



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Diseño e implementación de una plataforma de análisis y
consulta inteligente de datos de tráfico marítimo*

Grado en Ingeniería Mecánica

ALUMNO: Carlos de la Puente Álvarez

DIRECTORES: Belén Barragáns Martínez
Pablo Sendín Raña

CURSO ACADÉMICO: 2025-2026

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Diseño e implementación de una plataforma de análisis y
consulta inteligente de datos de tráfico marítimo*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

El análisis del tráfico marítimo mediante datos AIS se ha convertido en una herramienta clave para comprender y conocer el entorno marítimo y apoyar la toma de decisiones en ámbitos operativos y de seguridad. Sin embargo, el gran volumen de información generado por estos sistemas dificulta su explotación mediante herramientas tradicionales de consulta y análisis.

En este contexto, el presente TFG aborda el diseño e implementación de una plataforma de análisis y consulta inteligente de datos de tráfico marítimo basada en técnicas de inteligencia artificial. En particular, se emplean modelos de lenguaje de gran tamaño (LLM) combinados con técnicas de recuperación de información (RAG) con el objetivo de permitir la interacción con los datos AIS mediante consultas en lenguaje natural.

Para ello, se ha desarrollado una arquitectura capaz de interpretar las consultas del usuario y transformarlas en operaciones analíticas sobre los datos, generando respuestas estructuradas que faciliten su interpretación. El sistema integra módulos de recuperación y procesamiento de datos, junto con mecanismos de generación de respuestas, permitiendo realizar consultas sobre los datos en cuestión. Los resultados evidencian la viabilidad del enfoque para mejorar el acceso y la explotación de grandes volúmenes de datos AIS y facilitar su análisis en el ámbito marítimo, incluso por usuarios menos expertos.

PALABRAS CLAVE

AIS, *Large Language Models* (LLMs), *Retrieval-Augmented Generation* (RAG), Ollama, entorno marítimo.

AGRADECIMIENTOS

En primer lugar, quería agradecer a todo el personal de la Escuela Naval Militar los cinco años de formación y esfuerzo invertidos en mí y en mis compañeros, que nos ha ayudado a crecer tanto personal como profesionalmente.

En especial, agradecer a mis dos tutores, Belén y Pablo, por su incansable dedicación y tiempo invertido en guiarme durante este trabajo, siempre con alegría y predisposición para ayudarme, durante estos meses, a sortear todos los obstáculos que nos hemos ido encontrando por el camino.

Tampoco puedo olvidarme de mi familia: de mis padres, por inculcarme los valores que me han llevado a seguir este camino en mi vida y a escoger la profesión más bonita del mundo, y del resto de mi familia, especialmente de mis hermanas, que con tanta paciencia han estado siempre a mi lado apoyándome. Papá, mamá, Vito, Merchi, Almu, gracias. Quiero agradecer también a mi tío Manen el haber despertado siempre en mí ese lado más “científico” que, sin duda, me ha ayudado a superar estos años en la Escuela.

Por último, me gustaría dar las gracias a mis compañeros de la 426-156, que han sido responsables de que estos años hayan sido tan especiales. Sin ellos, esta etapa no habría sido lo mismo. A mis compañeros de camareta y de la 116, que han hecho de este año un año especialmente inolvidable.

A todos ellos, gracias por vuestro apoyo y compañía siempre. Este trabajo es también, en parte, fruto de vuestro apoyo.

CONTENIDO

Contenido	5
Índice de Figuras	9
Índice de Tablas	11
1 Introducción y objetivos.....	13
1.1 Contexto y motivación.....	13
1.2 Objetivos.....	14
1.3 Estructura de la memoria	14
2 Estado del arte	17
2.1 El análisis de datos en entornos complejos.....	17
2.1.1 El problema actual del <i>Big Data</i>	17
2.1.2 Aplicación a la defensa y análisis operacional	18
2.2 <i>Large Language Models</i> - LLMs.....	19
2.2.1 Origen y evolución de los modelos de lenguaje.....	19
2.2.2 Concepto de LLM	20
2.2.3 Tipos de LLMs	21
2.3 Agentes RAG – <i>Retrieval-Augmented Generation</i>	21
2.3.1 Integración de RAG con modelos de lenguaje.....	21
2.3.2 Concepto y funcionamiento de RAG	22
2.3.3 Componentes principales de una arquitectura RAG	23
2.3.4 Bases de datos vectoriales y mecanismos de recuperación	24
2.3.5 Evolución de los agentes RAG.....	24
2.3.6 Tipos de agentes RAG.....	25
2.4 Datos AIS.....	25
2.4.1 Origen y finalidad del sistema AIS	25
2.4.2 Funcionamiento general e información proporcionada.....	26
2.4.3 Volumen y características de los datos AIS	27
2.4.4 Retos asociados al análisis de datos AIS.....	28
2.4.5 Conexión de los datos AIS con el enfoque del proyecto.....	28
2.5 Estudio de trabajos relacionados	29
2.5.1 AIS-LLM.....	29
2.5.2 <i>Using LLMs for Analyzing AIS Data</i>	29
2.5.3 Democratización de los datos con Inteligencia Artificial: un asistente virtual basado en LLM	31
2.5.4 Técnicas de lenguaje natural aplicadas a consultas documentales.....	31

3 Desarrollo del TFG	33
3.1 Objetivo y principios de diseño del sistema	33
3.2 Preparación del entorno de ejecución y despliegue inicial de modelos LLM	34
3.2.1 Selección y despliegue del entorno de modelos de lenguaje.....	34
3.2.2 Implementación del entorno LLM local y carga inicial de datos.....	35
3.3 Aproximación inicial de la plataforma	36
3.4 Justificación de la arquitectura	37
3.5 Primera propuesta de arquitectura	38
3.5.1 Arquitectura general del sistema	38
3.5.2 Gestión y almacenamiento de los datos AIS	40
3.5.3 Enfoque inicial de consulta: traducción directa de lenguaje natural a SQL.....	40
3.5.1 Interfaz gráfica de consulta e interpretación de los datos	41
3.5.2 Limitaciones detectadas en el enfoque inicial	41
3.6 Segunda propuesta de arquitectura	42
3.6.1 Introducción de un módulo RAG	42
3.6.2 Integración del RAG en el flujo de generación de consultas	43
3.6.3 Flujo completo de una consulta en el sistema	44
3.6.4 Problemas detectados en la generación inicial de SQL.....	45
3.7 Tercera propuesta de arquitectura.....	46
3.7.1 Separación explícita entre datos dinámicos y estáticos.....	46
3.7.2 Incorporación de validación estructural de consultas generadas.....	46
3.7.3 Introducción de un formato intermedio estructurado (JSON).....	47
3.7.3.1 Justificación del uso de un formato intermedio	47
3.7.3.2 Estructura del plan de ejecución intermedio	48
3.7.3.3 Proceso de normalización y verificación del plan.....	48
3.7.4 Optimización del procesamiento de datos: <i>Polars</i> + <i>Parquet</i>	49
3.7.4.1 Sustitución del motor de análisis: de DuckDB a <i>Polars</i>	49
3.7.4.2 Adopción de Parquet como formato de almacenamiento principal	50
3.7.5 Arquitectura del proceso de consulta	50
3.7.6 Evaluación de la arquitectura propuesta.....	52
3.7.6.1 Resultados de las primeras pruebas de consulta.....	52
3.7.6.2 Identificación del cuello de botella	52
3.8 Arquitectura definitiva del sistema	53
3.8.1 Enfoque determinista de consulta.....	53
3.8.2 Arquitectura mixta de consulta.....	55
3.8.3 Principio general de la arquitectura.....	55

3.8.4 Clasificación de consultas	56
3.8.5 Papel del LLM y sistema RAG	57
3.8.6 Las herramientas deterministas y su organización por familias.....	57
3.8.7 Planes JSON y diferencias entre ambas rutas	58
3.8.8 Validación del plan y diferencias según la ruta.....	58
3.8.9 Función de los documentos en la arquitectura	59
3.8.10 Gestión temporal basada en el <i>dataset</i>	59
3.8.11 Ejecución determinista, control de calidad y presentación final	59
4 Resultados obtenidos y validación.....	61
4.1 Interfaz de usuario	61
4.2 Metodología de la evaluación.....	61
4.2.1 Clasificación de las consultas.....	61
4.2.2 <i>Dataset</i> utilizado.....	62
4.2.3 Métricas de evaluación.....	63
4.3 Consultas de atributos.....	63
4.3.1 Descripción de las consultas.....	63
4.3.2 Resultados	64
4.3.3 Análisis.....	64
4.4 Consultas de listados.....	65
4.4.1 Descripción de las consultas.....	65
4.4.2 Resultados	65
4.4.3 Análisis.....	66
4.5 Consultas de fusión de datos.....	66
4.5.1 Descripción de las consultas.....	66
4.5.2 Resultados	67
4.5.3 Análisis.....	68
4.6 Consultas analíticas	68
4.6.1 Descripción de las consultas.....	68
4.6.2 Resultados	69
4.6.3 Análisis.....	70
4.7 Comparación entre tipos de consulta.....	70
5 Conclusiones y líneas futuras.....	73
5.1 Conclusiones.....	73
5.2 Líneas futuras.....	74
5.2.1 Optimización del rendimiento y escalabilidad del sistema	74

5.2.2 Ampliación de las capacidades de análisis.....	74
5.2.3 Mejora del uso de modelos de lenguaje especializados	74
5.2.4 Integración con otras fuentes de información marítima.....	74
6 Bibliografía	75
Anexo I: Implicaciones sociales, económicas y ambientales	79
Anexo II: Reflexiones éticas y sociales	81
Anexo III: Explicación general del código	83
A3.1. Detalle de las bibliotecas empleadas	83
A3.2 Bloque de configuración	83
A3.3 Columnas y <i>helpers</i> básicos.....	84
A3.4 Extracción de parámetros desde la pregunta.....	85
A3.5 Acceso a datos y caché	86
A3.6 Clasificación	87
A3.7 Herramientas deterministas: trabajo real sobre los datos.....	88
A3.8 Redacción de la respuesta	89
Anexo IV: Glosario de Siglas	91
Anexo V: Guía de instalación y configuración del entorno de ejecución.....	93

ÍNDICE DE FIGURAS

Figura 2-1 Evolución de la cantidad de datos generada anualmente (tomada de [1])	17
Figura 2-2 Operaciones multidominio en la actualidad (tomada de [3])	18
Figura 2-3 Evolución de los modelos de lenguaje (generada con IA Gen [7]).....	20
Figura 2-4 Esquema simplificado de una arquitectura RAG (tomada de [19])	23
Figura 2-5 Densidad del tráfico marítimo (tomada de [24]).....	27
Figura 3-1 Captura del log cuando se sube un documento de 6Mb con 100,000 filas.	37
Figura 3-2 Representación del flujo de una consulta en la primera propuesta de arquitectura	39
Figura 3-3 Representación del flujo de una consulta en la segunda propuesta de arquitectura.....	45
Figura 3-4 Ejemplo de plan JSON	48
Figura 3-5 Representación del flujo de una consulta en la tercera propuesta de arquitectura	51
Figura 3-6 Ejemplos de las consultas.....	52
Figura 3-7 Ejemplos de las consultas con tiempos desglosados	53
Figura 3-8 Ejemplo de consulta sobre los datos con <i>Polars</i> + <i>Parquet</i>	53
Figura 3-9 Ejemplos de las consultas con tiempos desglosados	54
Figura 3-10 Representación del flujo de una consulta en la propuesta definitiva de arquitectura	55
Figura 3-11 Comparativa de tiempos para cada arquitectura	56
Figura 4-1 Presentación de la interfaz gráfica	61
Figura 4-2 Ejemplos de consultas de atributos	64
Figura 4-3 Gráfica de resultados de las consultas de atributos	64
Figura 4-4 Ejemplos de consultas de listados	65
Figura 4-5 Gráfica de resultados de las consultas de listados.....	66
Figura 4-6 Ejemplos de consultas de fusión de datos	67
Figura 4-7 Gráfica de resultados de las consultas de fusión de datos.....	67
Figura 4-8 Ejemplos de consultas de análisis	69
Figura 4-9 Gráfica de resultados de las consultas de análisis	69
Figura 4-10 Gráfica de comparación de tiempos	71
Figura 4-11 Gráficas de desglose de los tiempos empleados.....	72
Figura 4-12 Gráfica de comparación de puntuación de calidad de las respuestas.....	72
Figura A3-1 Listado de las bibliotecas empleadas.....	83
Figura A3-2 Configuración	83
Figura A3-3 Detalle de las columnas definidas	84
Figura A3-4 <i>Helpers</i> básicos.....	84
Figura A3-5 Extractores de parámetros	85

Figura A3-6 Acceso a los datos y caché	86
Figura A3-7 Recuperación de contexto para clasificación	87
Figura A3-8 Validación de la ruta escogida.....	87
Figura A3-9 Ejemplo del código de una herramienta	88
Figura A3-10 Ejemplo del código de una llamada al modelo para la generación de una respuesta.....	89

ÍNDICE DE TABLAS

Tabla 2-1 Datos transmitidos en mensajes AIS (tomada de [23])	26
Tabla 3-1 Características de la máquina virtual	35
Tabla 4-1 Rúbrica de evaluación de las consultas	63
Tabla A3-1 Ejemplos de los extractores	85
Tabla A3-2 Ejemplos de las funciones	86
Tabla A3-3 Otras herramientas utilizadas	88
Tabla A3-4 Otras funciones utilizadas	89

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contexto y motivación

El transporte marítimo constituye uno de los pilares fundamentales del comercio internacional y de la economía global. Una parte muy significativa del intercambio de mercancías a nivel mundial se realiza por vía marítima, lo que implica la existencia de un elevado número de buques operando simultáneamente en rutas comerciales, zonas portuarias y áreas de interés estratégico. En este contexto, disponer de herramientas que permitan comprender y analizar el comportamiento del tráfico marítimo resulta esencial tanto para la seguridad de la navegación como para la gestión eficiente de los espacios marítimos.

Uno de los principales sistemas utilizados para monitorizar el tráfico marítimo es el Sistema de Identificación Automático (*Automatic Identification System*, AIS). Este sistema permite que los buques transmitan de forma periódica información relevante sobre su identidad, posición, rumbo, velocidad y otras características de navegación. Gracias a estos mensajes es posible obtener una visión detallada del movimiento de los buques y del estado del tráfico marítimo en determinadas zonas.

A pesar de los retos asociados a su análisis, los datos AIS ofrecen un enorme potencial para el estudio y la comprensión del tráfico marítimo. Su carácter continuo, su amplia cobertura geográfica y la variedad de información que proporcionan permiten obtener una visión detallada de la actividad en el dominio marítimo, tanto a corto como a largo plazo. Estas características convierten al AIS en una fuente de datos especialmente valiosa para aplicaciones de monitorización, vigilancia y análisis operacional.

En los últimos años, los datos AIS han sido utilizados en numerosos ámbitos, como la seguridad marítima, la detección de comportamientos anómalos, la gestión del tráfico o el apoyo a operaciones de búsqueda y rescate. A partir del análisis de trayectorias, velocidades y patrones de movimiento, es posible identificar situaciones de interés, evaluar tendencias y mejorar la toma de decisiones en entornos complejos. Esta capacidad de análisis se ve reforzada cuando los datos AIS se combinan con información adicional, como características de los buques, zonas marítimas de interés o datos históricos de tráfico.

No obstante, aprovechar plenamente este potencial requiere ir más allá de los enfoques de análisis tradicionales. El volumen y la complejidad de los datos AIS hacen necesario disponer de herramientas capaces de facilitar su explotación, permitiendo formular consultas flexibles y obtener resultados comprensibles para el usuario. En este sentido, el desarrollo de plataformas que permitan interactuar con los datos de forma más intuitiva surge como una solución ante la dificultad y el tiempo que requiere actualmente el análisis de estos conjuntos de información.

En paralelo, los avances recientes en el campo de la inteligencia artificial y, en particular, en los modelos de lenguaje de gran tamaño (*Large Language Models*, LLM) han abierto nuevas posibilidades

para la interacción con sistemas informáticos mediante lenguaje natural. Estos modelos son capaces de interpretar consultas formuladas por un usuario y generar respuestas coherentes basadas en la información disponible. Sin embargo, también presentan ciertas limitaciones, como la posibilidad de generar respuestas incorrectas o basadas en información no verificada.

Para abordar estas limitaciones han surgido arquitecturas como *Retrieval-Augmented Generation* (RAG), que combinan la capacidad generativa de los modelos de lenguaje con mecanismos de recuperación de información desde fuentes externas. De este modo, el modelo puede apoyarse en datos específicos para generar respuestas más precisas y contextualizadas.

En este contexto, el presente Trabajo Fin de Grado se centra en el diseño e implementación de una plataforma que permita analizar datos de tráfico marítimo procedentes del sistema AIS mediante consultas en lenguaje natural, facilitando así el acceso a la información y permitiendo realizar análisis de forma más flexible e intuitiva.

1.2 Objetivos

El objetivo principal de este Trabajo Fin de Grado es diseñar e implementar una plataforma capaz de analizar datos de tráfico marítimo procedentes del sistema AIS mediante consultas formuladas en lenguaje natural.

Para alcanzar este objetivo general se plantean los siguientes objetivos específicos:

- Diseñar una arquitectura que permita almacenar y procesar datos AIS de forma eficiente.
- Implementar un sistema que interprete consultas realizadas por el usuario en lenguaje natural, que integre modelos de lenguaje LLM con mecanismos de recuperación RAG y análisis de datos que permitan generar respuestas basadas en la información real disponible.
- Desarrollar una interfaz que facilite la interacción entre el usuario y el sistema.
- Evaluar el funcionamiento de la plataforma mediante diferentes consultas analizando la calidad de las respuestas obtenidas además de los tiempos de procesamiento.

1.3 Estructura de la memoria

Tras hacer referencia al marco en el que se encuadra este trabajo, contextualizándolo y definiendo los objetivos que se esperan alcanzar, se pasa a detallar la estructura que seguirá la presente memoria. El documento está dividido en varios capítulos a los que se añade, al final, la bibliografía que recoge las fuentes empleadas durante el desarrollo del trabajo.

Esta memoria se organiza como sigue:

- Capítulo 1: Introducción y objetivos. En este primer capítulo se presenta el contexto del trabajo, la motivación que justifica su desarrollo y los objetivos que se pretenden alcanzar. Asimismo, se describe la estructura general de la memoria.
- Capítulo 2: Estado del arte. En este capítulo se realiza una revisión del marco teórico y tecnológico en el que se encuadra el trabajo. Para ello se abordan los siguientes aspectos principales:
 - Procesamiento de lenguaje natural y modelos de lenguaje: introducción a los conceptos básicos del procesamiento de lenguaje natural y al funcionamiento de los modelos de lenguaje de gran tamaño (LLM).
 - Sistema AIS (*Automatic Identification System*): fundamentos del sistema, tipo de información transmitida por los buques y su utilidad para la monitorización y análisis del tráfico marítimo.
 - Arquitecturas de recuperación de información: estudio de las arquitecturas RAG, que combinan modelos de lenguaje con fuentes de datos externas para mejorar la precisión y fiabilidad de las respuestas generadas.

- Revisión de trabajos relacionados con este TFG.
- Capítulo 3: Desarrollo del TFG. En este capítulo se describe el proceso de diseño e implementación de la plataforma desarrollada a lo largo del trabajo. Se detallan las decisiones técnicas adoptadas y los distintos componentes que forman parte de la solución propuesta. En particular, se abordan los siguientes aspectos:
 - Arquitectura general del sistema: descripción de la estructura de la plataforma y de la interacción entre sus diferentes componentes.
 - Gestión y almacenamiento de los datos AIS: tratamiento de los datos utilizados en el sistema, incluyendo su organización, almacenamiento y preparación para su análisis.
 - Procesamiento y análisis de consultas: mecanismos utilizados para interpretar las consultas realizadas por el usuario en lenguaje natural y transformarlas en operaciones de análisis sobre los datos disponibles.
 - Integración con modelos de lenguaje: utilización de modelos de lenguaje para facilitar la interacción con el sistema y generar respuestas comprensibles para el usuario.
 - Flujo completo de funcionamiento del sistema: descripción del proceso que sigue una consulta desde que es introducida por el usuario hasta la generación de la respuesta final.
- Capítulo 4: Resultados obtenidos y validación. En este capítulo se presentan los resultados obtenidos tras la implementación del sistema. Se analizan diferentes consultas realizadas sobre los datos AIS con el objetivo de evaluar el comportamiento de la plataforma y su capacidad para realizar análisis sobre la información disponible.
- Capítulo 5: Conclusiones y líneas futuras. En este último capítulo se recogen las conclusiones derivadas del desarrollo del trabajo y se plantean posibles mejoras o líneas de desarrollo futuro que podrían ampliar o perfeccionar la solución propuesta.
- Se completa con una serie de anexos que incluye lo siguiente:
 - Implicaciones sociales, económicas y ambientales.
 - Reflexiones éticas y sociales.
 - Una explicación de los bloques principales del código.
 - Glosario de siglas.
 - Guía de instalación y configuración del entorno de ejecución.

2 ESTADO DEL ARTE

En este apartado se presenta el estado del arte relacionado con el diseño de plataformas de análisis y consulta inteligente de datos de tráfico marítimo. Para ello, se revisan los principales conceptos, tecnologías y enfoques relevantes, incluyendo los retos asociados al análisis de grandes volúmenes de datos, el uso de modelos de lenguaje de gran escala, las arquitecturas basadas en *Retrieval-Augmented Generation* y las características específicas de los datos AIS. Finalmente, se analizan distintos trabajos relacionados con el objetivo de contextualizar el presente Trabajo Fin de Grado y situar su propuesta dentro del panorama actual de investigación y desarrollo en este ámbito.

