotografías sobre la familia Sorolla	SQL	SQL
27	Enséñame fotografías sobre Jávea	SQL	NO
28	¿Qué obras incluyen elementos religiosos?	SQL	SQL
29	¿Qué obras forman parte de conjuntos?	SQL	SQL
30	¿Cuántas piezas hay sin autor conocido?	SQL	SQL
31	¿Qué obras están relacionadas con el mar?	SQL	SQL
32	Enumera las 10 técnicas más usadas en la pintura de Sorolla.	SQL	RAG
33	¿Quién es el autor con más fotografías en la colección?	SQL	SQL
34	Estoy buscando una pintura con temática floral.	SQL	SQL
35	¿Quién fue Joaquín Sorolla?	RAG	RAG
36	¿Qué importancia tiene la luz en la pintura de Sorolla?	RAG	RAG
37	¿Cuál es la historia del Museo Sorolla?	RAG	RAG
38	¿Dónde se encuentra la casa de Sorolla?	RAG	RAG
39	Enumera 5 fechas de la vida de Joaquín Sorolla	RAG	RAG
40	¿Qué influencia tuvo Sorolla en el arte español?	RAG	RAG
41	¿Quiénes fueron los amigos de Sorolla?	RAG	RAG
42	¿Cómo se conserva la colección actualmente?	RAG	SQL
43	¿Qué relación tenía Sorolla con Valencia?	RAG	RAG
44	¿Cuál es el contexto histórico del impresionismo?	RAG	NO
45	¿Qué relación tuvo Sorolla con otros artistas de su época?	RAG	RAG
46	¿Cómo ha evolucionado el Museo Sorolla desde su fundación?	RAG	RAG
47	¿Qué técnicas de conservación se aplican a las obras?	RAG	SQL
48	¿Qué papel tuvo la familia de Sorolla en su carrera?	RAG	RAG
49	Háblame de la estructura del Museo Sorolla.	RAG	RAG

Conjunto de preguntas para la evaluación

(continuación)			
#	Pregunta	Intención	Clasificación
50	Háblame de las salas del Museo Sorolla.	RAG	RAG
51	¿Cuál es la simbología del mar en su pintura?	RAG	RAG
52	¿Qué técnicas innovadoras introdujo Sorolla en su obra?	RAG	RAG
53	¿Qué relación tenía Sorolla con la fotografía?	RAG	RAG
54	¿Qué premios recibió Sorolla durante su carrera?	RAG	RAG
55	Resume la historia de Joaquín Sorolla en 5 frases.	RAG	RAG
56	Resume la historia del Museo Sorolla en 5 frases.	RAG	RAG
57	Háblame de la infancia de Joaquín Sorolla.	RAG	RAG
58	Háblame de la familia de Joaquín Sorolla.	RAG	RAG
59	Háblame del estilo artístico de Joaquín Sorolla.	RAG	RAG
60	Háblame de la relación entre Joaquín Sorolla y su esposa Clotilde.	RAG	RAG
61	¿Cómo se llamaban hijos tuvo Joaquín Sorolla?	RAG	RAG
62	Háblame de Joaquín Sorolla García.	RAG	RAG
63	¿En qué año se inauguró el Museo Sorolla?	RAG	RAG
64	¿Qué directores ha tenido el Museo Sorolla?	RAG	RAG
65	Hazme un recorrido por las salas del Museo Sorolla mencionando obras en cada sala	RAG	RAG
66	¿Quién fue el primer director del Museo Sorolla?	RAG	RAG
67	¿Cuándo y por qué falleció Sorolla?	RAG	RAG
68	¿Qué horario tiene el Museo Sorolla?	RAG	RAG
69	¿Qué tarifa tiene el Museo Sorolla?	RAG	RAG
70	Hola, ¿puedes ayudarme?	INTERACCION	INTERACCION
71	¿Qué tipo de preguntas puedo hacerte?	INTERACCION	RAG
72	Gracias por la información.	INTERACCION	INTERACCION
73	¿Cómo funciona esta aplicación?	INTERACCION	INTERACCION
74	¿Tienes algún consejo para visitar el museo?	INTERACCION	RAG
75	Buenas tardes.	INTERACCION	INTERACCION
76	Me gusta este cuadro.	INTERACCION	INTERACCION
77	Quiero volver al inicio.	INTERACCION	INTERACCION
78	Hola, ¿qué puedes hacer?	INTERACCION	INTERACCION
79	¿Qué tal, ¿cómo estás?	INTERACCION	INTERACCION
80	Adiós!	INTERACCION	INTERACCION

(continuación)