2.1 El análisis de datos en entornos complejos

2.1.1 El problema actual del Big Data

En los últimos años, el desarrollo de sistemas digitales ha provocado un crecimiento muy significativo en la cantidad de datos que se generan y almacenan de forma continua (véase Figura 2-1). Actividades cotidianas como el uso de aplicaciones informáticas, servicios en línea o sistemas de monitorización digital producen grandes volúmenes de información que se almacenan a un ritmo cada vez mayor. Como consecuencia de este rápido crecimiento de los datos, surge el término *Big Data* para referirse a conjuntos de datos de gran tamaño caracterizados por la notable complejidad de su tratamiento. Estas características suponen un desafío importante para los sistemas clásicos de análisis de datos, que no fueron concebidos para operar de forma eficiente con información producida a esta escala ni en entornos dinámicos.



Figura 2-1 Evolución de la cantidad de datos generada anualmente (tomada de [1])

El análisis de *Big Data* persigue transformar grandes cantidades de datos en información útil que permita identificar patrones, extraer conocimiento y apoyar la toma de decisiones. Sin embargo, a medida que aumenta el volumen y la diversidad de los datos, también lo hace la dificultad de procesarlos, interpretarlos y explotarlos de manera eficiente. Organizaciones tecnológicas como IBM [2] destacan que el verdadero reto del *Big Data* no reside únicamente en la capacidad de almacenamiento o procesamiento, sino en la posibilidad de analizar los datos de forma ágil para obtener valor real a partir de ellos, como bien dictan las 4 V's del *Big Data* – volumen, velocidad, variedad y veracidad.

Tradicionalmente, el análisis de grandes volúmenes de datos se ha abordado mediante sistemas de bases de datos relacionales, lenguajes de consulta estructurados y herramientas de visualización estáticas. Aunque estos enfoques han demostrado ser eficaces para tareas bien definidas y conjuntos de datos acotados, encuentran limitaciones cuando se aplican a datos masivos, dinámicos y altamente interconectados. La formulación de consultas complejas suele requerir conocimientos técnicos avanzados, y la interpretación de los resultados depende en gran medida de la experiencia del analista.

Además, en contextos donde la toma de decisiones debe llevarse a cabo en un corto plazo de tiempo, la dependencia de procesos manuales o de herramientas poco flexibles supone una limitación importante. El acceso a la información suele requerir la consulta de distintos sistemas y la combinación de datos procedentes de múltiples fuentes, lo que dificulta la obtención de una visión clara y coherente del problema. Como resultado, el proceso de convertir los datos disponibles en información útil se vuelve más lento, disminuyendo la capacidad de reaccionar de forma eficaz ante situaciones cambiantes.

En este contexto, el principal desafío del *Big Data* no reside únicamente en la gestión técnica de grandes cantidades de información, sino en la dificultad de interactuar con los datos de forma eficiente, flexible e intuitiva. Esta problemática ha impulsado la necesidad de nuevas herramientas que faciliten el acceso al conocimiento contenido en los datos y permitan explotar su potencial de manera más directa y comprensible para los usuarios.

2.1.2 Aplicación a la defensa y análisis operacional

Las limitaciones asociadas al análisis tradicional de *Big Data* se agravan de forma significativa en ámbitos caracterizados por un alto grado de complejidad e incertidumbre, como es el ámbito de la defensa y el análisis operacional. En estos entornos, especialmente hoy en día debido al multidominio (ilustrado en la Figura 2-2), la toma de decisiones depende de la capacidad para integrar grandes volúmenes de datos heterogéneos, analizarlos en plazos reducidos y extraer conclusiones que permitan anticipar escenarios y evaluar distintas alternativas de actuación.

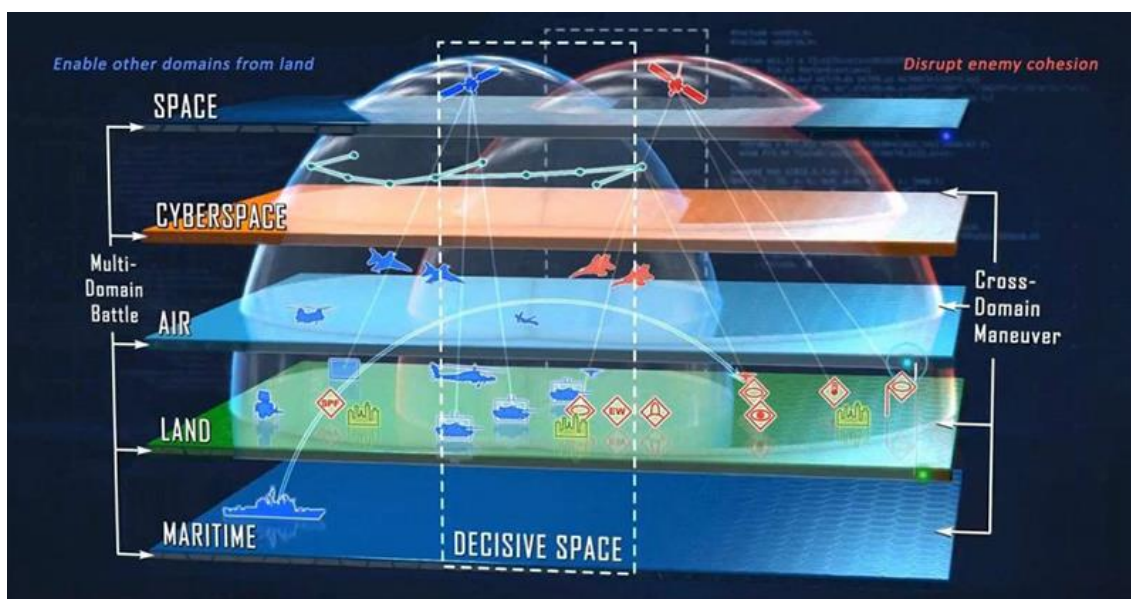


Figura 2-2 Operaciones multidominio en la actualidad (tomada de [3])

El análisis operacional en el ámbito de la defensa se centra en apoyar la toma de decisiones a partir del análisis sistemático de datos y del estudio de escenarios complejos. Trabajos presentados en el contexto de la *17th NATO Operations Research and Analysis Conference* [4] ponen en manifiesto el valor de explotar grandes volúmenes de datos en tiempo real para mejorar la comprensión del entorno marítimo y detectar comportamientos anómalos.

En este tipo de estudios se destaca la necesidad de contar con herramientas que permitan combinar distintas fuentes de información, analizar los datos de forma flexible y facilitar la comprensión de situaciones operacionales complejas. Este enfoque resulta especialmente relevante en entornos donde la información evoluciona constantemente y las decisiones deben ajustarse de manera continua a cambios en el escenario operativo.

Sin embargo, trasladar este tipo de análisis a sistemas operativos reales presenta importantes dificultades cuando se emplean herramientas tradicionales de análisis de datos. La necesidad de trabajar con información en tiempo real, combinar datos procedentes de distintas fuentes y adaptar el análisis a situaciones cambiantes hace que los enfoques habituales resulten poco adecuados. En la práctica, gran parte del proceso de análisis requiere una elevada intervención manual y un conocimiento profundo de las herramientas disponibles, lo que limita la agilidad del sistema y dificulta su aplicación en entornos operacionales exigentes.

Además, en el ámbito de la defensa, el análisis no se limita a la obtención de datos o resultados aislados, sino que requiere contextualizar la información, identificar patrones y considerar posibles escenarios futuros. Este tipo de análisis resulta especialmente complejo de abordar con herramientas convencionales, que suelen estar orientadas a consultas predefinidas y no facilitan una interacción dinámica con los datos.

Estas limitaciones ponen de manifiesto que los enfoques tradicionales de análisis de datos resultan insuficientes para dar respuesta a las necesidades actuales en entornos operacionales complejos. Las limitaciones propias de muchas herramientas convencionales, junto con la dificultad de interactuar de forma flexible con grandes volúmenes de información, restringen la capacidad de explotar plenamente los datos disponibles. Ante este escenario, en los últimos años han comenzado a explorarse nuevas aproximaciones basadas en modelos de lenguaje de gran tamaño (*Large Language Models* – LLMs), que permiten plantear una forma diferente de interactuar con los datos y abrir nuevas posibilidades para el análisis y el apoyo a la toma de decisiones.

2.2 Large Language Models - LLMs

2.2.1 Origen y evolución de los modelos de lenguaje

Los modelos de lenguaje tienen como objetivo principal modelar la probabilidad de aparición de una secuencia de palabras, permitiendo así tareas como la predicción del siguiente término o la generación de texto coherente. Los primeros enfoques en este ámbito se basaban en modelos estadísticos, como los modelos *n-gram* [5], que utilizaban la frecuencia de aparición de secuencias cortas de palabras para estimar probabilidades. Aunque estos métodos resultaban sencillos de implementar, presentaban importantes limitaciones a la hora de captar relaciones complejas dentro del texto y de manejar contextos amplios.

Posteriormente, la introducción de redes neuronales permitió superar parcialmente estas limitaciones. En particular, las redes neuronales recurrentes y sus extensiones, como las *Long Short-Term Memory* (memoria corto-largo plazo), ofrecieron la capacidad de modelar secuencias de longitud variable y capturar dependencias temporales más complejas. Sin embargo, estos modelos presentaban dificultades relacionadas con la capacidad del entrenamiento, la pérdida de información en secuencias largas y la imposibilidad de realizar simultáneamente varias operaciones con los datos de forma efectiva.

Un avance decisivo en la evolución de los modelos de lenguaje se produjo con la introducción de la arquitectura *Transformer*. En [6] se propuso un enfoque basado en mecanismos de atención que permiten analizar de forma conjunta las relaciones entre todas las palabras de una secuencia, sin recurrir a estructuras recurrentes. Gracias al uso de *self-attention*, el modelo puede asignar distinta relevancia a cada palabra en función del contexto, captando relaciones semánticas independientemente de la posición que ocupen dentro del texto. Además, al eliminar el procesamiento secuencial, el *Transformer* permite tratar de forma paralela las distintas secuencias, mejorando de forma significativa la eficiencia del entrenamiento y facilitando el escalado a modelos de mayor tamaño.

La arquitectura *Transformer* sentó las bases para el desarrollo de modelos de lenguaje de mayor tamaño y capacidad, permitiendo un escalado eficiente tanto en términos de datos como de parámetros. A partir de este punto, los modelos de lenguaje experimentaron una evolución significativa, dando lugar a los actuales *Large Language Models*, capaces de aprender representaciones lingüísticas complejas y de abordar tareas de comprensión y generación de texto con un alto nivel de precisión. En la Figura 2-3, se representa gráficamente esta evolución.



Figura 2-3 Evolución de los modelos de lenguaje (generada con IA Gen [7])

2.2.2 Concepto de LLM

A partir de la arquitectura *Transformer*, los modelos de lenguaje experimentaron un proceso de escalado significativo que dio lugar a los LLM (grandes modelos de lenguaje). De forma general, un LLM puede definirse como un modelo de lenguaje basado en redes neuronales profundas, entrenado con grandes cantidades de texto y caracterizado por un elevado número de parámetros. El término *large* hace referencia principalmente al tamaño del modelo y al volumen de datos utilizados durante su entrenamiento, responsables de su capacidad para aprender y generalizar.

Los LLMs aprenden a representar el lenguaje mediante el ajuste de un gran número de parámetros durante su entrenamiento, del orden de miles de millones. Para ello, el modelo se entrena con grandes volúmenes de texto y aprende a predecir palabras o fragmentos a partir del contexto que las rodea. Usando este proceso, el modelo adquiere la capacidad de entender el significado general de los textos, reconocer formas habituales de expresión y manejar la estructura de las frases de forma coherente, lo que le permite generalizar su comportamiento a situaciones no vistas previamente.

Una característica fundamental de los LLMs es su versatilidad, comúnmente conocida como su capacidad para “razonar”. A diferencia de enfoques tradicionales diseñados para tareas específicas, estos modelos pueden adaptarse a una amplia variedad de aplicaciones relacionadas con el procesamiento del lenguaje natural, como la generación de texto, la respuesta a preguntas, la clasificación de información o la síntesis de contenidos. Esta capacidad de generalización ha convertido a los LLMs en una tecnología de propósito general para la interacción con información textual.

Además, los LLMs no se limitan a reproducir información de forma literal, sino que son capaces de generar respuestas coherentes y contextualizadas a partir de entradas formuladas en lenguaje natural. Esta propiedad resulta especialmente relevante en escenarios de análisis de datos, donde el modelo puede actuar como una interfaz capaz de interpretar consultas complejas y expresar los resultados de forma comprensible. No obstante, estas capacidades dependen en gran medida de los datos y del proceso de entrenamiento, lo que introduce también ciertas limitaciones que deben ser consideradas en aplicaciones reales.

2.2.3 Tipos de LLMs

En la actualidad existe una amplia variedad de LLMs que pueden emplearse en sistemas de análisis y consulta de datos. Una primera distinción es separar entre modelos propietarios y modelos de código abierto. Un modelo propietario es aquel cuyo uso suele estar controlado por una empresa y se ofrece normalmente a través de servicios externos, sin acceso completo al modelo ni a sus parámetros. Por el contrario, un modelo de código abierto es aquel que puede descargarse y ejecutarse en infraestructuras propias, lo que permite un mayor control sobre su despliegue y sobre la información procesada.

A partir de esta diferencia, los modelos presentan variaciones importantes en cuanto a su forma de acceso, su rendimiento y las posibilidades de integración en sistemas reales, lo que hace necesario analizar sus características antes de seleccionar el más adecuado para un caso de uso concreto.

Entre los modelos propietarios más conocidos se encuentran los desarrollados por OpenAI [8], como los pertenecientes a la familia GPT (por ejemplo, GPT-3.5, GPT-4 o GPT-5) [9]. Estos modelos destacan por ofrecer resultados de alta calidad en tareas de comprensión y generación de lenguaje natural, así como por su buen comportamiento en una amplia variedad de aplicaciones.

No obstante, su uso se basa en el acceso a través de servicios externos, lo que implica depender de infraestructuras en la nube, asumir costes asociados y ceder el control sobre los datos que se procesan. Estas características pueden suponer una limitación en aplicaciones donde se trabaja con información sensible o donde se desea mantener el sistema completamente bajo control.

Otros modelos propietarios relevantes son los desarrollados por Google, como PaLM 2 [10] o Gemini [11], o los desarrollados por Microsoft como Copilot [12]. Estos presentan capacidades avanzadas similares y están estrechamente ligados a los servicios ofrecidos por la propia compañía.

Como alternativa a estos enfoques, han surgido en los últimos años diversos modelos de lenguaje de código abierto que permiten un mayor grado de flexibilidad. Entre ellos se encuentran modelos como BLOOM [13], Mistral [14] o Falcon [15]. Estos modelos pueden desplegarse en infraestructuras propias y facilitan la experimentación sin depender de servicios externos. No obstante, algunos de ellos presentan limitaciones prácticas, como una menor madurez del conjunto de herramientas disponibles o mayores dificultades a la hora de integrarlos en arquitecturas más complejas.

Dentro del conjunto de modelos de código abierto, la familia LLaMA [16], desarrollada por Meta [17], destaca por ofrecer un equilibrio adecuado entre calidad de los resultados y requisitos técnicos. Los modelos LLaMA cuentan con una amplia adopción en el ámbito académico, una documentación clara y una buena compatibilidad con bibliotecas y *frameworks* utilizados habitualmente en el desarrollo de aplicaciones basadas en LLMs. Además, como se pueden ejecutar en local, los datos no se envían a servicios externos, lo que resulta especialmente relevante en proyectos orientados al análisis de información sensible.

Teniendo en cuenta los objetivos de este proyecto, como la necesidad de controlar todo el proceso de desarrollo y evaluación del sistema, se considera que el uso de un modelo de la familia LLaMA es la opción más adecuada. Esta elección permite desarrollar una plataforma de análisis y consulta inteligente basada en LLMs de forma autónoma, manteniendo un buen equilibrio entre capacidades del modelo, facilidad de integración y control sobre los datos utilizados.

2.3 Agentes RAG – *Retrieval-Augmented Generation*

2.3.1 Integración de RAG con modelos de lenguaje

Tal y como se ha expuesto en los apartados anteriores, los LLMs han supuesto un avance significativo en la forma de interactuar con la información, especialmente gracias a su capacidad para comprender y generar lenguaje natural. Sin embargo, cuando se aplican a sistemas reales de análisis de

datos, el uso de LLMs de manera aislada presenta limitaciones importantes que restringen su aplicabilidad práctica.

Una de las principales limitaciones de los LLMs es que su conocimiento está ligado al proceso de entrenamiento y no se actualiza de forma dinámica. Esto implica que el modelo no tiene acceso directo a información reciente ni a datos específicos de un sistema concreto, lo que resulta especialmente problemático en contextos donde los datos evolucionan continuamente o donde se trabaja con información propia de una organización. Además, los LLMs no están diseñados para consultar bases de datos ni para operar directamente sobre grandes volúmenes de datos estructurados, lo que puede dar lugar a respuestas incompletas o poco fiables.

En el ámbito del análisis de grandes volúmenes de datos, estas limitaciones adquieren una mayor relevancia. La necesidad de basar el análisis en información actualizada y procedente de múltiples fuentes hace que un modelo puramente generativo no sea suficiente para garantizar resultados útiles y verificables. En este contexto, es necesario complementar las capacidades del LLM con mecanismos que permitan acceder de forma directa a los datos disponibles y utilizarlos como base para la generación de respuestas.

La integración de técnicas de *Retrieval-Augmented Generation* (RAG) surge como una respuesta práctica a las limitaciones de los modelos de lenguaje cuando se aplican a sistemas reales de análisis de datos. Al incorporar mecanismos que permiten recuperar información relevante en el momento de la consulta, RAG combina la capacidad de los LLMs para interpretar y generar lenguaje natural con el acceso directo a datos externos y actualizados. Este enfoque permite que las respuestas generadas se apoyen en información concreta y verificable, reduciendo la dependencia exclusiva del conocimiento adquirido durante el entrenamiento y facilitando su uso en escenarios donde la consulta y el análisis de datos deben realizarse de forma interactiva y fiable.

2.3.2 Concepto y funcionamiento de RAG

RAG es una arquitectura que combina modelos de lenguaje con sistemas de recuperación de información con el objetivo de mejorar la calidad y fiabilidad de las respuestas generadas. A diferencia de un enfoque puramente generativo, en el que el modelo produce respuestas a partir del conocimiento aprendido durante el entrenamiento, RAG introduce un paso intermedio en el que se recupera información relevante desde fuentes externas antes de generar la respuesta final [18].

El funcionamiento se basa en lo siguiente. El usuario formula una consulta en lenguaje natural, que es procesada para identificar información relevante en un conjunto de datos previamente almacenados, bien sea información histórica o datos en tiempo real. A partir de esta búsqueda, el sistema selecciona la información más relacionada con la consulta y se la proporciona al modelo de lenguaje como contexto adicional, siendo el modelo de lenguaje el que genera la respuesta.

Una vez incorporada esta información recuperada, el modelo de lenguaje genera la respuesta teniendo en cuenta tanto la consulta original como los datos proporcionados por el sistema de recuperación. De este modo, la generación no se basa únicamente en el conocimiento interno del modelo, sino que se apoya en información concreta y contextualizada. Este enfoque permite obtener respuestas más precisas, coherentes y alineadas con los datos disponibles.

Una característica clave de RAG es que separa claramente dos funciones dentro del sistema: por un lado, la recuperación de información relevante, y por otro, la generación de la respuesta en lenguaje natural. Esta separación facilita la actualización de los datos utilizados por el sistema sin necesidad de reentrenar el modelo de lenguaje, lo que resulta especialmente ventajoso en entornos donde la información cambia con frecuencia o donde se trabaja con datos propios de una organización.

En el contexto del análisis de grandes volúmenes de datos, RAG permite transformar conjuntos de información complejos en respuestas comprensibles para el usuario, manteniendo una conexión directa con los datos originales. De esta forma, se combina la flexibilidad del lenguaje natural con la solidez de

un análisis basado en información real, sentando las bases para el desarrollo de sistemas de consulta y análisis más avanzados.

2.3.3 Componentes principales de una arquitectura RAG

Una arquitectura basada RAG se compone de varios elementos que trabajan de forma coordinada para proporcionar respuestas basadas en datos reales. La separación en componentes distintos permite diseñar sistemas flexibles y adaptables a distintos tipos de aplicaciones y fuentes de información. Una versión simplificada de esta arquitectura se muestra en Figura 2-4.

El primer componente es el modelo de lenguaje, que actúa como núcleo del sistema y es el encargado de interpretar la consulta en lenguaje natural y generar la respuesta final. El LLM aporta la capacidad de comprensión del contexto, la generación de texto coherente y la presentación de la información de forma accesible para el usuario. No obstante, su funcionamiento depende del contexto que recibe, por lo que necesita apoyarse en información externa para ofrecer respuestas precisas y actualizadas.

El segundo componente es el sistema de recuperación de información, cuya función es localizar los datos más relevantes para una consulta determinada. Este sistema se encarga de buscar, dentro de un conjunto de datos previamente almacenados, aquellos fragmentos de información que guardan mayor relación con la pregunta planteada. La calidad de esta recuperación es un factor clave en el rendimiento global del sistema, ya que de esa recuperación depende el contexto que se proporciona al modelo de lenguaje.

Un tercer elemento fundamental son las fuentes de datos, que constituyen la base de conocimiento sobre la que opera el sistema RAG. Estas fuentes pueden incluir documentos de texto, bases de datos estructuradas, registros históricos o información generada en tiempo real. La arquitectura RAG permite que estas fuentes se actualicen de forma independiente, lo que ayuda a que la información más reciente sea utilizada por el modelo de lenguaje.

Por último, la arquitectura incorpora un elemento encargado de coordinar el funcionamiento del sistema en su conjunto. Este componente se ocupa de organizar los distintos pasos del proceso: desde que se recibe la consulta del usuario, pasando por la búsqueda de información relevante, hasta la generación de la respuesta final. Su función principal es asegurar que cada una de estas etapas se ejecute en el orden adecuado y que la información fluya correctamente entre los distintos módulos, garantizando así el funcionamiento claro del sistema.

La interacción entre estos componentes permite que una arquitectura RAG combine las capacidades de los modelos de lenguaje con el acceso directo a datos externos, sentando las bases para el desarrollo de sistemas de consulta y análisis adaptados a aplicaciones reales.

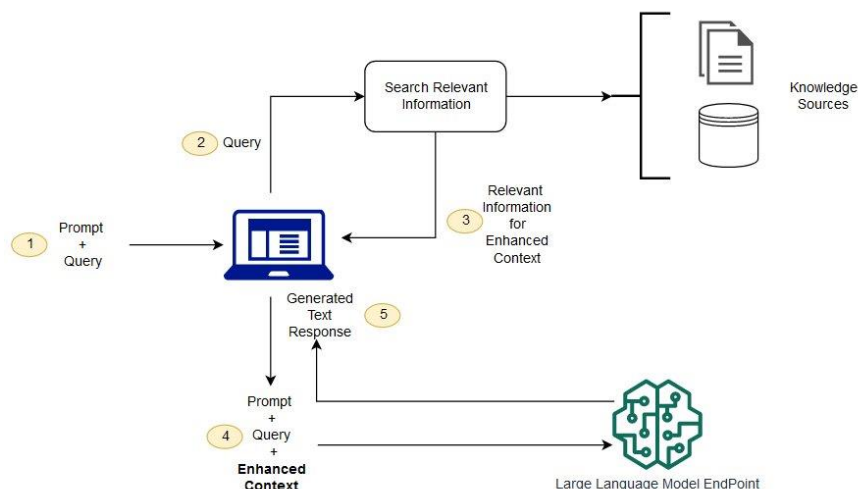


Figura 2-4 Esquema simplificado de una arquitectura RAG (tomada de [19])

2.3.4 Bases de datos vectoriales y mecanismos de recuperación

Para que un sistema RAG pueda recuperar información relevante a partir de una consulta en lenguaje natural, es necesario disponer de mecanismos de búsqueda que vayan más allá de las técnicas tradicionales basadas en coincidencias exactas de palabras. Las bases de datos vectoriales desempeñan un papel fundamental, ya que permiten representar la información de forma que se tenga en cuenta su significado y no únicamente su forma textual.

El funcionamiento de estas bases de datos se basa en la transformación de textos en vectores numéricos, también conocidos como *embeddings*. Estos vectores identifican el significado general del contenido, de manera que textos con significados parecidos quedan representados por vectores cercanos entre sí en el espacio vectorial. Tanto las consultas del usuario como los documentos almacenados se convierten a esta representación, lo que permite comparar su similitud de forma eficiente.

A diferencia de los sistemas de búsqueda tradicionales, donde los resultados dependen de palabras concretas, la búsqueda vectorial permite recuperar información relacionada, aunque no se utilicen exactamente las mismas palabras. Esto es especialmente útil en sistemas de análisis y consulta de datos, donde una misma idea puede redactarse de muchas formas y con terminología muy concreta.

El proceso de recuperación en una arquitectura RAG suele consistir en calcular la similitud entre el vector asociado a la consulta y los vectores de los datos almacenados, seleccionando los más parecidos. Los fragmentos recuperados se utilizan como contexto para el modelo de lenguaje, que genera la respuesta final apoyándose en esta información. De este modo, la base de datos vectorial actúa como un puente entre los datos originales y el modelo generativo.

En el contexto del análisis de grandes volúmenes de datos, el uso de bases de datos vectoriales permite trabajar con conjuntos de información extensos y complejos de forma eficiente, facilitando la búsqueda y mejorando la calidad de la información recuperada. Este enfoque resulta especialmente adecuado para sistemas interactivos basados en lenguaje natural, ya que ofrece una forma flexible y firme de acceder a los datos almacenados.

2.3.5 Evolución de los agentes RAG

Los primeros enfoques basados en RAG seguían un esquema relativamente sencillo. Ante una consulta, el sistema recuperaba información relevante y la proporcionaba al modelo de lenguaje para que generara una respuesta. Este enfoque, conocido habitualmente como RAG clásico, resultaba eficaz para tareas de consulta directa, donde una única recuperación de información era suficiente para responder a la pregunta planteada.

Sin embargo, en escenarios más complejos, como el análisis en profundidad de datos o el apoyo a la toma de decisiones, esto puede resultar limitado. Muchas consultas requieren encadenar varios pasos, reformular preguntas intermedias o acceder a distintas fuentes de información antes de generar una respuesta final. En estos casos, un único ciclo de recuperación y generación no siempre es suficiente para dar toda la información necesaria.

Para dar respuesta a estas limitaciones surgen los denominados agentes RAG, que incorporan capacidades de razonamiento iterativo y toma de decisiones. En lugar de llevar a cabo solo un proceso de recuperación, un agente RAG puede decidir qué información necesita, realizar múltiples consultas y combinar los resultados obtenidos antes de generar la respuesta final. De este modo, el modelo de lenguaje no se limita a generar texto, sino que actúa como un componente activo dentro del sistema.

Los agentes RAG permiten realizar procesos más cercanos al razonamiento humano, donde la obtención de conclusiones se basa en la exploración progresiva de la información disponible. Esto resulta especialmente beneficioso para entornos dinámicos y con grandes volúmenes de datos, ya que facilita un análisis más profundo y contextualizado, a diferencia de los sistemas RAG tradicionales.

2.3.6 Tipos de agentes RAG

A pesar de las ventajas que presentan los agentes RAG, su uso también introduce una serie de limitaciones que deben ser tenidas en cuenta. El hecho de que el sistema esté formado por distintos elementos que deben trabajar de manera coordinada incrementa la complejidad del sistema y exige un diseño cuidadoso para garantizar un funcionamiento adecuado. Además, el estado del sistema depende en gran medida de la calidad de los datos usados y de cómo se enlazan los distintos pasos del proceso, lo que puede influir tanto en la coherencia de las respuestas como en los tiempos de respuesta.

En la práctica, el desarrollo de agentes RAG suele apoyarse en bibliotecas y marcos de desarrollo que simplifican la construcción de este tipo de sistemas. Entre los más utilizados se encuentran LangChain y LlamaIndex, que ofrecen soluciones diferentes para conectar modelos de lenguaje con fuentes de datos externas. LangChain [20] destaca por permitir definir flujos de ejecución complejos en los que el modelo de lenguaje puede interactuar con distintas herramientas, como motores de búsqueda, bases de datos, funciones de procesamiento de texto o llamadas a APIs externas. Esta capacidad resulta útil en aplicaciones muy personalizadas, pero también puede implicar una mayor complejidad a la hora de configurar el sistema, especialmente en proyectos donde el objetivo principal es el análisis de datos y no la construcción de flujos lógicos complejos.

Por otro lado, LlamaIndex [21] está orientado sobre todo a simplificar la conexión entre modelos de lenguaje y conjuntos de datos, proporcionando mecanismos claros para la recuperación y contextualización de la información. Este enfoque resulta especialmente adecuado para proyectos centrados en la consulta y el análisis de grandes volúmenes de datos, ya que reduce la necesidad de definir manualmente flujos complejos y permite centrarse en la lógica del análisis. Además, su diseño facilita una integración directa con modelos de lenguaje de la familia LLaMA, lo que favorece la coherencia tecnológica del sistema y simplifica su despliegue en entornos controlados.

Para este proyecto en concreto, se considera que el uso de LlamaIndex es la opción más adecuada para la implementación de agentes RAG. Esta elección permite aprovechar las capacidades del modelo LLaMA seleccionado, limitar la complejidad del sistema y centrar el desarrollo en el diseño y evaluación de la arquitectura propuesta.

2.4 Datos AIS

2.4.1 Origen y finalidad del sistema AIS

El AIS es un sistema de comunicación diseñado para mejorar la seguridad de la navegación marítima y facilitar la supervisión del tráfico en zonas costeras y rutas marítimas congestionadas. Su origen se remonta a la necesidad de disponer de un mecanismo estandarizado que permitiera a los buques intercambiar información básica de forma automática, reduciendo el riesgo de colisiones y mejorando el conocimiento de la situación tanto a bordo como desde tierra.

La implantación del AIS fue impulsada por la IMO (*International Maritime Organization*) [22], que estableció su uso obligatorio como parte del marco normativo internacional en materia de seguridad marítima. En particular, el sistema se incorporó a las regulaciones del Convenio SOLAS (*Safety of Life at Sea*), lo que supuso su adopción progresiva por parte de los buques mercantes de mayor tamaño y de aquellos dedicados al transporte de pasajeros. Esta obligatoriedad permitió garantizar un uso amplio del sistema a escala internacional.

Desde su inicio, el AIS fue pensado como una herramienta orientada principalmente a la seguridad y a la prevención de accidentes, proporcionando información básica como la identidad del buque, su posición, rumbo y velocidad. Estos datos permiten a los navegantes y a los centros de control marítimo conocer la situación del tráfico en tiempo real y anticipar posibles situaciones de riesgo. Con el tiempo, el uso del AIS se ha extendido más allá de su finalidad inicial, convirtiéndose en una fuente de

información de gran valor para la vigilancia marítima, la gestión del tráfico y el análisis de la actividad en el entorno marítimo.

En la actualidad, el AIS constituye uno de los pilares fundamentales para la obtención de una visión global del tráfico marítimo, ya que ofrece un flujo continuo de información generada directamente por los propios buques. Esta característica ha favorecido su adopción en múltiples ámbitos, tanto civiles como gubernamentales, y ha sentado las bases para su utilización en aplicaciones avanzadas de análisis y apoyo a la toma de decisiones.

2.4.2 Funcionamiento general e información proporcionada

El sistema AIS funciona mediante la transmisión periódica de mensajes digitales a través de radiofrecuencia, que son emitidos de forma automática por los buques equipados con un transpondedor AIS. Estos mensajes pueden ser recibidos tanto por otros buques cercanos como por estaciones terrestres y satélites, lo que permite disponer de información del tráfico marítimo a diferentes escalas, desde zonas costeras hasta áreas oceánicas extensas.

Los mensajes AIS contienen distintos tipos de información que pueden agruparse en varias categorías. En primer lugar, incluyen datos de identificación, como el identificador del buque o su nombre, que permiten distinguir de forma precisa a cada embarcación. En segundo lugar, se transmiten datos dinámicos, entre los que se encuentran la posición geográfica, el rumbo y la velocidad, obtenidos normalmente a partir de sistemas de posicionamiento por satélite. Esta información se actualiza con una frecuencia elevada, especialmente cuando el buque se encuentra en movimiento.

Además, el AIS puede transmitir datos estáticos y de viaje, como el tipo de buque, sus dimensiones o el puerto de destino, como se puede ver en la Tabla 2-1. Aunque este tipo de información no se actualiza con la misma frecuencia que los datos dinámicos, resulta fundamental para contextualizar el comportamiento de una embarcación y para diferenciar distintos tipos de tráfico marítimo. La combinación de estos datos permite construir una descripción relativamente completa del estado y la actividad de cada buque.

Información dinámica	Información estática	Información de viaje
Posición	MMSI/Nº IMO	Calado
Fecha y Hora (GMT)	Call sign y nombre del barco	Mercancía peligrosa
Rumbo	Eslora y manga	Destino
Velocidad y rumbo sobre fondo	Tipo de barco	Tiempo de llegada estimado
Estado de navegación (p.e. fondeado)	Localización y posicionamiento de sus sistemas	Mensajes relacionados con la seguridad
Velocidad de caída		

Tabla 2-1 Datos transmitidos en mensajes AIS (tomada de [23])

La frecuencia de transmisión de los mensajes AIS depende de factores como la velocidad del buque o su maniobrabilidad. En situaciones normales, un buque en movimiento puede emitir varios mensajes por minuto, lo que da lugar a un flujo continuo de información. Como consecuencia, el AIS genera grandes volúmenes de datos de forma constante, especialmente en zonas con alta densidad de tráfico marítimo.

En conjunto, el funcionamiento del AIS da lugar a una fuente de datos rica y detallada sobre la actividad marítima, caracterizada por su carácter continuo y su cobertura global. Estas características

hacen que los datos AIS resulten especialmente valiosos para tareas de monitorización y análisis, pero también introducen desafíos importantes en términos de gestión, procesamiento y explotación de la información.

2.4.3 Volumen y características de los datos AIS

El uso generalizado del sistema AIS a nivel mundial ha dado lugar a la generación de un volumen de datos muy elevado, que crece de forma continua con el tiempo. Cada buque equipado con AIS transmite mensajes de manera periódica, y en zonas con alta densidad de tráfico marítimo pueden coexistir miles de embarcaciones emitiendo información de forma simultánea. Como resultado, el AIS se ha convertido en una de las principales fuentes de datos para el análisis del tráfico marítimo a escala global.

Una de las características más relevantes de los datos AIS es su alta frecuencia de generación. Dependiendo de la velocidad y del estado del buque, los mensajes pueden emitirse cada pocos segundos, lo que produce flujos de datos continuos y de gran intensidad. Esta característica es especialmente notable en áreas costeras, estrechos y puertos, donde el número de mensajes recibidos por unidad de tiempo puede alcanzar cifras muy elevadas, como se puede observar en Figura 2-5.

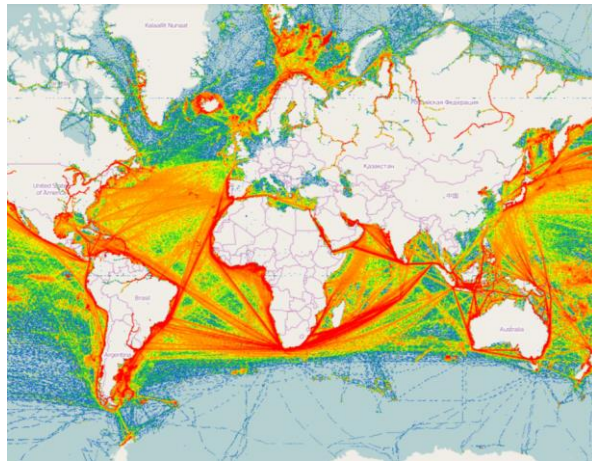


Figura 2-5 Densidad del tráfico marítimo (tomada de [24])

Además del volumen, los datos AIS presentan una serie de particularidades que incrementan la complejidad de su tratamiento. Entre ellas se encuentran la presencia de mensajes duplicados, datos incompletos o inconsistentes, y errores derivados tanto de fallos técnicos como de la introducción manual de información. Asimismo, no todos los campos transmitidos tienen el mismo grado de fiabilidad, lo que obliga a aplicar procesos de filtrado y validación antes de poder realizar un análisis riguroso.

Otra característica importante de los datos AIS es la variedad en la forma en que se generan y se distribuyen a lo largo del tiempo y del espacio. Los mensajes se producen de manera continua y cubren áreas geográficas muy amplias, desde zonas portuarias con alta concentración de tráfico hasta áreas de alta mar. Esto da lugar a conjuntos de datos que combinan información histórica con datos en tiempo real, lo que resulta especialmente interesante para el análisis, pero también plantea retos adicionales en términos de almacenamiento y procesamiento.

El volumen y las características de los datos AIS hacen que su análisis encaje plenamente dentro del ámbito del *Big Data*. La combinación de grandes cantidades de información, generación continua de datos y necesidad de tratamiento en tiempo real convierte al AIS en un caso representativo de los desafíos actuales en el análisis de datos de gran tamaño. Estas circunstancias justifican la necesidad de herramientas y enfoques avanzados que permitan explotar de forma eficiente todo el potencial de esta fuente de información.

2.4.4 Retos asociados al análisis de datos AIS

A pesar del gran valor informativo de los datos AIS, su análisis plantea una serie de retos importantes que dificultan su interpretación mediante las herramientas de análisis habituales. Estos retos no están relacionados únicamente con el volumen de datos, sino también con la calidad de la información, su estructura y la forma en que se genera y distribuye a lo largo del tiempo.

Uno de los principales retos en el análisis de datos AIS es la fiabilidad de la información disponible. Aunque el sistema proporciona un gran volumen de datos, estos pueden incluir valores erróneos, inconsistentes o incompletos, derivados tanto de fallos técnicos como de la introducción manual de ciertos campos por parte de la tripulación. Además, la calidad de la información no es igual entre distintos buques, lo que obliga a aplicar procesos adicionales de revisión y corrección antes de poder extraer conclusiones fiables.

Otro reto relevante es la gestión del volumen y la velocidad de los datos. El flujo continuo de mensajes, especialmente en zonas con alta densidad de tráfico marítimo, exige sistemas capaces de almacenar y procesar grandes cantidades de información en tiempo real o casi en tiempo real. Las soluciones basadas en consultas estáticas [25] o en el análisis *off-line* en profundidad resultan insuficientes cuando se requiere una supervisión continua del tráfico o una detección temprana de comportamientos relevantes.

Asimismo, el análisis de datos AIS requiere una adecuada contextualización de la información. Los valores de posición, rumbo o velocidad adquieren significado únicamente cuando se interpretan en relación con otros factores, como el tipo de buque, la zona geográfica, el comportamiento histórico o la presencia de infraestructuras marítimas cercanas. Esta necesidad de combinar múltiples fuentes de información incrementa la complejidad del análisis y dificulta su automatización mediante herramientas convencionales.

Por último, el carácter exploratorio de muchas tareas basadas en datos AIS constituye un reto adicional. En numerosos casos, el objetivo no es responder a una consulta predefinida, sino analizar patrones de comportamiento, identificar situaciones anómalas o formular hipótesis a partir de los datos disponibles. Este tipo de análisis resulta especialmente complejo de abordar con sistemas rígidos, ya que requiere una interacción flexible y dinámica con la información.

Estos retos ponen de manifiesto que, aunque los datos AIS ofrecen un enorme potencial para el análisis del tráfico marítimo, su explotación efectiva requiere herramientas y plataformas capaces de manejar grandes volúmenes de información, adaptarse a datos en tiempo real y facilitar una interacción más intuitiva con los resultados del análisis. Esta necesidad constituye uno de los principales puntos de partida para el enfoque propuesto en este Trabajo Fin de Grado.

2.4.5 Conexión de los datos AIS con el enfoque del proyecto

Las características y el potencial de los datos AIS muestran su relevancia como fuente de información para el análisis del tráfico marítimo. Sin embargo, el verdadero valor de estos datos no reside únicamente en el volumen de información disponible o en el detalle de los parámetros que proporcionan, sino en su capacidad para ser transformados en conocimiento útil que apoye la toma de decisiones en entornos operativos complejos.

En ámbitos como la vigilancia marítima, el control del tráfico o el análisis de patrones de navegación resulta necesario interpretar grandes volúmenes de información generados de forma continua por múltiples buques y sensores. Los datos AIS, por su carácter dinámico y su amplia cobertura, permiten fundamentar estas decisiones en información objetiva y actualizada [26]. No obstante, el volumen y la complejidad de los datos dificultan su explotación mediante herramientas tradicionales de consulta y análisis.

Esta situación pone de manifiesto la necesidad de desarrollar plataformas capaces de facilitar la interacción con grandes conjuntos de datos de forma más flexible e intuitiva. En particular, el uso de

técnicas de inteligencia artificial y de modelos de lenguaje permite plantear nuevos enfoques para la exploración y el análisis de información marítima, reduciendo la complejidad técnica asociada al acceso a los datos.

2.5 Estudio de trabajos relacionados

El rápido crecimiento del uso de datos AIS y el interés creciente por aplicar técnicas avanzadas de análisis en el dominio marítimo han motivado diversos trabajos de investigación y desarrollos académicos relacionados con esta temática. En este subapartado se revisan y analizan algunos de los trabajos más relevantes para el contexto de este Trabajo Fin de Grado, con el objetivo de identificar sus principales aportaciones, así como las limitaciones que presentan en relación con el enfoque propuesto.

2.5.1 AIS-LLM

El documento *AIS-LLM: A Unified Framework for Maritime Trajectory Prediction, Anomaly Detection, and Collision Risk Assessment with Explainable Forecasting* [27] integra datos del Sistema de Identificación Automática (AIS) con modelos de lenguaje de gran tamaño (LLM) para el análisis avanzado del tráfico marítimo. El sistema permite realizar simultáneamente la predicción de trayectorias, detección de anomalías y evaluación de riesgos de colisión de buques. Además, el documento destaca su capacidad para generar informes explicativos en lenguaje natural, aportando interpretabilidad y contexto a los resultados numéricos.

El enfoque adoptado se centra en el modelo, analizando su capacidad para interpretar información procedente de los mensajes AIS y generar resultados y explicaciones en lenguaje natural. Una de las aportaciones más destacables del trabajo es la demostración de que los modelos de lenguaje pueden emplearse de forma efectiva en tareas complejas de análisis marítimo, integrando distintos tipos de análisis dentro de un mismo marco. Asimismo, la generación de explicaciones en lenguaje natural contribuye a mejorar la interpretabilidad de los resultados, aspecto especialmente relevante en contextos operacionales.

Sin embargo, el trabajo presenta ciertas limitaciones desde la perspectiva de la interacción con los datos y el análisis exploratorio. El sistema está planteado como un marco experimental orientado a evaluar el rendimiento del modelo en un conjunto concreto de tareas previamente definidas, más que como una plataforma de consulta flexible orientada al usuario. La interacción con los datos se realiza de forma estructurada y guiada por el propio diseño del sistema, lo que restringe la posibilidad de formular consultas abiertas o adaptar el análisis a nuevas necesidades no contempladas inicialmente.

En relación con el presente Trabajo Fin de Grado, este trabajo resulta relevante al demostrar el potencial de los modelos de lenguaje para el análisis de datos AIS. No obstante, su enfoque se centra en la evaluación del modelo en tareas concretas previamente definidas, mientras que el sistema desarrollado en este proyecto se orienta a permitir una interacción más flexible con los datos mediante consultas en lenguaje natural, facilitando el análisis exploratorio y el apoyo a la toma de decisiones.

2.5.2 Using LLMs for Analyzing AIS Data

El trabajo *Using LLMs for Analyzing AIS Data* [28] aborda el uso de modelos de lenguaje aplicados a datos AIS con un enfoque evaluativo. A diferencia de otros trabajos centrados en tareas específicas, los autores analizan distintas estrategias para interactuar con grandes volúmenes de datos marítimos mediante lenguaje natural, evaluando tanto enfoques basados en traducción de consultas como métodos de razonamiento directo sobre los datos.