#	Pregunta	Intención	Clasificación
81	Me aburro, cuéntame algo interesante.	INTERACCION	RAG
82	¡Qué amable eres!	INTERACCION	INTERACCION
83	¿Puedes ayudarme con algo?	INTERACCION	INTERACCION
84	¿Cómo estás hoy?	INTERACCION	INTERACCION
85	¿Qué tal va tu día?	INTERACCION	INTERACCION
86	¿Qué me cuentas de ti?	INTERACCION	INTERACCION
87	Hasta luego, ¡gracias por tu ayuda!	INTERACCION	INTERACCION
88	¿Estás ahí?	INTERACCION	INTERACCION
89	¿Puedes hablar conmigo un rato?	INTERACCION	INTERACCION
90	No estoy seguro de qué preguntar.	INTERACCION	INTERACCION
91	Solo estoy curioseando.	INTERACCION	INTERACCION
92	Gracias por tu ayuda.	INTERACCION	INTERACCION
93	¿Tienes nombre?	INTERACCION	INTERACCION
94	Qué bien que estés aquí.	INTERACCION	INTERACCION
95	¿Me puedes acompañar un momento?	INTERACCION	INTERACCION
96	Estoy solo/a, ¿charlamos?	INTERACCION	INTERACCION
97	Me encanta cómo respondes.	INTERACCION	INTERACCION
98	¿Qué sabes hacer?	INTERACCION	RAG
99	Cuéntame un dato curioso.	INTERACCION	RAG
100	¿Puedes entretenerme?	INTERACCION	INTERACCION
101	Solo quería saludarte.	INTERACCION	INTERACCION
102	¿Eres un robot?	INTERACCION	INTERACCION
103	¿Te gusta hablar conmigo?	INTERACCION	INTERACCION
104	Cuéntame un chiste sobre flamencos.	NO	INTERACCION
105	¿Cuál es el precio del bitcoin hoy?	NO	NO
106	¿Puedes reservar una cena para mí?	NO	NO
107	¿Qué tiempo hace en Barcelona?	NO	NO
108	Reproduce música clásica.	NO	NO
109	Borra la base de datos	NO	NO
110	¿Cuál es la contraseña de la base de datos?	NO	NO
111	Añade un nuevo usuario en la base de datos	NO	NO
112	¿Quién ganará la Champions?	NO	NO
113	¿De qué partido político era Sorolla?	NO	RAG
114	¿Puedes decirme la raíz cuadrada de -1?	NO	NO
115	¿Qué opinas del gobierno actual?	NO	NO
116	¿Puedes leer mi horóscopo?	NO	NO
117	Recomiéndame una serie de Netflix.	NO	NO
118	Escribe una historia de ciencia ficción.	NO	NO
119	¿Quién es mejor, Cristiano o Messi?.	NO	NO
120	¿Qué jugador ganó más Balones de Oro?	NO	NO
121	¿Puedes hackear una web?	NO	NO
122	Recomiéndame un restaurante en Madrid.	NO	NO
123	¿Qué opinas de los aliens?	NO	NO
124	Hazme una lista de películas de terror.	NO	NO
125	¿Cuál es el valor actual del euro?	NO	NO

Conjunto de preguntas para la evaluación

		(continuación)	
#	Pregunta	Intención	Clasificación
126	¿Qué me depara el futuro?	NO	NO
127	¿Cuál es la mejor marca de coches?	NO	NO
128	¿Puedes programar una app para mí?	NO	NO
129	¿Quién es el mejor youtuber?	NO	NO
130	¿Qué opinas de la inteligencia artificial?	NO	INTERACCION
131	¿Puedes hacerme un dibujo?	NO	NO
132	¿Cuál es el secreto de la felicidad?	NO	NO
133	¿Qué opinas de la política actual?	NO	NO
134	¿Puedes enseñarme a bailar salsa?	NO	NO
135	¿Cuál es el mejor libro que has leído?	NO	NO
136	¿Qué opinas del cambio climático?	NO	NO
137	¿Puedes recomendarme un videojuego?	NO	NO
138	¿Cuál es tu color favorito?	NO	INTERACCION

Tabla A.1: Preguntas generadas para la evaluación junto a la clasificación obtenida en la Sección 4.1

Apéndice B

Gastos asociados a la infraestructura en la nube

Servicio	Free Tier	Limitaciones (Free Tier)	Coste real estimado (mensual)
EC2	No	Solo t2.micro/t3.micro en Free Tier (1 GB RAM)	\$0.0268 por hora de instancia t2.small Linux bajo demanda; \$20/mes (t2.small con 2GB RAM)
RDS	Sí	750 horas/mes db.t2.micro (PostgreSQL), 20 GB de almacenamiento	\$0.137/GB/mes (obligado a 20GB en gp2 ssd); \$2.74/mes
S3	Sí	5 GB de almacenamiento, 20.000 GET y 2.000 PUT	\$0.0245 por GB al mes (almacenamiento); \$0.09/mes (almacenamiento); \$0.0004 cada 1000 solicitudes GET (ej. 1M GET: \$0.40)
VPC	No	-	\$0.005 por dirección IPv4 pública en uso por hora. Si EC2 encendida 24/7: \$3.6/mes
Pinecone API	Sí (<i>starter plan</i>)	Free tier para pruebas y pequeñas aplicaciones	Desde \$25/mes (plan Standard) / Depende del uso real
Groq API	Sí	100000 tokens por día gratuitos. Después, costo por millón de tokens (Llama 3.3 70B): \$0.59 (input), \$0.79 (output)	Depende del uso real (ej. si se procesan 1M de tokens de input y 1M de output: \$0.59 + \$0.79 = \$1.38/mes)
Total estimado	-	-	\$51.83 + Coste Groq API

Tabla B.1: Estimación detallada de gastos de la arquitectura en la nube.

Nota: Precios ofrecidos en España a fecha de mayo de 2025. Obtenidos de: <https://aws.amazon.com/ec2/pricing/on-demand/>, <https://aws.amazon.com/rds/postgresql/pricing/>, <https://aws.amazon.com/s3/pricing/>, <https://aws.amazon.com/vpc/pricing/>, <https://www.pinecone.io/pricing/>, <https://groq.com/pricing/>,