El estudio compara cuatro estrategias principales. Por un lado, se analiza un enfoque clásico de tipo NLIDB (*Natural Language Interface to DataBases*), en el que las consultas en lenguaje natural se traducen a consultas estructuradas. Por otro, se evalúan distintos métodos de razonamiento directo, como ZSA (*Zero-Shot Analysis*), que se diferencian en la forma en que se presenta la información AIS al

modelo. Estos métodos incluyen el razonamiento sobre datos AIS brutos, sobre trayectorias simplificadas y sobre trayectorias con una representación más compleja. Esta diferenciación permite analizar cómo la forma de estructurar los datos influye en la capacidad del modelo para extraer información relevante.

Una de las contribuciones más interesantes del trabajo es el análisis del impacto de la ventana de contexto del modelo en la escalabilidad del sistema. La ventana de contexto se refiere a la cantidad máxima de información que un modelo de lenguaje puede procesar simultáneamente en una única consulta, generalmente medida en número de *tokens* (fragmentos de texto).

Para poder manejar grandes volúmenes de información, los autores emplean un modelo con una ventana de contexto excepcionalmente amplia. Sin embargo, el estudio muestra que, incluso en estas condiciones, el razonamiento directo sobre datos espaciotemporales consume rápidamente el contexto disponible. A medida que aumenta el tamaño del conjunto de datos, es necesario recortar información inicial, lo que provoca una pérdida progresiva de contexto y un deterioro notable del rendimiento de los métodos de razonamiento directo. Este fenómeno evidencia una limitación estructural de los LLMs cuando se utilizan sin mecanismos adicionales de filtrado o recuperación de información.

Otro resultado especialmente relevante es que, en escenarios con volúmenes de datos reducidos, el razonamiento directo sobre datos AIS brutos ofrece resultados comparables, e incluso superiores, a los obtenidos con representaciones más elaboradas de las trayectorias. Este hallazgo sugiere que los modelos actuales son capaces de interpretar coordenadas numéricas y series temporales sin necesidad de transformar la información en descripciones complejas, al menos cuando la escala del problema lo permite. No obstante, esta ventaja desaparece a medida que el tamaño del conjunto de datos aumenta, reforzando de nuevo la importancia de la escalabilidad.

El trabajo también pone de manifiesto las limitaciones actuales de los LLMs para abordar tareas de análisis de datos más sofisticadas. En particular, los autores muestran que el modelo no es capaz de resolver correctamente problemas de extracción de datos complejos, como la identificación de grupos de buques que navegan conjuntamente durante largos periodos de tiempo. Este tipo de tareas requiere una capacidad de agregación y análisis estructurado que va más allá del razonamiento directo sobre los datos proporcionados.

Este estudio ofrece una evaluación crítica y realista de las capacidades y limitaciones de los modelos de lenguaje aplicados al análisis de datos AIS. Los resultados obtenidos refuerzan la idea de que el uso de LLMs de forma aislada no es suficiente para garantizar un análisis robusto y escalable de grandes volúmenes de datos. En este sentido, el trabajo justifica la necesidad de arquitecturas que incorporen mecanismos de recuperación, filtrado y estructuración previa de la información, como las basadas en enfoques RAG, para poder aprovechar de forma efectiva el potencial de los modelos de lenguaje en escenarios reales de análisis marítimo.

Los resultados del estudio apuntan de forma clara hacia la necesidad de separar el acceso a los datos del proceso de generación de respuestas. Los problemas derivados del uso directo de grandes volúmenes de datos dentro de la ventana de contexto del modelo, así como la pérdida de rendimiento a medida que aumenta la escala del conjunto de datos, evidencian las limitaciones de un enfoque puramente generativo. Frente a estas dificultades, las arquitecturas basadas en RAG permiten recuperar únicamente la información relevante para cada consulta, reduciendo la carga de contexto y proporcionando al modelo datos filtrados y estructurados sobre los que razonar.

En relación con el Trabajo Fin de Grado, los resultados de este estudio ponen de manifiesto algunas de las limitaciones que presentan los modelos de lenguaje cuando se aplican directamente al análisis de grandes volúmenes de datos AIS, especialmente en lo relativo a la gestión del contexto y la escalabilidad. Estas limitaciones refuerzan la necesidad de emplear enfoques que integren mecanismos de recuperación de información, como las arquitecturas RAG, que permiten complementar las capacidades del modelo con acceso a información externa relevante.

2.5.3 Democratización de los datos con Inteligencia Artificial: un asistente virtual basado en LLM

El Trabajo Fin de Máster desarrollado por Paola Herrezuelo de la Fuente [29] aborda el diseño de un asistente virtual basado en modelos de lenguaje de gran escala para la consulta de bases de datos relacionales mediante lenguaje natural. El objetivo principal del trabajo es facilitar el acceso a la información contenida en sistemas de datos estructurados, permitiendo que usuarios sin conocimientos técnicos avanzados puedan formular consultas sin verse limitados.

Desde el punto de vista arquitectónico, el sistema propuesto va más allá de una simple cadena lineal de procesamiento. La lógica de funcionamiento se articula como un grafo de ejecución, gestionado mediante LangGraph [30], en el que el flujo de acciones puede bifurcarse en función de los resultados intermedios obtenidos. Un elemento especialmente relevante es el nodo encargado de evaluar los datos recuperados tras la ejecución de la consulta, que decide dinámicamente cómo debe generarse la respuesta final. En función del volumen de información obtenida, el sistema adopta comportamientos distintos, desde la generación de una respuesta informativa cuando no se obtienen resultados, hasta la activación de flujos más complejos cuando se recuperan múltiples registros, incluyendo la preparación de estructuras adecuadas para su posterior visualización. Esta lógica aporta flexibilidad al sistema y permite adaptar la respuesta a la consulta.

Otro aspecto destacable del trabajo es su énfasis en la presentación de la información y en la accesibilidad de los resultados. El sistema no se limita a ofrecer respuestas textuales, sino que incorpora mecanismos de visualización interactiva que facilitan la comprensión de los datos recuperados. Este enfoque resulta clave para la democratización del acceso a la información, ya que permite que usuarios con distintos niveles de formación puedan interpretar los resultados de forma clara. En entornos organizativos amplios, como puede ser el ámbito de la Armada, donde conviven perfiles técnicos con otros de carácter más operativo, este tipo de visualización contribuye a reducir la dependencia de conocimientos especializados y favorece una toma de decisiones más informada a distintos niveles jerárquicos.

En relación con las limitaciones del sistema, el trabajo pone de manifiesto la importancia del factor humano en entornos profesionales. Aunque el asistente está diseñado para funcionar de forma completamente automatizada con el objetivo de agilizar la interacción, la autora destaca la necesidad de incorporar un enfoque de tipo *human-in-the-loop* en escenarios reales de producción. En particular, se plantea la conveniencia de que un experto valide las consultas SQL generadas por el modelo antes de su ejecución, como medida de seguridad y control. Esta reflexión resulta especialmente pertinente en contextos sensibles, tanto en el sector bancario analizado en el trabajo como en el ámbito de la defensa, donde los oficiales desempeñan un papel fundamental en la supervisión y validación de la información utilizada para la toma de decisiones.

Este trabajo aporta una referencia clara sobre cómo los modelos de lenguaje pueden integrarse en sistemas de consulta en lenguaje natural de forma estructurada y controlada. No obstante, su aplicación se limita a bases de datos relacionales y a entornos de datos acotados, sin abordar los retos asociados a grandes volúmenes de información, flujos de datos en tiempo real o análisis exploratorio complejo. Estas limitaciones refuerzan la necesidad de arquitecturas más generales y flexibles, como las basadas en RAG, que permitan combinar mecanismos de recuperación de información, razonamiento del modelo y supervisión humana en escenarios de análisis más exigentes, como el tratamiento de datos AIS.

2.5.4 Técnicas de lenguaje natural aplicadas a consultas documentales

El Trabajo Fin de Grado desarrollado por León Martín [31] ofrece un análisis detallado y práctico del diseño de sistemas basados en agentes RAG orientados a la consulta documental mediante lenguaje natural. A diferencia de otros trabajos revisados, este proyecto profundiza no solo en el uso conceptual

de RAG, sino en las decisiones técnicas necesarias para construir un sistema completo, autónomo y ejecutable en entornos reales.

Uno de los aspectos más destacados del trabajo es la apuesta explícita por una infraestructura de ejecución completamente local, eliminando la dependencia de servicios en la nube. Para ello, el sistema se apoya en la herramienta Ollama como pieza central de la arquitectura, utilizando un modelo de la familia LLaMA, concretamente LLaMA 3 en su versión de 8 mil millones de parámetros (Llama3.1:8B), ejecutado íntegramente en el equipo dedicado. Esta decisión garantiza que los documentos procesados no abandonen el entorno local, reforzando el control sobre los datos y la privacidad de la información. Este enfoque conecta directamente con una de las preocupaciones fundamentales del presente TFG, como es la necesidad de mantener el control del sistema y de los datos en escenarios sensibles.

El trabajo describe además de forma detallada el flujo de ingesta y procesamiento de los documentos, poniendo de manifiesto la importancia de una arquitectura modular bien definida. El *pipeline* comienza con la carga de documentos mediante herramientas específicas para el tratamiento de archivos PDF, seguida de un proceso de fragmentación del texto en unidades manejables. Esta fragmentación permite equilibrar la cantidad de información proporcionada al modelo y facilita una recuperación más precisa. Posteriormente, los fragmentos de texto se transforman en representaciones vectoriales mediante un modelo de *embeddings* optimizado para búsqueda semántica, lo que constituye la base para los procesos de recuperación de información posteriores.

En relación con el almacenamiento y la recuperación de estos *embeddings*, el trabajo propone una estrategia dual que combina persistencia y eficiencia. Por un lado, los vectores generados se almacenan de forma persistente en una base de datos relacional extendida para soportar operaciones vectoriales (PostgreSQL), lo que garantiza la persistencia de la información. Por otro, en el momento de la consulta se construye un índice en memoria utilizando FAISS (*Facebook AI Similarity Search*) [32], que permite realizar búsquedas de similitud de forma muy eficiente. No obstante, el propio autor identifica como una limitación el hecho de que este índice deba reconstruirse en cada consulta, lo que introduce latencia adicional, y plantea como línea de mejora futura la persistencia directa del índice vectorial.

Otro aspecto crítico abordado en el trabajo es el impacto del hardware en el rendimiento del sistema. El estudio documenta de forma explícita que, en entornos sin una GPU dedicada, la ejecución completa del *pipeline*, incluyendo recuperación y generación, puede alcanzar tiempos de respuesta del orden de varios minutos por consulta. Este resultado pone de manifiesto que la viabilidad operativa de los agentes RAG no depende únicamente del diseño lógico del sistema, sino también de la infraestructura disponible. Esta observación refuerza una de las ideas clave del presente TFG: la selección de herramientas, modelos y hardware condiciona de manera directa la aplicabilidad real de estas tecnologías en entornos operacionales.

La lógica de orquestación del sistema se articula mediante el uso de LangChain, que actúa como elemento integrador entre los distintos componentes del *pipeline*. Este *framework* permite coordinar el proceso de recuperación de información, enriquecer el *prompt* del modelo con el contexto relevante y gestionar la generación de la respuesta final. De este modo, el modelo de lenguaje no opera de forma aislada, sino como parte de un agente que controla el flujo completo de la consulta, ejemplificando de forma clara el papel central de los agentes RAG en este tipo de arquitecturas.

Desde una perspectiva más amplia, el trabajo demuestra que los agentes RAG son capaces de manejar información heterogénea, como manuales técnicos, artículos o documentos normativos, proporcionando respuestas contextualizadas y fundamentadas en los datos recuperados.

3 DESARROLLO DEL TFG

En este apartado se presenta el desarrollo de la plataforma propuesta, describiendo de forma progresiva las decisiones técnicas adoptadas durante el proceso de diseño e implementación del sistema. A lo largo de las distintas subsecciones se detalla la configuración del entorno de ejecución, la selección y despliegue de los modelos de lenguaje, así como la construcción de la arquitectura que permite transformar consultas en lenguaje natural en operaciones analíticas sobre datos AIS. Asimismo, se exponen las distintas iteraciones realizadas durante el desarrollo, incluyendo las limitaciones detectadas en los enfoques iniciales y las mejoras introducidas hasta alcanzar la arquitectura final del sistema.

3.1 Objetivo y principios de diseño del sistema

Antes de abordar el desarrollo técnico de la plataforma, resulta necesario establecer los objetivos generales del sistema y los principios de diseño que han guiado su implementación. Estos criterios permiten definir con claridad el propósito del proyecto y sirven como marco de referencia para la toma de decisiones a lo largo del proceso de desarrollo.

El objetivo principal del sistema desarrollado en este proyecto es el diseño de una plataforma de consulta inteligente orientada al análisis de datos de tráfico marítimo procedentes del sistema AIS. Estos datos contienen información relevante sobre la posición, velocidad, rumbo y características de los buques, lo que permite estudiar distintos aspectos relacionados con el comportamiento del tráfico marítimo.

La plataforma propuesta busca facilitar el acceso y explotación de esta información mediante la posibilidad de realizar consultas en lenguaje natural sobre grandes volúmenes de datos estructurados. De este modo, el sistema permite obtener información relevante de forma más accesible e intuitiva, evitando la necesidad de que el usuario disponga de conocimientos técnicos avanzados sobre bases de datos o lenguajes de consulta especializados.

Desde el inicio del diseño, se establecen una serie de principios fundamentales que condicionan las decisiones adoptadas:

- En primer lugar, se prioriza la **ejecución completamente local del sistema**, evitando el uso de servicios en la nube o APIs externas. Este enfoque garantiza el control total sobre los datos procesados, un aspecto especialmente relevante en entornos donde la confidencialidad, la integridad de la información o las restricciones operativas son factores críticos. Asimismo, este planteamiento permite asegurar que los datos AIS no abandonan en ningún momento el entorno de ejecución controlado.
- Otro objetivo clave del sistema es la **obtención de respuestas fundamentadas exclusivamente en información real** contenida en los conjuntos de datos AIS, evitando la

generación de resultados no verificables o inconsistentes. Para ello, se diseña la arquitectura de forma que el razonamiento semántico proporcionado por los modelos de lenguaje se apoye siempre en mecanismos de consulta y análisis de datos estructurados. Este principio resulta esencial para minimizar las alucinaciones propias de los modelos LLM, especialmente en consultas que implican cálculos numéricos o análisis estadísticos.

- Además, el sistema se concibe bajo un **enfoque modular y escalable**, permitiendo la separación clara entre los distintos componentes funcionales. En particular, se diferencian de manera explícita los procesos de comprensión semántica de las consultas en lenguaje natural, la ejecución analítica sobre los datos AIS y la presentación de los resultados al usuario final. Esta separación facilita tanto el mantenimiento del sistema como la evolución futura de la arquitectura, permitiendo la sustitución o mejora de componentes concretos sin necesidad de rediseñar el conjunto completo.
- Finalmente, el **diseño del sistema** se aborda desde una **perspectiva incremental**, en la que cada decisión se evalúa progresivamente a partir de pruebas funcionales y resultados intermedios. Este enfoque permite solventar las limitaciones y oportunidades detectadas durante el desarrollo, para ajustar el sistema a los objetivos iniciales.

Los principios descritos constituyen, por tanto, la base conceptual sobre la que se estructura el desarrollo del sistema. A lo largo de los distintos apartados de este capítulo se describen las decisiones técnicas adoptadas para la implementación de la plataforma, todas ellas alineadas con estos objetivos iniciales y orientadas a garantizar un funcionamiento fiable, eficiente y adaptado a las particularidades del análisis de datos AIS.

3.2 Preparación del entorno de ejecución y despliegue inicial de modelos LLM

3.2.1 Selección y despliegue del entorno de modelos de lenguaje

El entorno utilizado para el desarrollo de este TFG consiste en una máquina virtual desplegada sobre un servidor de altas prestaciones. Esta configuración permite trabajar con mayores capacidades computacionales que las ofrecidas por un equipo local estándar, lo que resulta especialmente relevante para las tareas planteadas en el proyecto.

La elección de este entorno se justifica principalmente por la disponibilidad de una gran cantidad de memoria RAM y de GPU. Este recurso es fundamental para el despliegue y ejecución de modelos de lenguaje de gran tamaño, así como para el procesamiento de volúmenes elevados de datos durante las fases de análisis.

Además de la memoria disponible, el servidor proporciona recursos de procesamiento y almacenamiento suficientes para manejar simultáneamente distintos componentes del sistema, lo cual es muy útil durante el desarrollo y las pruebas, donde es necesario ejecutar modelos, realizar consultas sobre grandes conjuntos de datos y validar diferentes configuraciones del sistema.

Asimismo, las características técnicas del sistema (véanse en la Tabla 3-1) permiten almacenar tanto los modelos LLM utilizados como los conjuntos de datos AIS empleados en el proyecto sin restricciones significativas de espacio. Esto facilita trabajar con los datos de forma local y mantener todos los componentes del sistema dentro del mismo entorno de ejecución.

En conjunto, este entorno proporciona unas prestaciones claramente superiores a las de un equipo convencional, permitiendo desarrollar y ejecutar la plataforma en condiciones adecuadas de rendimiento y estabilidad.

Características de la máquina virtual

CPU	8 (0-7)
ID Fabricante	GenuineIntel
Nombre del modelo	Intel(R) Core(TM) i9-14900K
Familia de la CPU	6
Modelo	183
Hilos por núcleo	1
Núcleos por socket	1
Sockets	8
Revisión del procesador	1
BogoMIPS	6374,40
Memoria RAM	40 GB
Almacenamiento secundario	500 GB
Tarjeta gráfica	NVIDIA RTX 4090 24 GB

Tabla 3-1 Características de la máquina virtual

Una vez definido el entorno de ejecución, con el sistema operativo Ubuntu 24.04 LTS, se lleva a cabo un análisis de los distintos modelos disponibles dentro de la familia LLaMA, evaluando su adecuación en función de los recursos hardware disponibles y de los objetivos del proyecto. Este estudio tuvo en cuenta aspectos como el tamaño del modelo, el consumo de memoria, el rendimiento esperado en tareas de razonamiento y consulta en lenguaje natural, así como la viabilidad de su despliegue en infraestructura propia. Como resultado de este análisis, se decide optar por el modelo LLaMA 3.1:8B, al ofrecer un equilibrio adecuado entre capacidad de procesamiento, calidad de las respuestas y requisitos computacionales.

Por último, se opta por utilizar contenedores *Docker* como entorno de ejecución de los distintos componentes del sistema. Esta decisión permite encapsular cada servicio con sus dependencias, garantizando un entorno reproducible y aislado, especialmente relevante en un proyecto experimental.

3.2.2 Implementación del entorno LLM local y carga inicial de datos

Para la implementación del entorno LLM local se hace uso de manuales y documentación oficial correspondientes a cada una de las tecnologías empleadas en el proyecto. El uso de estas fuentes permite garantizar una correcta instalación, configuración y funcionamiento de los distintos componentes del sistema, así como asegurar la reproducibilidad del entorno y el cumplimiento de las buenas prácticas recomendadas por los desarrolladores de cada herramienta.

En primer lugar, se procede a la instalación y configuración de *Docker* [33] en la máquina virtual que aloja la arquitectura. A partir de este momento, cada servicio (modelo de lenguaje, motor de indexado, aplicación web, etc.) se despliega como un contenedor independiente, lo que permite orquestar su ejecución, supervisar su estado y reconstruir el entorno completo de forma controlada ante cualquier incidencia.

Dado que el proyecto requiere la ejecución de modelos de lenguaje de gran tamaño con aceleración por GPU, es necesaria la instalación del *NVIDIA Container Toolkit* [34]. Esta herramienta permite que los contenedores *Docker* tengan acceso directo a los recursos gráficos de la máquina virtual, resultando fundamental para la inferencia eficiente de los modelos LLM y para la generación de *embeddings* asociados a los datos AIS.

Los *embeddings* son representaciones vectoriales de información (como palabras, frases o documentos) en un espacio numérico de múltiples dimensiones. Mediante este proceso, los modelos transforman texto u otros datos en vectores de números que capturan relaciones semánticas entre los elementos. De este modo, contenidos con significados similares tienden a situarse cerca entre sí dentro del espacio vectorial, lo que permite realizar tareas como búsqueda semántica, clasificación o recuperación de información de forma eficiente.

Una vez configurada la infraestructura base, se crea un contenedor *Docker* dedicado a Ollama [35], que actúa como motor de ejecución de los modelos de lenguaje. Dentro de este contenedor se instalan los modelos Llama3.1:8b y Llama3.1:latest, con el objetivo de poder evaluar y comparar el comportamiento de ambas versiones en tareas de consulta en lenguaje natural. Adicionalmente, se instala el modelo *nomic-embed-text*, destinado a la generación de *embeddings*, que será utilizado posteriormente en la vectorización de los datos y en la implementación de la arquitectura RAG del sistema. Durante esta fase se ejecutaron diversos comandos de instalación y configuración, que se documentan de forma detallada en el Anexo V.

Una vez desplegado el entorno LLM local y configurados los principales componentes del sistema, se procede a la selección y carga inicial de datos AIS, con el objetivo de disponer de un conjunto de información real sobre el que validar el correcto funcionamiento de la plataforma y preparar las siguientes fases del proyecto.

Se seleccionan datos AIS correspondientes al mes de enero de 2023, de carácter global, incluyendo información de carácter dinámico y estático (en torno a 16 GB en total). Esta selección permite trabajar con un conjunto de datos representativo del tráfico marítimo real, abarcando tanto información temporal sobre la posición y el movimiento de los buques como datos más estables relacionados con sus características generales. El uso de diferentes tipologías de datos AIS resulta especialmente relevante para evaluar, en fases posteriores, la capacidad del sistema para integrar y consultar información heterogénea.

3.3 Aproximación inicial de la plataforma

Como primera aproximación de este proyecto se decide utilizar Open WebUI [36] como interfaz de interacción con los modelos desplegados, para el cual se crea un contenedor independiente. Open WebUI permite utilizar de forma directa los modelos de lenguaje instalados en Ollama y presentarlos a través de una interfaz gráfica intuitiva, facilitando su uso durante las fases de desarrollo y validación.

Adicionalmente, Open WebUI incorpora de forma nativa un mecanismo básico de RAG, que posibilita la realización de consultas en lenguaje natural sobre los datos cargados, gestionando internamente procesos como la generación de *embeddings* y la recuperación de información relevante. Además, la herramienta permite la carga de archivos en formato CSV, formato en el que se almacenan los datos AIS empleados en este proyecto. Estas características convierten a Open WebUI en una solución adecuada tanto para la exploración inicial de los datos como para una primera aproximación funcional al sistema de consulta inteligente propuesto.

Una vez cargados los conjuntos de datos AIS en el entorno de Open WebUI, se realizaron diversas pruebas iniciales de consulta en lenguaje natural con el objetivo de verificar el correcto funcionamiento del sistema y evaluar su comportamiento sobre datos reales.

Durante esta fase inicial de carga y análisis de los datos AIS, se identifica una limitación relevante relacionada con el procesamiento de los conjuntos de datos de mayor tamaño. Es importante destacar que dicha limitación no está asociada a errores de subida de archivos ni a restricciones de memoria RAM o de tamaño máximo de fichero, sino al tiempo necesario para el procesamiento interno de los datos

durante la generación de *embeddings* dentro de la herramienta Open WebUI, como se puede observar en la Figura 3-1, donde se visualiza un ejemplo de carga de un archivo de tamaño reducido.

```
Batches: 5%| | 46546/999999 [11:47<4:46:51, 55.40it/s]
2026-02-05 19:02:19.111 INFO | uvicorn.protocols.http.httptools_impl:send:483 - 91.117.19.229:64712 - "GET /_app/version.1
Batches: 5%| | 47457/999999 [12:10<4:51:38, 54.43it/s]
2026-02-05 19:02:42.323 INFO | uvicorn.protocols.http.httptools_impl:send:483 - 91.117.19.229:63804 - "GET /_app/version.1
Batches: 5%| | 48918/999999 [12:47<5:43:16, 46.18it/s]
2026-02-05 19:03:18.590 INFO | uvicorn.protocols.http.httptools_impl:send:483 - 91.117.19.229:65136 - "GET /_app/version.1
Batches: 5%| | 49873/999999 [13:11<4:57:06, 53.30it/s]
2026-02-05 19:03:42.713 INFO | uvicorn.protocols.http.httptools_impl:send:483 - 91.117.19.229:63632 - "GET /_app/version.1
Batches: 5%| | 51302/999999 [13:47<10:39:02, 24.74it/s]
2026-02-05 19:04:18.593 INFO | uvicorn.protocols.http.httptools_impl:send:483 - 91.117.19.229:64880 - "GET /_app/version.1
Batches: 5%| | 52288/999999 [14:11<5:07:35, 51.35it/s]
2026-02-05 19:04:43.259 INFO | uvicorn.protocols.http.httptools_impl:send:483 - 91.117.19.229:64382 - "GET /_app/version.1
Batches: 5%| | 53697/999999 [14:47<4:37:44, 56.79it/s]
2026-02-05 19:05:18.905 INFO | uvicorn.protocols.http.httptools_impl:send:483 - 91.117.19.229:63848 - "GET /_app/version.1
Batches: 5%| | 54671/999999 [15:12<5:09:09, 50.96it/s]
2026-02-05 19:05:43.724 INFO | uvicorn.protocols.http.httptools_impl:send:483 - 91.117.19.229:64254 - "GET /_app/version.1
Batches: 6%| | 55394/999999 [15:30<5:50:16, 44.95it/s]
```

Figura 3-1 Captura del log cuando se sube un documento de 6Mb con 100,000 filas.

En concreto, al cargar archivos CSV de gran volumen, el sistema no produce errores explícitos ni interrupciones del servicio, pero la interfaz permanece aparentemente inactiva durante periodos prolongados de tiempo. Este comportamiento se debe a la lentitud del proceso de vectorización en segundo plano, en el que el *backend* continúa trabajando sin proporcionar retroalimentación visual clara al usuario, dando la sensación de que la aplicación se encuentra bloqueada o congelada.

Con el objetivo de mitigar este problema, se intenta vincular explícitamente el proceso de generación de *embeddings* al modelo *nomic-embed-text*, desplegado previamente en Ollama, con la intención de aprovechar un modelo de *embeddings* optimizado y ejecutado de forma local. No obstante, aunque esta configuración permite un mayor control sobre el proceso de vectorización, los tiempos de procesamiento siguen siendo excesivos para conjuntos de datos AIS de gran tamaño. En el caso de los datos empleados en este trabajo, cada archivo correspondiente a un día de datos estáticos ocupa aproximadamente 200 MB, mientras que los archivos de datos dinámicos alcanzan en torno a 400 MB por día, lo que implica manejar volúmenes de información considerables. Esta circunstancia dificulta su uso práctico en esta fase del proyecto.

Como consecuencia de estas limitaciones, se decide descartar el uso de Open WebUI como herramienta principal para la ingestión y vectorización de grandes volúmenes de datos AIS, reservándola únicamente para pruebas exploratorias y validaciones preliminares. Sin embargo, con el fin de evaluar el enfoque propuesto, se realizan pruebas adicionales utilizando archivos CSV cuyo tamaño había sido reducido de forma significativa mediante técnicas de filtrado y selección de columnas relevantes.

Estas pruebas permiten comprobar que, una vez reducido el volumen de datos, la arquitectura basada en RAG funciona correctamente, siendo capaz de generar *embeddings*, recuperar información relevante y responder a consultas en lenguaje natural de manera coherente y eficiente. De este modo, aunque se identifican limitaciones prácticas en la herramienta utilizada para la carga inicial de grandes conjuntos de datos, se puede demostrar la viabilidad conceptual y funcional del enfoque RAG aplicado al análisis de datos de tráfico marítimo.

3.4 Justificación de la arquitectura

Tras las pruebas iniciales realizadas con herramientas que incorporan arquitecturas de consulta ya preconfiguradas, se identifican diversas limitaciones que dificultan su uso como base para el desarrollo de la plataforma propuesta. Estas herramientas resultan útiles para una primera aproximación al funcionamiento de los modelos de lenguaje y de los sistemas de consulta en lenguaje natural, permitiendo realizar pruebas preliminares. Sin embargo, presentan restricciones importantes en cuanto al control sobre los distintos componentes internos del sistema.

En primer lugar, el elevado volumen de información asociado a los datos AIS hace que los mecanismos de vectorización integrados en estas plataformas resulten poco eficientes cuando se trabaja con conjuntos de datos de gran tamaño.

Además, este tipo de herramientas oculta gran parte de la lógica interna del sistema, lo que dificulta modificar u optimizar aspectos clave del proceso de consulta, como la forma en la que el modelo de lenguaje interpreta las preguntas del usuario, cómo se recupera la información relevante o cómo se ejecutan finalmente las operaciones analíticas sobre los datos.

Por este motivo, se decide abandonar el uso de herramientas de consulta preconfiguradas como solución principal y proceder al diseño e implementación de una arquitectura propia desarrollada desde cero. Este enfoque permite definir de forma explícita cada uno de los componentes del sistema, incluyendo el modelo de lenguaje utilizado, los mecanismos de recuperación de contexto, el motor encargado del procesamiento de los datos y la lógica que conecta las consultas en lenguaje natural con las operaciones analíticas sobre los datos AIS.

De este modo, la arquitectura desarrollada busca combinar las capacidades de comprensión del lenguaje natural de los modelos de lenguaje con la precisión y eficiencia de motores especializados en el procesamiento de datos. Esta aproximación permite mantener la flexibilidad de la interacción en lenguaje natural sin renunciar al control sobre el tratamiento de grandes volúmenes de información.

Además, en lugar de abordar el desarrollo como una solución única desde el inicio, se adopta un enfoque progresivo e incremental, en el que la arquitectura es construida de manera escalonada. Este planteamiento consiste en probar de forma gradual distintos componentes y configuraciones, evaluando su comportamiento y resultados en cada etapa, descartando aquellas soluciones que no ofrecen un rendimiento adecuado y consolidando las que demuestran ser más eficaces. Este proceso iterativo permite refinar la arquitectura final del sistema, sentando una base sólida para la implementación del *pipeline* de análisis y consulta inteligente que se describe en los apartados siguientes.

3.5 Primera propuesta de arquitectura

3.5.1 Arquitectura general del sistema

La arquitectura implementada se estructura en una serie de componentes interconectados que permiten transformar una consulta en lenguaje natural en un análisis preciso sobre datos AIS estructurados. Desde un punto de vista funcional, el sistema integra mecanismos de interacción con el usuario, generación y procesamiento de consultas, recuperación de contexto semántico y ejecución analítica sobre base de datos.

A nivel general, la solución se organiza en tres bloques principales: interfaz de usuario, lógica de aplicación, motor de modelos de lenguaje y sistema de almacenamiento y análisis de datos (véase en Figura 3-2). Cada uno de estos bloques cumple una función específica dentro del flujo completo de una consulta, cuya coordinación permite combinar razonamiento semántico y ejecución determinista de operaciones sobre datos estructurados.

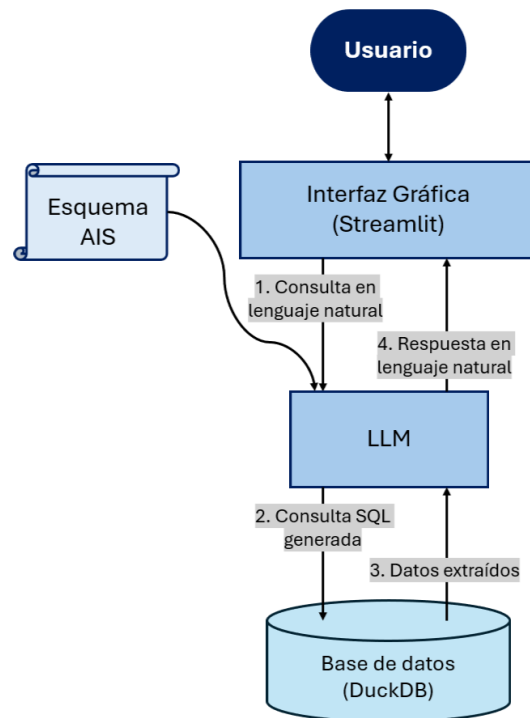


Figura 3-2 Representación del flujo de una consulta en la primera propuesta de arquitectura

Desde el punto de vista del despliegue, el sistema se ejecuta íntegramente en un entorno virtualizado, utilizando contenedores *Docker* como mecanismo de aislamiento y coordinación de servicios. Esta decisión permite garantizar reproducibilidad, control del entorno y facilidad de despliegue. La arquitectura se organiza en dos contenedores principales: un contenedor dedicado a la ejecución de los modelos de lenguaje mediante Ollama y otro contenedor que integra la lógica de aplicación, la gestión de consultas y la interfaz gráfica desarrollada con Streamlit [37]. Ambos contenedores se comunican a través de una red interna de *Docker*, evitando exposiciones innecesarias y manteniendo el sistema dentro de un entorno controlado.

El contenedor de Ollama actúa como servidor local de modelos, alojando tanto el modelo de lenguaje principal (llama3.1:8b) como el modelo de *embeddings* (*nomic-embed-text*). Este componente es responsable de la generación de texto, la traducción de lenguaje natural a SQL. La ejecución local de estos modelos elimina la dependencia de servicios externos y garantiza que ningún dato procesado abandone la infraestructura definida, aspecto especialmente relevante cuando se trabaja con datos operativos potencialmente sensibles.

Por su parte, el contenedor de la aplicación integra dos módulos diferenciados. En primer lugar, el módulo de gestión de datos, encargado de almacenar y procesar la información AIS utilizada por la plataforma. Para ello se emplea DuckDB [38], un sistema gestor de bases de datos analítico diseñado para ejecutar consultas SQL de forma eficiente sobre grandes volúmenes de datos en entornos locales.

En el contexto de este proyecto, los datos AIS procedentes de los archivos CSV originales se importan previamente en una base de datos gestionada por DuckDB, donde quedan almacenados en formato *.db*. Este proceso permite transformar los archivos de datos en una estructura optimizada para la ejecución de consultas analíticas. De este modo, DuckDB actúa como motor analítico dentro de la arquitectura, recibiendo las consultas SQL generadas por el sistema y devolviendo los resultados correspondientes, que posteriormente son interpretados y presentados al usuario a través del modelo de lenguaje.

Finalmente, el módulo de interfaz gráfica se desarrolla con Streamlit, elegido por su capacidad para construir interfaces web interactivas directamente desde código Python sin necesidad de desarrollar un *frontend* independiente. Esta herramienta permite prototipar rápidamente aplicaciones de análisis de datos, integrar de forma sencilla visualizaciones y gestionar estados de sesión, lo que resulta especialmente adecuado en un proyecto experimental donde la iteración rápida y la simplicidad de mantenimiento son prioritarias.

Uno de los aspectos clave de esta arquitectura es la separación clara entre razonamiento del lenguaje y ejecución analítica. Mientras que el modelo de lenguaje se encarga de interpretar la intención del usuario y generar la consulta SQL correspondiente, la ejecución real de los cálculos y agregaciones se delega exclusivamente en DuckDB.

Esta división de responsabilidades permite separar la interpretación de las consultas en lenguaje natural del procesamiento efectivo de los datos. De este modo, el modelo de lenguaje se encarga de comprender la intención del usuario y estructurar la consulta, mientras que el motor de base de datos ejecuta las operaciones analíticas sobre los datos AIS, garantizando resultados precisos y consistentes.

La arquitectura resultante integra modelos de lenguaje, herramientas analíticas y mecanismos de recuperación en un entorno completamente local, modular y escalable.

3.5.2 Gestión y almacenamiento de los datos AIS

Los datos empleados en el sistema corresponden a información procedente del AIS, almacenada inicialmente en ficheros CSV diferenciados entre datos estáticos y dinámicos. Los datos estáticos incluyen información asociada al buque, como identificador MMSI, tipo de embarcación o dimensiones, mientras que los datos dinámicos contienen información variable en el tiempo, como posición geográfica, velocidad, rumbo o marca temporal.

Con el objetivo de acceder de forma eficiente al gran volumen de información del que disponemos (mensajes AIS), se decide importar los datos contenidos en los ficheros CSV a una base de datos persistente, DuckDB. Este sistema está orientado al procesamiento analítico de datos (OLAP, *Online Analytical Processing*), un tipo de base de datos diseñado para realizar consultas de análisis sobre grandes volúmenes de información. De este modo, los datos originales se cargan en una única fase de ingesta y se almacenan como tablas internas dentro de un archivo de base de datos, proporcionando una estructura organizada y optimizada para su posterior explotación.

Este enfoque permite trabajar sobre un sistema diseñado específicamente para el procesamiento analítico de grandes volúmenes de datos. En los sistemas OLAP, las consultas no están orientadas a operaciones transaccionales de lectura y escritura frecuentes, como ocurre en bases de datos operacionales, sino al análisis agregado de grandes conjuntos de información. Aunque estas consultas se expresan habitualmente mediante el lenguaje SQL, su objetivo es realizar operaciones analíticas como filtrados complejos, agregaciones estadísticas o exploración de patrones en los datos. Este tipo de procesamiento permite obtener resultados de forma más eficiente cuando se trabaja con grandes volúmenes de datos históricos.

La gestión de datos implementada constituye, por tanto, un elemento clave en la arquitectura del sistema, ya que permite combinar la flexibilidad del lenguaje natural con la precisión y eficiencia propias de los sistemas de análisis estructurado de datos.

3.5.3 Enfoque inicial de consulta: traducción directa de lenguaje natural a SQL

Una vez establecida la infraestructura del sistema y organizada la gestión de los datos AIS mediante DuckDB, el siguiente paso consiste en definir el mecanismo de interacción entre el usuario y la base de datos. El enfoque inicial adoptado se basa en un esquema de traducción directa desde lenguaje natural a consultas SQL ejecutables sobre la base de datos.

En este modelo, el usuario introduce una consulta en lenguaje natural a través de la interfaz gráfica. El modelo de lenguaje, ejecutado localmente mediante Ollama, interpreta la intención de la consulta y genera una consulta estructurada compatible con el esquema de tablas previamente definido en DuckDB. Esta consulta es posteriormente ejecutada por el sistema gestor de base de datos, obteniendo un resultado estructurado que contiene exclusivamente datos reales almacenados en el sistema.

Finalmente, el modelo de lenguaje recibe el resultado de la consulta y genera una respuesta textual, que es mostrada al usuario. De esta manera, el modelo actúa como un intermediario entre el lenguaje natural y el análisis estructurado de datos, manteniendo una separación clara entre el componente de razonamiento semántico y el componente de ejecución analítica.

Este enfoque presenta una ventaja fundamental: la precisión numérica queda garantizada por el motor de base de datos. El modelo de lenguaje no realiza cálculos directamente, sino que delega completamente la ejecución de estos, de los filtrados y de las operaciones estadísticas en DuckDB. Esto reduce significativamente el riesgo de generación de valores incorrectos o inconsistentes, uno de los problemas habituales en sistemas basados exclusivamente en modelos generativos.

Además, esta arquitectura permite aprovechar la forma de expresar el lenguaje natural sin sacrificar la rigurosidad de las consultas estructuradas. El sistema combina la flexibilidad interpretativa del modelo de lenguaje con la determinación exacta de un motor SQL, configurando una solución robusta para el análisis de grandes volúmenes de datos AIS.

3.5.1 Interfaz gráfica de consulta e interpretación de los datos

Para facilitar el uso de la plataforma por parte del usuario, se diseña una interfaz gráfica ligera basada en Streamlit, concebida como punto de entrada único a todo el flujo de consulta. El objetivo de esta capa, además de ocultar la complejidad del sistema, es ofrecer una experiencia directa: el usuario introduce una pregunta en lenguaje natural y recibe una respuesta interpretada, evitando la interacción manual con SQL, con el esquema de datos o con resultados en bruto.

Desde el punto de vista funcional, la interfaz se encarga de recoger la consulta del usuario y activar el flujo completo de procesamiento del sistema. Al enviar una pregunta, la aplicación inicia internamente las distintas etapas del proceso: generación de una consulta compatible con DuckDB y ejecución de dicha consulta sobre los datos almacenados. Finalmente, la interfaz presenta al usuario únicamente la explicación textual del resultado, manteniendo ocultos los pasos intermedios para preservar la claridad de la interacción.

Este planteamiento, además, encaja con el enfoque modular adoptado en el diseño del sistema. La interfaz se mantiene desacoplada de los componentes encargados de la recuperación de contexto, la generación de consultas y la ejecución analítica, lo que permite que pueda evolucionar de forma independiente sin alterar la lógica interna del sistema. De este modo, la plataforma queda preparada para incorporar mejoras futuras en la interacción con el usuario sin comprometer el núcleo de análisis.

3.5.2 Limitaciones detectadas en el enfoque inicial

A pesar de que el enfoque basado en la traducción directa de lenguaje natural a consulta SQL demuestra ser funcional y preciso desde el punto de vista numérico, durante las pruebas realizadas se identifican diversas limitaciones relacionadas con la interpretación semántica de las consultas.

La principal dificultad detectada es la variabilidad del lenguaje natural empleado por los usuarios. Diferentes formas de expresar una misma intención pueden generar interpretaciones distintas por parte del modelo de lenguaje, especialmente cuando se utilizan sinónimos, abreviaturas o términos específicos del dominio marítimo que no coinciden exactamente con los nombres de columnas o tablas definidas en la base de datos. Esta situación obliga a diseñar *prompts* cada vez más específicos para guiar correctamente la generación de consultas SQL.

Asimismo, se observa una dependencia significativa del *prompt* utilizado para mapear conceptos del dominio AIS a la estructura real de los datos. Aunque el modelo de lenguaje es capaz de generar consultas válidas en muchos casos, pequeñas variaciones en la formulación pueden producir consultas incompletas o ambigüedades en la selección de columnas, filtros o agrupaciones.

Otra limitación relevante está relacionada con la falta de contexto semántico adicional. El modelo de lenguaje dispone del esquema de la base de datos y de la pregunta del usuario, pero no cuenta con una representación estructurada del conocimiento específico del dominio AIS, como definiciones formales de campos, reglas de interpretación o convenciones habituales en el análisis del tráfico marítimo. Esto puede derivar en interpretaciones inconsistentes cuando la consulta requiere comprensión más profunda del significado de determinados términos.

En conjunto, estas limitaciones no invalidan el enfoque inicial, pero muestran la necesidad de incorporar un mecanismo que refuerce la comprensión semántica del dominio sin comprometer la precisión analítica garantizada por el motor SQL, lo que conduce a la introducción de un componente adicional de recuperación de contexto.

3.6 Segunda propuesta de arquitectura

3.6.1 Introducción de un módulo RAG

Con el objetivo de mitigar las limitaciones identificadas en el enfoque inicial, se decide incorporar un mecanismo de recuperación de información basado en la arquitectura RAG. Este enfoque combina modelos de lenguaje con sistemas de recuperación de información, permitiendo que el modelo acceda dinámicamente a conocimiento externo durante el proceso de generación de respuestas. En lugar de depender únicamente del conocimiento almacenado en los parámetros del modelo, el sistema recupera información relevante de una fuente externa y la incorpora como contexto adicional en el *prompt*, mejorando así la precisión y coherencia de las respuestas generadas.

Sin embargo, la implementación adoptada en este sistema difiere del enfoque RAG clásico aplicado sobre documentos textuales extensos o bases de conocimiento abiertas. En lugar de utilizar RAG para recuperar directamente datos numéricos o registros estructurados, se diseña un módulo de recuperación orientado exclusivamente a proporcionar contexto semántico de dominio. Este módulo actúa como soporte interpretativo para el modelo de lenguaje durante la generación de consultas SQL, reforzando la correspondencia entre el lenguaje natural del usuario y la estructura real de la base de datos AIS.

El contenido incorporado en este sistema de recuperación está compuesto por elementos como un glosario de términos AIS, descripciones formales de columnas, reglas de mapeo entre conceptos habituales y nombres reales de campos, convenciones de consulta y ejemplos representativos de interpretación. Estos documentos no contienen datos dinámicos del tráfico marítimo, sino conocimiento estructurado sobre cómo deben interpretarse y consultarse dichos datos.

Para su implementación, este conjunto de información documental se fragmenta y transforma en representaciones vectoriales mediante el modelo de *embeddings*, *nomic-embed-text*, ejecutado localmente a través de Ollama. Este modelo convierte los fragmentos de texto en vectores numéricos de alta dimensión (*embeddings*) que capturan su significado semántico, permitiendo representar el contenido textual en un espacio vectorial. Los vectores generados se almacenan en un índice vectorial persistente utilizando FAISS, explicado previamente en el apartado 2.5.4.

FAISS es una biblioteca optimizada para almacenar estos vectores y realizar búsquedas rápidas de similitud entre ellos, lo que permite recuperar de forma eficiente los fragmentos de texto más relevantes para cada consulta. Este índice se emplea exclusivamente para la recuperación de contexto textual dentro del módulo RAG, mientras que los datos AIS son procesados mediante el motor analítico descrito anteriormente [39].

Sobre esta base se incorpora LlamaIndex, que actúa como capa de orquestación del módulo RAG. Esta biblioteca se encarga de gestionar el flujo completo de recuperación y generación, incluyendo la creación de *embeddings* de los documentos, su indexación en el almacén vectorial y la recuperación del contexto relevante en función de cada consulta. En este sistema, LlamaIndex utiliza FAISS como motor de búsqueda vectorial, mientras que coordina la interacción entre el índice vectorial y el modelo de lenguaje. De esta forma, FAISS proporciona la capacidad eficiente de búsqueda por similitud, mientras que LlamaIndex organiza el *pipeline* de recuperación de contexto y su integración en el *prompt* del modelo.

Durante el proceso de generación de una respuesta, el sistema recupera los fragmentos más relevantes del índice vectorial en función de la similitud semántica con la consulta introducida por el usuario. Este contexto recuperado se incorpora posteriormente en el *prompt* del modelo de lenguaje junto con el esquema de la base de datos, guiando la generación de la consulta SQL hacia una interpretación más consistente con el dominio AIS.

De esta manera, el módulo RAG no sustituye el componente analítico basado en SQL, sino que actúa como una capa de apoyo semántico. La ejecución de cálculos y agregaciones continúa siendo responsabilidad exclusiva de DuckDB, mientras que el modelo de lenguaje utiliza el contexto recuperado únicamente para mejorar la comprensión de la intención del usuario y reducir ambigüedades en la generación de consultas.

Esta arquitectura híbrida, basada en la combinación de modelo de lenguaje, recuperación de contexto de dominio y ejecución sobre la base de datos, permite equilibrar flexibilidad interpretativa y precisión analítica.

3.6.2 Integración del RAG en el flujo de generación de consultas

Una vez definido el módulo de recuperación del contexto, es necesario establecer su integración dentro del flujo de generación de consultas del sistema. El objetivo no es sustituir el mecanismo de traducción de lenguaje natural a SQL, sino reforzarlo mediante la incorporación dinámica de información contextual relevante.

El proceso comienza cuando el usuario introduce una consulta en lenguaje natural a través de la interfaz gráfica. Antes de generar la consulta SQL, el sistema utiliza dicha entrada para realizar una búsqueda en el índice vectorial construido con FAISS. Esta búsqueda devuelve los fragmentos de conocimiento de dominio más relevantes en función de la similitud semántica con la consulta.

Los fragmentos recuperados no se muestran al usuario ni se ejecutan como instrucciones independientes, sino que se integran como contexto adicional dentro del *prompt* enviado al modelo de lenguaje. De esta manera, el modelo recibe simultáneamente:

1. La consulta original del usuario.
2. El esquema estructural de la base de datos AIS.
3. El contexto recuperado desde el índice vectorial.

La combinación de estos tres elementos permite al modelo generar una consulta SQL más alineada con la terminología real del dominio y con la estructura efectiva de las tablas. El módulo de recuperación actúa, por tanto, como un mecanismo de aclaración contextual que reduce la dependencia exclusiva del *prompt* y mejora la consistencia en la generación de consultas.

Una vez generada la consulta SQL, el proceso continúa sin intervención del módulo de recuperación. La consulta se ejecuta por DuckDB sobre la base de datos materializada, y el modelo de lenguaje genera una respuesta textual basada únicamente en el resultado devuelto por el motor de base de datos, garantizando que la interpretación final esté sustentada exclusivamente en datos reales.

Esta integración permite mantener una separación clara de responsabilidades dentro de la arquitectura implementada en esta fase del proyecto. El módulo de recuperación aporta comprensión contextual; el modelo de lenguaje realiza el proceso de interpretación y conversión; y el sistema gestor de base de datos ejecuta el análisis determinista. La coordinación de estos componentes configura una arquitectura híbrida que equilibra flexibilidad interpretativa y rigor analítico.

3.6.3 Flujo completo de una consulta en el sistema

Con el objetivo de comprender el funcionamiento global del sistema, resulta conveniente describir de forma secuencial el flujo completo que sigue una consulta desde su introducción por parte del usuario hasta la generación de la respuesta final, véase de forma esquematizada en la Figura 3-3.

El proceso comienza cuando el usuario introduce una consulta en lenguaje natural a través de la interfaz gráfica desarrollada con Streamlit. Esta consulta constituye el punto de partida del procesamiento y se transmite al módulo de lógica de aplicación.

En una primera etapa, el sistema realiza una búsqueda por similitud en el índice vectorial FAISS utilizando la consulta como entrada. Esta operación permite recuperar los fragmentos de conocimiento de dominio más relevantes, previamente vectorizados mediante el modelo de *embeddings* (*nomic-embed-text*). El objetivo de esta recuperación es aportar contexto adicional que facilite la correcta interpretación de la intención del usuario.

A continuación, el modelo de lenguaje recibe como entrada conjunta la consulta original introducida por el usuario, el esquema estructural de la base de datos AIS y el contexto recuperado desde el índice vectorial. La combinación de estos tres elementos proporciona al modelo la información necesaria para interpretar correctamente la intención de la consulta y alinearla con la estructura real de las tablas almacenadas en DuckDB. Con esta información integrada, el modelo genera una consulta SQL adaptada al esquema de la base de datos, completando así el proceso de conversión del lenguaje natural en una instrucción estructurada ejecutable.

Una vez generada la consulta SQL, esta es enviada al sistema gestor de base de datos, que ejecuta la operación sobre la base de datos materializada. El resultado obtenido es un conjunto estructurado de datos reales, producto exclusivo de la ejecución determinista del motor SQL.

Finalmente, el modelo de lenguaje recibe el resultado de la consulta y genera una respuesta textual clara y comprensible para el usuario. Es importante destacar que esta respuesta se construye únicamente a partir de los datos devueltos por la base de datos, garantizando la coherencia entre la interpretación final y la información almacenada.

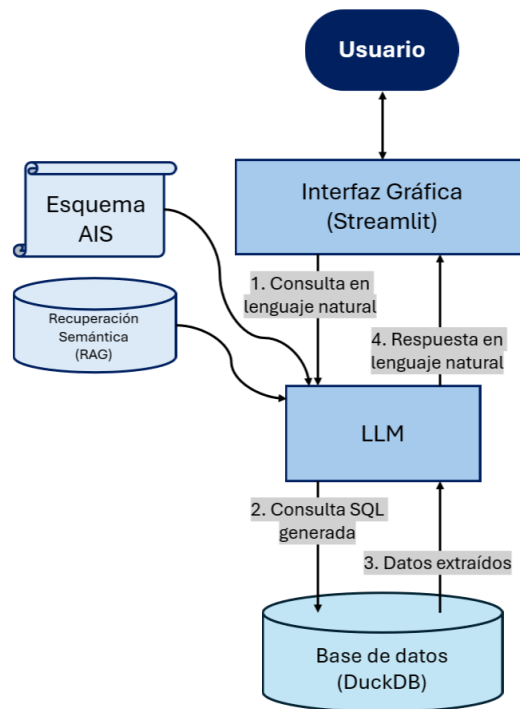


Figura 3-3 Representación del flujo de una consulta en la segunda propuesta de arquitectura

Este flujo secuencial permite mantener una separación definida entre las distintas responsabilidades del sistema: la recuperación contextual aporta información de dominio, el modelo de lenguaje interpreta y convierte la consulta, y DuckDB ejecuta el análisis estructurado.

3.6.4 Problemas detectados en la generación inicial de SQL

El modelo de lenguaje genera directamente una consulta SQL a partir de la pregunta del usuario, del esquema y del contexto. Aunque este mecanismo funciona en múltiples escenarios, durante las pruebas se identifican problemas recurrentes que afectan tanto a la estabilidad del sistema como a la calidad de la respuesta.

En primer lugar, el sistema detecta variabilidad en la salida del modelo: preguntas idénticas o muy similares generan consultas SQL diferentes entre sí. Esta variación se traduce en cambios inesperados en columnas seleccionadas, filtros omitidos o interpretaciones alternativas de una misma intención. En particular, cuando el usuario solicita un valor concreto (por ejemplo, una magnitud derivada a partir de dos columnas), el modelo puede producir consultas incompletas o interpretaciones parciales si no está guiado con suficiente contexto.

Además, se observa que el modelo tiende a cometer errores relacionados con el mapeo entre lenguaje natural y esquema real. En consultas donde el usuario emplea términos del dominio AIS (p. ej., “calado”, “eslora”, “rumbo”) el modelo no siempre selecciona las columnas correctas o intenta utilizar nombres inexistentes. Este comportamiento provoca errores de ejecución o respuestas erróneas si no existe una capa que valide el resultado.

Por otro lado, aparece un problema práctico importante: en algunos casos, el modelo devuelve una salida que no cumple el formato esperado por el sistema (por ejemplo, texto explicativo sin la consulta completa, o respuestas sin estructura clara), lo que genera fallos como la imposibilidad de extraer una consulta válida. Este tipo de fallo se vuelve especialmente crítico cuando el sistema depende de que el modelo genere un fragmento ejecutable sin ambigüedad.

Por último, se identifica que el enfoque de SQL directo dificulta el control del sistema, porque cualquier desviación del modelo (aunque sea pequeña) se convierte en un fallo operativo. Como consecuencia, se concluye que la generación directa de SQL necesita una evolución hacia un enfoque más controlado y verificable.

Conviene señalar que muchos de estos problemas ya aparecen en enfoques de generación directa de SQL sin mecanismos adicionales de contextualización. En el sistema desarrollado en esta aproximación, la incorporación de un módulo de recuperación de contexto permite mitigar en gran medida estas limitaciones, ya que el modelo recibe información adicional sobre el esquema de la base de datos y sobre el dominio AIS antes de generar la consulta.

Este contexto reduce la ambigüedad en la interpretación de la pregunta del usuario y mejora la correspondencia entre el lenguaje natural y las columnas reales de la base de datos. No obstante, aunque el uso de RAG mejora significativamente la estabilidad del sistema, las pruebas realizadas muestran que siguen existiendo situaciones en las que la generación directa de SQL introduce variabilidad o errores estructurales.

3.7 Tercera propuesta de arquitectura

Una vez validado el enfoque “consulta en lenguaje natural a ejecución determinista sobre datos estructurados”, se pretende evolucionar el mecanismo de generación de consultas para mejorar su robustez, consistencia y tolerancia a variaciones lingüísticas. Esta evolución se produce como respuesta directa a los fallos observados anteriormente.

3.7.1 Separación explícita entre datos dinámicos y estáticos

Para reducir errores derivados de ambigüedades estructurales, se introduce una separación explícita entre datos dinámicos y estáticos, tratándolos como fuentes diferenciadas desde el inicio del proceso, separando los datos en carpetas y aplicando filtros para que las consultas únicamente accedan a los datos de la carpeta correspondiente.

Esta decisión se justifica porque una parte significativa de los fallos iniciales se produce cuando el sistema intenta resolver una consulta sobre una tabla que no contiene el campo solicitado. Por ejemplo, consultas sobre dimensiones o calado no deben resolverse sobre el conjunto dinámico, ni consultas sobre trayectoria, velocidad o posición sobre el conjunto estático. Si el sistema no fuerza esta separación, el modelo puede asumir erróneamente que cualquier atributo está disponible en cualquier fuente.

A partir de esta mejora, el sistema incorpora reglas de selección de fuente basadas en intención: identifica términos, patrones y necesidades de la consulta para dirigir el procesamiento hacia el conjunto más apropiado. Esta separación reduce de forma directa el número de errores de “columna inexistente” y facilita que la generación de consultas sea más consistente, al limitar el espacio de búsqueda del modelo a un esquema más acotado.

3.7.2 Incorporación de validación estructural de consultas generadas

Una vez identificados los límites de la generación directa, el sistema incorpora una etapa de validación estructural previa a la ejecución. En este punto, la salida del modelo deja de considerarse una instrucción que se ejecuta sin comprobación y pasa a tratarse como una propuesta que debe cumplir condiciones mínimas antes de aplicarse a los datos.

Esta validación se centra en tres objetivos:

1. Coherencia con el esquema disponible: el sistema comprueba que las columnas solicitadas existen en la fuente seleccionada y que el tipo de operación es compatible con ellas.
2. Consistencia operativa: el sistema verifica que la consulta incluye los elementos necesarios para ejecutarse (selección, filtros cuando son necesarios, límites razonables, ordenación válida si se solicita).

3. Seguridad y control del coste: el sistema impone restricciones de ejecución, como límites máximos de resultados o prohibición de operaciones potencialmente costosas sin agregación o sin filtros, evitando respuestas lentas o inestables.

La consecuencia directa es que los fallos del modelo dejan de convertirse automáticamente en fallos del sistema: si el modelo propone una consulta incorrecta o incompleta, el sistema la detecta antes de ejecutarla y aplica correcciones o devuelve un error controlado, que posteriormente es expresado por el modelo en la respuesta generada al usuario.

Esta evolución del mecanismo de generación de consultas constituye una transición hacia una arquitectura más robusta, en la que el modelo de lenguaje mantiene su papel interpretativo, pero el sistema conserva el control determinista sobre las operaciones que finalmente se ejecutan sobre los datos AIS.

3.7.3 Introducción de un formato intermedio estructurado (JSON)

Tras identificar las limitaciones del enfoque basado en la generación directa de SQL, el sistema evoluciona hacia un modelo de dos etapas: el modelo de lenguaje deja de producir consultas ejecutables directamente y pasa a generar un plan intermedio estructurado, que posteriormente es interpretado y transformado en operaciones deterministas sobre los datos.

Esta modificación no implica abandonar el uso del modelo de lenguaje, sino redefinir su papel dentro de la arquitectura. En lugar de delegar completamente en el modelo la responsabilidad de producir una consulta final, el sistema le solicita que exprese la intención de la pregunta en un formato formal y validable, sobre el cual se pueden aplicar reglas de control antes de la ejecución.

3.7.3.1 Justificación del uso de un formato intermedio

La principal motivación para introducir un formato intermedio es la necesidad de reducir la dependencia directa entre la salida textual del modelo y la ejecución sobre los datos reales. Cuando el modelo genera SQL de forma libre, cualquier desviación en el formato puede provocar un fallo inmediato del sistema.

El uso de un formato estructurado como JSON [40] permite establecer un contrato explícito entre el modelo y el motor de ejecución. JSON se selecciona por varias razones técnicas:

- Es un formato ampliamente adoptado para el intercambio de datos estructurados.
- Permite validación automática mediante esquemas.
- Facilita la detección temprana de errores estructurales.
- Resulta fácilmente interpretable por el *backend* del sistema.

Además, el uso de salidas estructuradas en integraciones con modelos de lenguaje se encuentra alineado con las recomendaciones actuales en el desarrollo de aplicaciones basadas en LLMs, donde se prioriza la generación de respuestas formalizadas y verificables antes de su uso en procesos automáticos. La posibilidad de definir y validar esquemas estructurados mejora significativamente la robustez de la integración entre modelos y herramientas externas.

Desde el punto de vista conceptual, el formato intermedio actúa como una capa de abstracción entre la comprensión semántica y la ejecución analítica. El modelo expresa qué quiere hacer el usuario, mientras que el sistema decide cómo ejecutarlo realmente sobre los datos. Véase un ejemplo de un plan JSON intermedio generado para la consulta “¿Cuál es la última posición conocida del barco con MMSI 224581000 el día 9 de enero de 2023?” en la Figura 3-4.

```
{
  "source": "dynamic",
  "select": ["timestamp", "lat", "lon", "speed", "course"],
  "filters": {
    "mmsi": 224581000,
    "time_from": 1673222400,
    "time_to": 1673308800
  },
  "order_by": [
    {"col": "timestamp", "desc": true}
  ],
  "limit": 1,
  "agg": null
}
```

Figura 3-4 Ejemplo de plan JSON

3.7.3.2 Estructura del plan de ejecución intermedio

El plan intermedio se define como un objeto estructurado cuya finalidad es representar, de manera explícita y formal, la intención analítica de la consulta del usuario. Este objeto no contiene instrucciones ejecutables directas, sino una descripción de los elementos necesarios para construir la operación sobre los datos AIS.

En primer lugar, el plan incorpora un identificador de la fuente de datos, que permite determinar si la operación debe realizarse sobre el conjunto dinámico, el conjunto estático o, en determinados casos, sobre ambos. Esta decisión resulta fundamental para evitar errores derivados de la selección incorrecta de esquemas.

A continuación, el plan especifica las columnas que deben formar parte del resultado. Estas columnas no se interpretan como texto libre, sino como referencias que deben corresponderse con los campos reales del sistema. De esta forma, el modelo expresa qué información desea obtener, pero no determina directamente cómo se formula la consulta final.

El plan también incluye la definición de filtros, que permiten restringir el conjunto de datos a aquellos registros que cumplen determinadas condiciones. Estos filtros pueden referirse a identificadores concretos, rangos temporales, valores numéricos o categorías específicas. La representación estructurada de los filtros facilita su validación posterior y evita ambigüedades propias del lenguaje natural.

En aquellos casos en los que la consulta requiere operaciones estadísticas, el plan incorpora una descripción explícita de las agregaciones necesarias. Esto permite diferenciar claramente entre consultas de recuperación simple de registros y consultas de carácter analítico, como conteos o promedios.

Asimismo, el plan puede contener criterios de ordenación y un límite máximo de resultados, lo que introduce un control explícito sobre el tamaño y organización del conjunto devuelto. Este último aspecto resulta especialmente relevante en conjuntos AIS de gran volumen, donde una consulta sin límite podría producir salidas de tamaño excesivo.

Esta estructura permite que el modelo de lenguaje exprese la intención de la consulta de forma controlada, trasladando al sistema la responsabilidad de interpretar y ejecutar dicha intención mediante mecanismos deterministas.

3.7.3.3 Proceso de normalización y verificación del plan

Una vez que el modelo genera el plan en formato estructurado, no se procede inmediatamente a su ejecución, sino que se inicia un proceso de normalización y verificación orientado a garantizar la coherencia y viabilidad de este.

Durante la fase de normalización, el sistema revisa la estructura recibida y completa aquellos elementos que puedan estar ausentes pero que resulten necesarios para la ejecución. Por ejemplo, si el plan no incluye un límite explícito de resultados, se introduce un valor por defecto predefinido. De igual forma, si los nombres de columnas no coinciden exactamente con la nomenclatura interna del sistema, se aplican reglas de mapeo que alinean la intención expresada con el esquema real de los datos.

Posteriormente, se lleva a cabo la fase de verificación estructural. En esta etapa, el sistema comprueba que cada uno de los componentes del plan es compatible con el esquema seleccionado. Se valida que las columnas solicitadas existen efectivamente en la fuente indicada y que las operaciones a realizar son coherentes con el tipo de dato correspondiente, evitando ejecuciones potencialmente costosas o innecesarias.

Si durante esta verificación se detecta una incoherencia menor susceptible de corrección automática, el sistema ajusta el plan antes de continuar. En cambio, si se identifica una inconsistencia estructural significativa, el proceso se detiene y se devuelve un mensaje de error controlado al modelo, evitando que el fallo se propague a la capa de ejecución. Ese fallo se muestra al usuario en la redacción de la respuesta por parte del LLM.

Este procedimiento introduce una capa de seguridad esencial en la arquitectura. El plan generado por el modelo deja de ser una instrucción ejecutable directa y pasa a convertirse en una propuesta que debe superar una validación. Como resultado, el sistema incrementa su robustez frente a salidas impredecibles del modelo y reduce de manera significativa el riesgo de errores en el procesamiento de datos AIS.

Si la salida no puede ser interpretada como un objeto JSON válido, el sistema no continúa con la ejecución, sino que devuelve un error gestionado o solicita una reformulación implícita del plan.

Este comportamiento resulta esencial para evitar fallos en cadena dentro del *pipeline*. En lugar de permitir que una salida incorrecta del modelo se traduzca en una excepción en el motor de análisis, el sistema interrumpe el flujo en la etapa más temprana posible.

3.7.4 Optimización del procesamiento de datos: Polars + Parquet

Una vez consolidado el mecanismo de generación estructurada de planes y reforzada la capa de validación, el siguiente punto crítico del sistema se centra en la optimización del procesamiento de datos. Durante las pruebas realizadas con volúmenes reales de información AIS utilizando DuckDB, se observa que los tiempos en la recuperación de la información son excesivos.

Para mejorar este comportamiento, se adopta una evolución en la capa de procesamiento orientada a aumentar el control sobre la ejecución y a adaptar mejor la arquitectura a las características de los datos AIS. Esta optimización se articula en dos decisiones complementarias: por un lado, la sustitución del enfoque inicial basado en DuckDB por un procesamiento mediante *Polars* [41]; y, por otro, la adopción de *Parquet* [42] como formato principal de almacenamiento para la fase de ejecución.

3.7.4.1 Sustitución del motor de análisis: de DuckDB a Polars

Con el objetivo de aumentar el control sobre el proceso de ejecución, el sistema adopta un enfoque basado en *DataFrames* utilizando *Polars*. *Polars* es una biblioteca orientada a procesamiento de datos de alto rendimiento implementada sobre *Rust* [43] y basada en *Apache Arrow* [44], lo que le permite ejecutar operaciones vectorizadas y paralelizadas de forma eficiente.

Además, a diferencia del enfoque basado en la generación dinámica de consultas SQL, el modelo *DataFrame-first* construye las operaciones analíticas directamente en el código del programa. En lugar de generar una consulta SQL como texto y ejecutarla, el sistema traduce la consulta a una secuencia de operaciones sobre *DataFrames* utilizando la librería *Polars*. De este modo, cada paso del procesamiento puede controlarse y validarse antes de ejecutarse sobre los datos.

Por otro lado, *Polars* incorpora un modo de ejecución *lazy*, en el que las transformaciones no se materializan inmediatamente, sino que el sistema construye un plan de ejecución optimizado antes de evaluar el resultado. Este comportamiento mejora el rendimiento en escenarios donde se encadenan múltiples filtros, agregaciones y selecciones de columnas, como ocurre frecuentemente en consultas AIS.

La transición hacia *Polars* permite, por tanto, reforzar la separación entre interpretación y ejecución: el modelo genera un plan declarativo, y el *backend* lo traduce en operaciones *DataFrame* explícitas, controladas y verificables. Esto reduce la probabilidad de fallos inesperados y simplifica el mantenimiento del sistema.

3.7.4.2 Adopción de Parquet como formato de almacenamiento principal

En paralelo a la sustitución del motor de análisis, el sistema adopta el formato *Parquet* como mecanismo principal de almacenamiento de los datos AIS para la fase de ejecución. En el enfoque inicial basado en DuckDB, los datos se almacenaban en una base de datos en formato *.db*, sobre la cual se ejecutaban consultas SQL. En la nueva arquitectura, los datos se transforman y almacenan en archivos *Parquet* que posteriormente son procesados directamente por el motor *Polars*.

Parquet es un formato de almacenamiento diseñado para el análisis eficiente de grandes volúmenes de datos. A diferencia de un modelo basado en bases de datos tradicionales, donde las consultas se ejecutan a través de un motor SQL sobre tablas completas, el formato *Parquet* permite acceder únicamente a las columnas necesarias para cada operación. Este comportamiento resulta especialmente relevante en el contexto de los datos AIS, donde muchas consultas requieren únicamente un subconjunto reducido de variables.

Otra ventaja significativa del formato *Parquet* es su capacidad de compresión y su compatibilidad con diversas técnicas de optimización del acceso a datos. Estas características permiten reducir el volumen de información leído durante la ejecución de operaciones analíticas, por ejemplo, cuando se aplican filtros sobre rangos temporales o identificadores específicos de buques. Como consecuencia, el tiempo de acceso a los datos disminuye y la eficiencia global del sistema mejora de forma notable.

Desde el punto de vista arquitectónico, el uso de *Parquet* se integra de forma natural con *Polars*, que ofrece soporte nativo y optimizado para este formato. Esta combinación permite que el motor de procesamiento opere directamente sobre estructuras de columnas, reduciendo la sobrecarga asociada a capas adicionales de abstracción y proporcionando un mayor control sobre la ejecución de las operaciones analíticas.

En conjunto, esta transición hacia un enfoque basado en *Polars* y *Parquet* no supone un abandono del análisis estructurado, sino una adaptación del motor de ejecución hacia un modelo más eficiente y flexible, mejor alineado con las características de los datos AIS y con las necesidades analíticas del sistema.

3.7.5 Arquitectura del proceso de consulta

Tras la incorporación del plan intermedio estructurado, la implementación de mecanismos de validación y la optimización del procesamiento mediante *Polars* y *Parquet*, el sistema consolida una arquitectura madura y coherente en la que cada componente cumple una función claramente delimitada (véase Figura 3-5). El resultado es un *pipeline* de consulta híbrido que combina comprensión semántica flexible con ejecución determinista y controlada sobre datos estructurados.

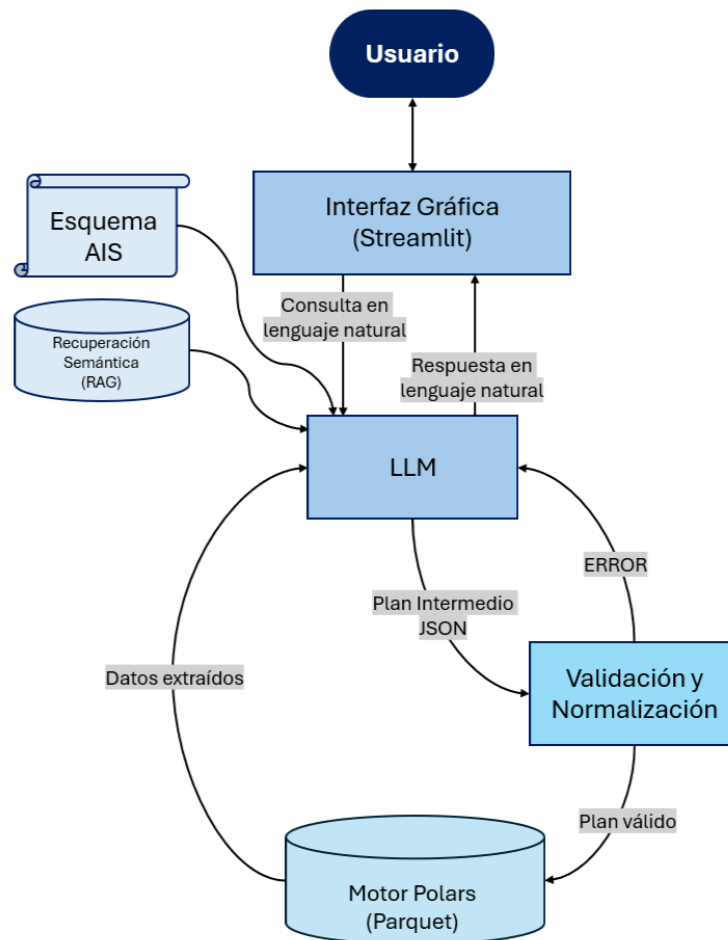


Figura 3-5 Representación del flujo de una consulta en la tercera propuesta de arquitectura

El flujo completo de una consulta comienza cuando el usuario introduce una pregunta en lenguaje natural a través de la interfaz gráfica desarrollada con Streamlit. Esta entrada constituye el punto inicial del procesamiento y se transmite al módulo de lógica de aplicación.

El sistema analiza la consulta y, cuando procede, realiza una recuperación de contexto semántico desde el índice vectorial. Este contexto no contiene datos numéricos del tráfico marítimo, sino conocimiento estructurado del dominio AIS que ayuda al modelo a interpretar correctamente la intención del usuario.

A continuación, el modelo de lenguaje genera un plan intermedio en formato JSON que describe la intención analítica de la consulta. Este plan no es ejecutable directamente, sino que representa una especificación declarativa de la operación deseada.

El sistema recibe dicho plan y lo somete a un proceso de normalización y verificación estructural. En esta etapa se corrigen inconsistencias menores, se validan columnas y se comprueba la coherencia global de la operación. Solo si el plan supera esta validación se procede a la fase de ejecución. Posteriormente, el *backend* traduce el plan validado en operaciones explícitas sobre estructuras *DataFrame* utilizando *Polars*, accediendo a los datos almacenados en formato *Parquet*.

Una vez obtenido el resultado estructurado real, el modelo de lenguaje interviene nuevamente para generar una respuesta textual clara y comprensible para el usuario. Es importante destacar que esta respuesta se basa exclusivamente en los datos devueltos por la ejecución determinista, garantizando coherencia entre la información presentada y el contenido real de los conjuntos AIS.

3.7.6 Evaluación de la arquitectura propuesta

Una vez consolidada la arquitectura descrita en el apartado anterior, se realizan diversas pruebas de funcionamiento con el objetivo de evaluar de forma preliminar su comportamiento en escenarios de consulta reales. Estas pruebas permiten comprobar que el sistema es capaz de interpretar consultas en lenguaje natural, generar planes estructurados y ejecutar operaciones analíticas sobre los datos AIS de forma correcta. No obstante, durante este proceso se identifican ciertas limitaciones relacionadas principalmente con el rendimiento del sistema y la consistencia en la interpretación de algunas consultas.

En particular, se observa que, aunque las respuestas generadas resultan, en general, coherentes y correctas desde el punto de vista analítico, los tiempos de respuesta son elevados para un sistema concebido como herramienta interactiva de consulta. Este análisis conduce a la realización de pruebas adicionales orientadas a identificar el origen de dichas limitaciones y a evaluar posibles alternativas de mejora.

3.7.6.1 Resultados de las primeras pruebas de consulta

Desde el punto de vista funcional, el sistema muestra un comportamiento satisfactorio. En la mayoría de los casos, el modelo es capaz de interpretar correctamente la intención del usuario, generar un plan coherente y producir resultados adecuados a partir de los datos almacenados (véase Figura 3-6). Este comportamiento confirma la viabilidad del enfoque híbrido basado en interpretación semántica mediante modelos de lenguaje y ejecución determinista sobre datos estructurados.

Consulta de datos AIS

Consulta

¿Cuál es la última posición del MMSI 255739000?

Enviar Limpiar

Consulta: ¿Cuál es la última posición del MMSI 255739000?

Respuesta: La última posición del MMSI 255739000 es:

• Ubicación: Latitud 59.733482, Longitud 10.23651 • Velocidad: 0 nudos • Rumbo: 185,6 grados • Hacia: 98 grados

La última posición del MMSI 255739000 se encuentra en los datos dinámicos y cumple con las reglas de la plantilla "últimas posiciones de un MMSI".

Consulta de datos AIS

Consulta

a qué velocidad suelen ir los barcos tipo Tanker?

Enviar Limpiar

Consulta: a qué velocidad suelen ir los barcos tipo Tanker?

Respuesta: Los barcos tipo Tanker suelen navegar con velocidades entre 10 a 15 nudos.

Figura 3-6 Ejemplos de las consultas

Sin embargo, durante estas pruebas se detecta que los tiempos de respuesta resultan significativamente elevados. En consultas simples o de carácter directo, como aquellas orientadas a recuperar valores concretos o realizar agregaciones básicas, los tiempos de generación de respuesta se sitúan habitualmente entre 30 y 40 segundos. En consultas de carácter analítico, que implican operaciones más complejas o mayor procesamiento de datos, los tiempos pueden aumentar hasta valores comprendidos aproximadamente entre un minuto y medio y dos minutos.

Si bien tiempos en torno a los 40 segundos no resultan necesariamente excesivos en el contexto de consultas analíticas complejas, sí constituyen una limitación importante en consultas básicas, donde se espera una respuesta prácticamente inmediata. Este comportamiento motiva la necesidad de analizar con mayor detalle el *pipeline* interno del sistema para identificar el origen de la etapa que demanda tiempo de procesamiento.

3.7.6.2 Identificación del cuello de botella

Con el objetivo de comprender el comportamiento observado, se analiza el tiempo empleado en cada una de las etapas que componen el flujo completo de una consulta.

El *pipeline* de procesamiento se divide en varias fases principales: recuperación de contexto semántico, generación del plan estructurado por parte del modelo de lenguaje, normalización y validación del plan, ejecución de la operación analítica sobre los datos y generación de la respuesta final.

El análisis de tiempos permite comprobar que aproximadamente la mitad del tiempo total de generación de la respuesta se concentra en las fases relacionadas con la creación y validación del plan estructurado en formato JSON (véase Figura 3-7). Estas etapas implican la intervención directa del modelo de lenguaje, así como un proceso posterior de verificación y corrección del plan antes de su ejecución.

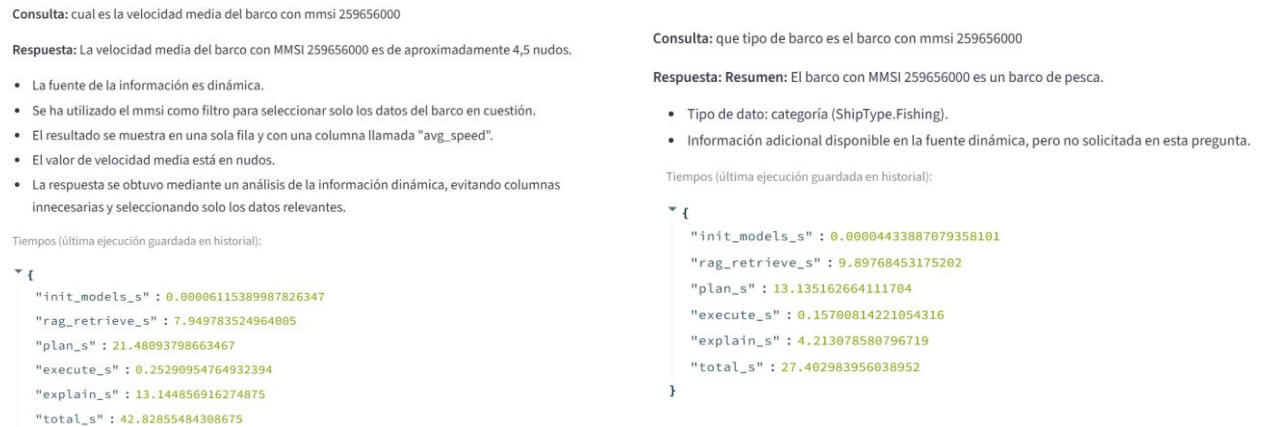


Figura 3-7 Ejemplos de las consultas con tiempos desglosados

Este comportamiento tiene una explicación directa en la naturaleza del proceso. La generación del plan requiere que el modelo interprete la intención de la consulta en lenguaje natural, identifique los elementos analíticos relevantes y produzca una representación estructurada compatible con el sistema. Posteriormente, el *backend* debe verificar que dicha representación sea coherente con el esquema de datos y con las reglas de ejecución definidas en el sistema.

3.8 Arquitectura definitiva del sistema

3.8.1 Enfoque determinista de consulta

Para determinar el coste real de la ejecución analítica sobre los datos AIS almacenados en formato *Parquet*, se desarrolla un script experimental independiente basado en un enfoque completamente determinista (véase un ejemplo en la Figura 3-8). En este prototipo, las consultas se definen directamente mediante operaciones analíticas sobre los datos, sin intervención del modelo de lenguaje ni del sistema de recuperación de contexto.

```
mmsi = 367784630
path = "/app/storage/parquet/dynamic/*.parquet"

df = (
  pl.scan_parquet(path)
  .filter((pl.col("mmsi") == mmsi) & (pl.col("speed") >= 0.5) & (pl.col("speed") <= 40))
  .select(pl.col("speed").mean().alias("avg_speed"))
  .collect()
)

print(f"Velocidad media del MMSI {mmsi}: {df['avg_speed'][0]:.2f} kn")
```

Figura 3-8 Ejemplo de consulta sobre los datos con *Polars* + *Parquet*

El objetivo de esta prueba no consiste en sustituir el sistema de interpretación en lenguaje natural, sino en medir de forma precisa el tiempo necesario para ejecutar consultas típicas sobre los datos AIS utilizando exclusivamente el motor analítico basado en *Polars* y los archivos almacenados en formato *Parquet*.

Los resultados obtenidos muestran un comportamiento claramente diferente al observado en el sistema completo. La mayoría de las consultas deterministas se ejecutan en menos de un segundo, incluso cuando implican operaciones de filtrado, agregación o transformación de datos. En aquellos casos en los que el tiempo supera ligeramente este valor, el incremento es mínimo y se mantiene dentro de un rango claramente inferior al observado en la arquitectura basada en generación de planes mediante LLM.



Figura 3-9 Ejemplos de las consultas con tiempos desglosados

Estos resultados permiten confirmar que el procesamiento analítico de los datos AIS mediante *Polars* resulta altamente eficiente y que el principal coste computacional del sistema no se encuentra en la fase de ejecución de las consultas, sino en la etapa previa de interpretación y generación del plan estructurado.

No obstante, este enfoque determinista presenta una limitación evidente cuando se considera como sistema de consulta completo. Al prescindir completamente de la intervención del modelo de lenguaje, el sistema pierde la capacidad de interpretar consultas formuladas en lenguaje natural y queda restringido a un conjunto de operaciones previamente definidas. Esto implicaría que el usuario debe conocer de antemano qué operación desea ejecutar y cómo debe formularla, lo que elimina la flexibilidad que motiva el diseño de la plataforma desarrollada en este trabajo, orientada precisamente a permitir la exploración de los datos mediante lenguaje natural.

Como consecuencia, aunque el enfoque determinista demuestra ser extremadamente eficiente desde el punto de vista computacional, no resulta adecuado como solución completa para un sistema de consulta inteligente.

A partir de estas observaciones surge la necesidad de diseñar una arquitectura que combine las ventajas de ambos enfoques: la capacidad interpretativa proporcionada por los modelos de lenguaje y la eficiencia del procesamiento determinista sobre los datos AIS. Esta motivación conduce al diseño de una arquitectura mixta que se describe en el apartado siguiente.

3.8.2 Arquitectura mixta de consulta

A partir de los resultados obtenidos en el apartado anterior, el sistema evoluciona hacia una arquitectura mixta en la que se combinan dos enfoques complementarios: por un lado, una ruta puramente determinista orientada a consultas directas y de baja ambigüedad; y por otro, una ruta de análisis en la que se mantiene el uso del modelo de lenguaje para interpretar objetivos complejos y redactar respuestas, pero delegando siempre la ejecución real de cálculos en herramientas deterministas.

El objetivo de esta evolución es doble. En primer lugar, reducir drásticamente el tiempo de respuesta para preguntas básicas, evitando el coste asociado a la generación y validación de un plan complejo cuando no es necesario. En segundo lugar, mantener la capacidad del sistema para resolver preguntas de análisis y consultas abiertas, incorporando mecanismos de control que limiten ambigüedades y garanticen que los resultados numéricos proceden únicamente de la ejecución sobre los datos AIS.

La Figura 3-10 muestra de forma esquemática la arquitectura general del sistema tras esta evolución. En ella se representa el flujo completo de procesamiento de una consulta, desde su introducción por parte del usuario hasta la generación de la respuesta final. El diagrama permite visualizar cómo el sistema decide la ruta de procesamiento de cada consulta y cómo se integran los distintos componentes encargados de la interpretación, ejecución analítica y generación de la respuesta. En los apartados siguientes se describe con mayor detalle el funcionamiento de cada una de las fases que componen este flujo.

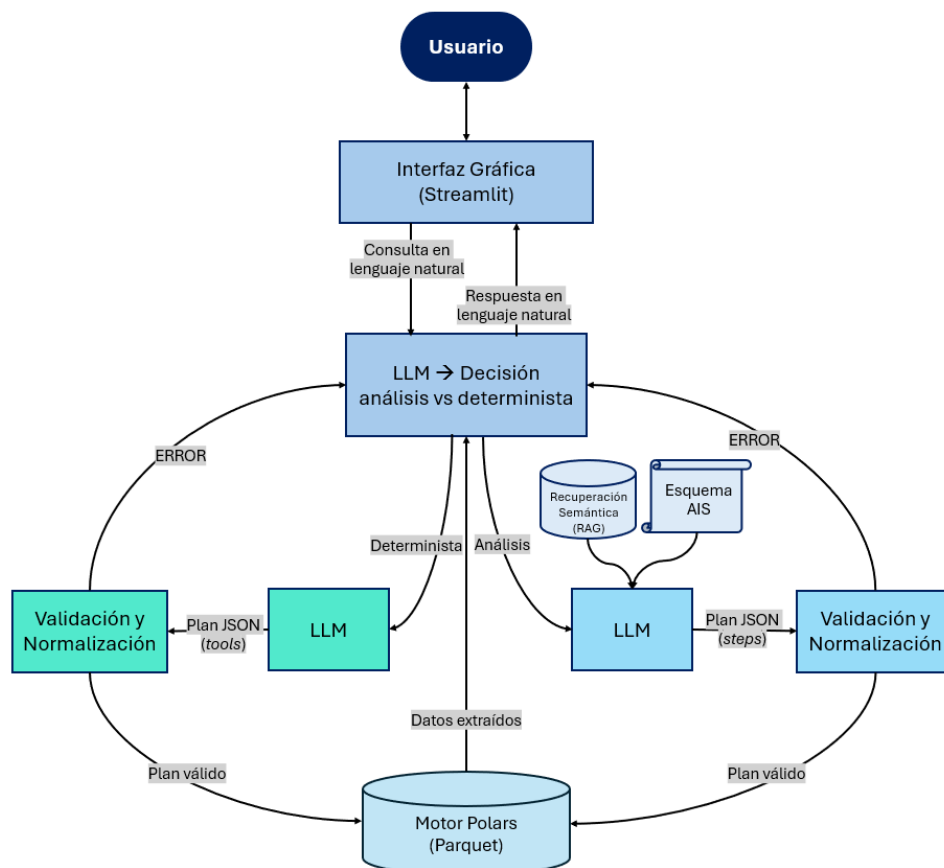


Figura 3-10 Representación del flujo de una consulta en la propuesta definitiva de arquitectura

3.8.3 Principio general de la arquitectura

La arquitectura desarrollada se apoya en una idea central: el modelo de lenguaje no realiza los cálculos sobre los datos AIS, sino que actúa únicamente como componente de interpretación y de

redacción. De este modo, se evita que el sistema dependa de capacidades generativas del modelo para producir valores numéricos, agregaciones o comparaciones que deben obtenerse exclusivamente a partir de los datos reales.

Bajo este planteamiento, toda operación cuantitativa se ejecuta mediante procedimientos deterministas implementados sobre el motor analítico, mientras que el modelo de lenguaje interviene en otros momentos del flujo de la consulta. En primer lugar, participa en la interpretación de la consulta del usuario y en la generación de un plan estructurado de ejecución. En segundo lugar, se utiliza para redactar una explicación textual a partir de resultados ya calculados. Esta separación reduce el riesgo de alucinaciones, mejora la trazabilidad de las respuestas y permite que la exactitud dependa del código ejecutado y no de la generación textual del modelo.

Además, se consigue mitigar de forma considerable el cuello de botella descrito en el apartado 3.7.6.2. Como se puede observar en la Figura 3-11, el evitar la recuperación de contexto para una consulta sencilla reduce el tiempo de respuesta en más de 10 segundos.



Figura 3-11 Comparativa de tiempos para cada arquitectura

3.8.4 Clasificación de consultas

Cuando el usuario introduce una pregunta en la interfaz, el sistema no ejecuta directamente una operación sobre los datos. Antes de cualquier acceso al *dataset*, se realiza una fase de clasificación cuyo objetivo es decidir qué tipo de problema plantea la consulta y, por tanto, qué camino de ejecución debe seguirse.

- La primera posibilidad es la **ruta determinista**, pensada para consultas cuyo significado puede traducirse de forma relativamente directa a una o varias operaciones cerradas. En este grupo entrarían, por ejemplo, consultas de recuperación de atributos de un buque, cálculo de máximos o promedios, extracción de trayectorias, rankings o agregaciones sencillas. En estos casos el sistema intenta identificar con precisión qué herramienta debe ejecutarse y con qué argumentos.
- La segunda posibilidad es la **ruta analítica**, reservada para preguntas en las que no basta con una única operación cerrada, sino que es necesario descomponer el problema en varios pasos, combinar resultados intermedios o redactar una interpretación final más elaborada. En esta ruta el sistema no responde solo con una tabla o con un valor, sino que genera una pequeña secuencia de operaciones y posteriormente construye una explicación textual apoyada en esos resultados.

En la práctica, la clasificación de la consulta es realizada por el modelo de lenguaje, que analiza la pregunta del usuario y determina la categoría más adecuada. No obstante, esta decisión no se toma de forma aislada. El modelo se apoya en distintos elementos del sistema que facilitan su interpretación.

Utiliza contexto adicional recuperado mediante el sistema RAG, que proporciona información relevante sobre la estructura de los datos y las capacidades del sistema.

3.8.5 *Papel del LLM y sistema RAG*

El modelo de lenguaje y el sistema RAG cumplen funciones complementarias. El LLM se utiliza como mecanismo de interpretación de la consulta y como redactor de la respuesta final. Su función no es operar sobre el *dataset* ni calcular métricas, sino transformar una petición en lenguaje natural en una estructura formal manejable por el sistema.

El RAG, por su parte, no se emplea para obtener valores numéricos del conjunto de datos AIS. Su función es aportar contexto documental al modelo de lenguaje. Ese contexto se usa en dos momentos distintos. Primero, durante el enrutado, para ayudar al modelo a identificar mejor la intención de la consulta y seleccionar la familia de herramientas adecuada. Segundo, en las consultas analíticas, para proporcionar soporte contextual adicional en la redacción de la explicación final. En consecuencia, el RAG actúa como una memoria externa de conocimiento descriptivo, mientras que los resultados cuantitativos siguen obteniéndose exclusivamente mediante ejecución determinista sobre los datos.

Este diseño es importante porque impide que el sistema mezcle dos planos distintos: el plano del conocimiento contextual y el plano del cálculo sobre datos reales. El primero ayuda a interpretar y explicar; el segundo produce los resultados verificables.

3.8.6 *Las herramientas deterministas y su organización por familias*

El núcleo operativo del sistema está formado por un conjunto cerrado de herramientas deterministas. Estas funciones encapsulan operaciones analíticas recurrentes sobre los datos AIS y permiten traducir la consulta del usuario a acciones concretas sobre el motor de datos. Además, el sistema no restringe la ejecución a esta biblioteca de operaciones conocidas y validadas, también permite la generación libre de código para consultas que necesiten usar funciones adicionales.

Estas herramientas pueden agruparse por familias, en función del tipo de información que manejan y de la lógica común que comparten:

- **Herramientas de perfil estático:** Incluyen operaciones orientadas a recuperar o calcular atributos relativamente estables de los buques, como IMO, calado, tipo de buque o dimensiones.
- **Herramientas de comportamiento dinámico:** Trabajan sobre la serie temporal AIS y permiten calcular métricas como número de mensajes, velocidad media, velocidad máxima o trayectoria de un buque en una ventana temporal.
- **Herramientas mixtas o de fusión:** Sirven para combinar información dinámica y estática cuando la consulta lo requiere.
- **Herramientas de control y depuración:** No responden directamente a una consulta operativa del usuario, sino que aportan información sobre la calidad del cálculo o del propio *pipeline*.

Esta organización por familias no es solo una clasificación conceptual. También ayuda al modelo y al validador a acotar qué herramienta resulta razonable para cada consulta. En la práctica, el sistema no selecciona entre funciones aisladas, sino entre grupos de operaciones que responden a problemas recurrentes: consultar atributos, agregar medidas, ordenar resultados, reconstruir trayectorias o combinar tablas.

Las herramientas actúan como el mecanismo mediante el cual el sistema transforma la intención expresada en lenguaje natural en una operación analítica concreta sobre el *dataset*. De este modo, el modelo se encarga de identificar la operación adecuada, mientras que la herramienta correspondiente ejecuta el cálculo y obtiene el resultado real.

3.8.7 Planes JSON y diferencias entre ambas rutas

Una vez interpretada la consulta, el sistema no ejecuta directamente instrucciones en lenguaje natural. En su lugar, genera un plan JSON con un esquema cerrado. Este plan constituye una representación intermedia entre la pregunta del usuario y las herramientas deterministas del sistema. Su uso permite tratar la salida del modelo no como una orden libre, sino como una propuesta estructurada que todavía debe ser normalizada y validada antes de ejecutarse.

En la ruta determinista, el plan JSON es compacto. Normalmente contiene tres elementos: la ruta seleccionada, la herramienta a utilizar y el conjunto de argumentos necesarios para ejecutarla. En esencia, este tipo de plan responde a la lógica “qué herramienta usar” y “con qué parámetros hacerlo”. Es, por tanto, un plan de acción simple, adecuado para consultas que pueden resolverse con una única operación o con una operación compuesta pero cerrada.

En la ruta analítica, en cambio, el plan JSON tiene una estructura más compleja. Además de indicar que la consulta pertenece a la ruta de análisis, incorpora un objetivo textual y una secuencia ordenada de pasos. Cada paso especifica una herramienta y sus argumentos. La diferencia fundamental es que aquí el sistema no está describiendo una sola operación, sino una pequeña estrategia de resolución. Es decir, el JSON deja de representar una llamada individual y pasa a representar un flujo de trabajo analítico.

Dicho de forma sencilla, en la ruta determinista el JSON responde a la pregunta: “qué operación concreta hay que ejecutar”. En la ruta analítica responde a otra distinta: “qué conjunto de operaciones hay que encadenar para responder bien a la consulta”. Esta distinción resulta importante porque refleja que ambas rutas no solo producen salidas diferentes, sino que modelan problemas de distinta complejidad.

3.8.8 Validación del plan y diferencias según la ruta

La validación del plan es una de las partes más importantes de la arquitectura, ya que convierte el plan generado por el modelo en una estructura segura y ejecutable. El sistema no acepta cualquier JSON producido por el LLM. Antes de ejecutar una sola operación, comprueba que la ruta sea válida, que la herramienta pertenezca al conjunto permitido, que las métricas solicitadas estén soportadas, que los límites no superen el máximo fijado y que los argumentos sean coherentes con la semántica de la herramienta correspondiente.

En la ruta determinista, esta validación se centra sobre todo en verificar la correspondencia entre una herramienta concreta y sus argumentos. Por ejemplo, se comprueba que una agregación dinámica use una métrica válida, que un ranking incluya un límite acotado, que una trayectoria disponga de un MMSI o que un filtro espacial tenga un formato correcto. En otras palabras, el sistema comprueba que la operación pedida existe y que puede ejecutarse sin ambigüedad.

En la ruta analítica, la validación es más exigente, porque no basta con revisar una única operación. Aquí el sistema debe validar una lista de pasos. Por ello, además de verificar cada herramienta individualmente, también comprueba que la secuencia resultante tenga sentido, que no haya pasos no soportados y que el análisis no se convierta en una cadena arbitraria de acciones generadas por el modelo. El plan analítico, por tanto, se filtra paso a paso antes de ser aceptado.

A esto se suma un segundo mecanismo de control: las reglas deterministas de reescritura. Si la intención del usuario está expresada con claridad, el sistema puede imponer el uso de una herramienta concreta, aunque el modelo haya propuesto otra. Esto aparece, por ejemplo, en consultas de tipo “top”, “ranking” o máximos, donde el validador fuerza determinadas herramientas y limita así la variabilidad del LLM. De este modo, el modelo propone, pero el validador conserva la última palabra sobre lo que realmente se ejecuta.

3.8.9 Función de los documentos en la arquitectura

Los documentos, almacenados en formato *.md*, forman parte de la base documental empleada por el sistema RAG. Su función no es almacenar los datos AIS ni sustituir a las tablas analíticas, sino proporcionar conocimiento auxiliar que ayuda al modelo en tareas de análisis, clasificación y redacción (*análisis.md*, *routing.md* y *writing.md*, respectivamente). El propio código prevé su ingesta junto con otros formatos documentales y los incorpora al índice vectorial utilizado por el componente RAG.

Dentro de la arquitectura, estos documentos pueden cumplir varios papeles. Por un lado, sirven para describir el dominio y el significado de ciertos conceptos o campos del *dataset*, de modo que el modelo entienda mejor qué quiere decir el usuario cuando formula la consulta. Por otro, pueden recoger guías de enrutado, criterios de redacción o ejemplos de interpretación, de forma que el sistema disponga de una base textual para decidir con mayor precisión entre distintas herramientas o para explicar mejor los resultados.

Además, el código muestra que el contexto recuperado no se utiliza de manera uniforme, sino que puede filtrarse según el rol asignado al documento, diferenciando entre uso para clasificación, para análisis o para redacción. Esto significa que no todos los *.md* cumplen exactamente la misma función: algunos están orientados a ayudar al enrutado inicial, otros a enriquecer la respuesta analítica y otros a apoyar la generación textual final. Desde el punto de vista de diseño, esta separación resulta útil porque evita mezclar en un único repositorio documental contenidos con finalidades distintas.

3.8.10 Gestión temporal basada en el dataset

Otro aspecto importante de la arquitectura es la gestión de las ventanas temporales utilizadas en las consultas. En lugar de basarse directamente en el reloj del sistema, las consultas que hacen referencia a periodos recientes se interpretan con respecto al máximo *timestamp* disponible en el *dataset* dinámico.

Este enfoque permite que expresiones como “últimos días” o “última semana” se interpreten de manera coherente incluso cuando el conjunto de datos utilizado corresponde a periodos históricos. De este modo, las consultas se formulan siempre con respecto al intervalo temporal realmente cubierto por los datos disponibles, evitando inconsistencias derivadas de utilizar referencias temporales basadas en la fecha actual del sistema.

Cabe destacar que esta decisión se introduce principalmente con el objetivo de facilitar las pruebas realizadas durante el desarrollo del sistema, ya que los datos AIS utilizados en esta fase corresponden a registros históricos. En un entorno operativo donde el sistema procese datos recientes o incluso flujos de datos en tiempo real, esta adaptación no sería necesaria, puesto que el propio *dataset* reflejaría automáticamente el periodo temporal más actual.

Precisamente por este motivo, anclar las ventanas temporales al máximo *timestamp* del *dataset* permite que el sistema mantenga un comportamiento coherente independientemente de si los datos analizados pertenecen a periodos históricos o a información reciente en tiempo real.

3.8.11 Ejecución determinista, control de calidad y presentación final

Una vez validado el plan, la ejecución se realiza siempre mediante procedimientos deterministas sobre los datos AIS almacenados en *Parquet* y procesados con *Polars*. Esta elección permite aprovechar el procesamiento columnar y la ejecución eficiente sobre grandes volúmenes de datos, manteniendo tiempos de respuesta reducidos y un comportamiento reproducible.

Dentro de esta ejecución se integran también reglas de control de calidad sobre determinadas magnitudes. En particular, para las consultas relacionadas con velocidad se descartan registros incompatibles con el objetivo analítico, como valores prácticamente nulos al calcular velocidades medias o velocidades excesivamente altas que no resultan plausibles para el tipo de tráfico estudiado. Estas

reglas no las aplica el modelo, sino que forman parte de la propia lógica determinista de las herramientas, por lo que se ejecutan siempre bajo las mismas condiciones.

Finalmente, la presentación del resultado se realiza mediante el modelo de lenguaje en ambas rutas de ejecución. Una vez que las operaciones deterministas sobre el conjunto de datos han sido completadas, los resultados obtenidos se proporcionan al LLM junto con el contexto documental recuperado mediante el sistema RAG. A partir de esta información, el modelo genera una respuesta final en lenguaje natural dirigida al usuario.

En la ruta determinista, el modelo recibe directamente los valores o estructuras de datos producidas por la herramienta correspondiente y se encarga de formular la respuesta de forma clara y comprensible, sin modificar los resultados calculados. En la ruta analítica, el proceso es similar, aunque los resultados pueden proceder de una secuencia de operaciones encadenadas que conforman el análisis solicitado.

De este modo, ambas rutas comparten un mismo principio de funcionamiento: el modelo de lenguaje actúa como interfaz de comunicación y redacción de la respuesta final, mientras que la obtención de los datos y el cálculo de los resultados se realizan siempre mediante procedimientos deterministas sobre el motor analítico. Esta separación permite aprovechar las capacidades expresivas del modelo generativo sin comprometer la fiabilidad de los resultados numéricos.

4 RESULTADOS OBTENIDOS Y VALIDACIÓN

El objetivo de este capítulo es evaluar el comportamiento de la plataforma desarrollada ante distintos tipos de consultas formuladas en lenguaje natural. La evaluación se centra en analizar cómo responde el sistema cuando se enfrenta a preguntas de complejidad creciente, observando tanto la capacidad de interpretar correctamente la consulta como la eficiencia en la obtención de los resultados.

4.1 Interfaz de usuario

Para ilustrar cómo se materializa esta arquitectura desde el punto de vista del usuario, la Figura 4-1 muestra la interfaz gráfica desarrollada para el sistema. A través de esta interfaz, el usuario puede formular consultas en lenguaje natural y visualizar las respuestas generadas a partir de los resultados obtenidos sobre los datos AIS.

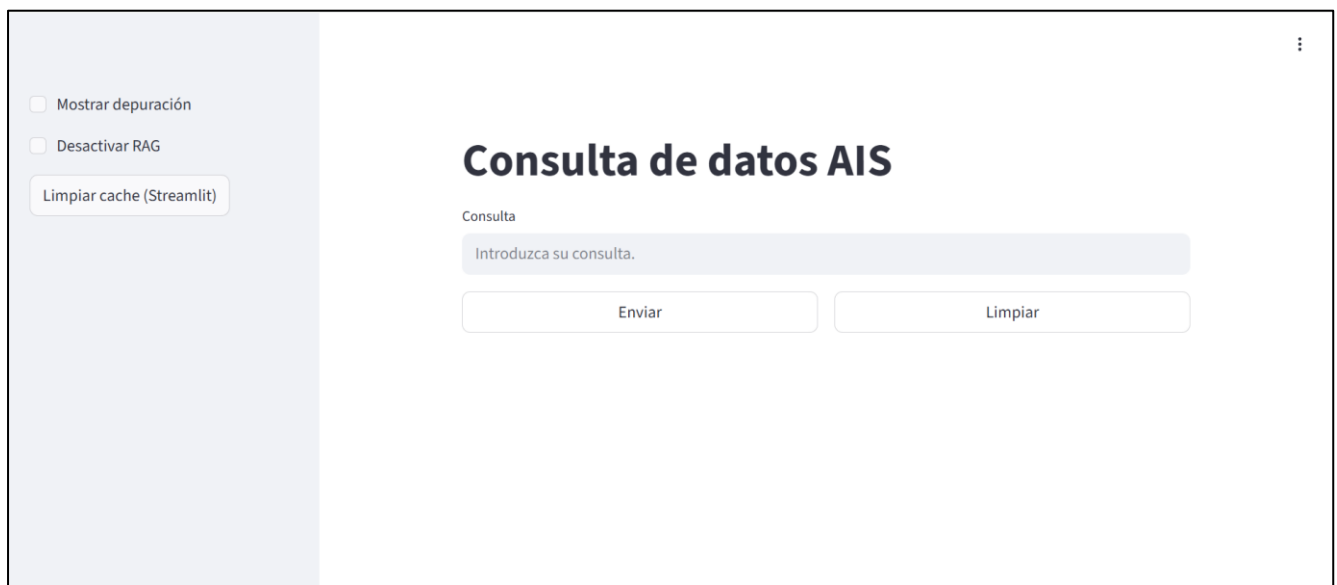


Figura 4-1 Presentación de la interfaz gráfica

4.2 Metodología de la evaluación

4.2.1 Clasificación de las consultas

Las pruebas experimentales se organizan en función del tipo de consulta realizada sobre los datos AIS, estableciendo una clasificación que permite estudiar el rendimiento del sistema en diferentes escenarios de uso. Esta clasificación no responde únicamente a criterios funcionales, sino también al

nivel de procesamiento necesario para resolver cada consulta. De este modo, se distinguen cuatro categorías principales:

- **Consultas de atributos:** Son consultas directas que buscan obtener el valor de un campo concreto a partir de un identificador o condición simple. Normalmente implican operaciones de filtrado sobre una única tabla y requieren poco procesamiento. Un ejemplo de este tipo de consulta sería obtener la velocidad media registrada para un buque concreto identificado por su MMSI.
- **Consultas de listados:** En este caso, el objetivo es obtener un conjunto de elementos que cumplen determinadas condiciones, generalmente acompañado de operaciones de ordenación o agregación. Este tipo de consultas requiere recorrer un volumen mayor de datos y realizar operaciones sobre conjuntos, como la identificación de los buques más rápidos en una zona determinada.
- **Consultas de fusión de datos:** Estas consultas requieren combinar información procedente de diferentes fuentes o tablas del sistema AIS. En este trabajo, esto implica principalmente la integración de información dinámica de navegación con datos estáticos de los buques. Este tipo de operaciones introduce una mayor complejidad, ya que es necesario realizar procesos de unión entre conjuntos de datos antes de obtener el resultado final.
- **Consultas analíticas:** Representan el nivel más alto de complejidad dentro de la evaluación. En este caso, la consulta no busca únicamente recuperar datos, sino analizar el comportamiento de los buques a partir de la información disponible. Este análisis se ha enfocado en la identificación de patrones de navegación habituales, la interacción entre buques y la detección de comportamientos anómalos.

4.2.2 Dataset utilizado

Las pruebas se han realizado utilizando datos correspondientes al mes de enero de 2023. Con el objetivo de simplificar el procesamiento y centrar el análisis en la información relevante para las consultas planteadas, los campos del *dataset* se han acotado a un conjunto reducido de variables, separadas en datos estáticos y datos dinámicos.

Los datos estáticos describen características estructurales del buque y están formados por los siguientes campos:

- *timestamp*: instante temporal en el que se recibe el mensaje AIS.
- *MMSI*: identificador único del buque en el sistema AIS.
- *IMO*: número de identificación asignado por la Organización Marítima Internacional.
- *to_bow* y *to_stern*: distancias desde el transpondedor AIS hasta la proa y la popa respectivamente. La suma de ambos valores permite estimar la eslora del buque.
- *to_port* y *to_starboard*: distancias desde el transpondedor hasta babor y estribor. La suma de estos valores permite estimar la manga del buque.
- *draught*: calado del buque, es decir, la profundidad máxima de la parte sumergida del casco.
- *shiptype*: tipo de buque según la clasificación AIS (pesquero, carguero, de pasajeros, remolcador, etc.).

Por otro lado, los datos dinámicos describen el estado de navegación del buque en cada instante y contienen los siguientes campos:

- *timestamp*: instante temporal asociado a la posición registrada.
- *MMSI*: identificador del buque que transmite el mensaje.
- *lat* y *lon*: coordenadas geográficas del buque (latitud y longitud).

- *speed*: velocidad sobre el fondo, expresada en nudos.
- *course*: rumbo sobre el fondo, medido en grados.
- *heading*: dirección hacia la que apunta la proa del buque, también expresada en grados.

4.2.3 Métricas de evaluación

Para cada tipo de consulta del apartado 4.2.1 se analizan aspectos como el tiempo de respuesta, la corrección de los resultados obtenidos y la capacidad del sistema para interpretar correctamente la intención del usuario.

Este enfoque permite estudiar de forma estructurada cómo se comporta la plataforma a medida que aumenta la complejidad de las consultas, proporcionando una visión clara de sus capacidades y limitaciones en distintos escenarios de análisis de tráfico marítimo.

Para evaluar el sistema se realizaron un total de 100 peticiones, registrando en cada caso dos métricas principales: la puntuación obtenida y el tiempo de respuesta del sistema.

La puntuación se utiliza como indicador de corrección del resultado obtenido, valorando si la respuesta generada por el sistema coincide con el resultado esperado, y si la presentación de dicha respuesta es coherente. En la Tabla 4-1 se muestra la rúbrica seguida para la obtención de la puntuación.

Puntuación	Exactitud de los resultados (70%)	Presentación de los datos (30%)
9,6-10	Los valores obtenidos coinciden completamente con el resultado esperado.	La información se presenta de forma clara, estructurada y sin ambigüedades.
7,6-9,5	Los datos son correctos con pequeñas imprecisiones numéricas o diferencias menores que no afectan al resultado global.	La presentación es clara, aunque puede faltar algún detalle menor.
5,6-7,5	El resultado es parcialmente correcto, con algunos errores o aproximaciones en los valores.	La información se presenta de forma comprensible, aunque con cierta falta de claridad o estructura.
3,6-5,5	El sistema identifica parcialmente la consulta, pero los datos contienen errores relevantes.	La presentación es poco clara o incompleta.
0-3,5	El resultado es incorrecto o no corresponde con la consulta realizada.	La respuesta es confusa o no presenta adecuadamente la información solicitada.

Tabla 4-1 Rúbrica de evaluación de las consultas

4.3 Consultas de atributos

4.3.1 Descripción de las consultas

El objetivo de estas consultas es obtener el valor de un campo concreto a partir de una condición o identificador específico. Este tipo de consultas representa el nivel más básico de interacción con el sistema, ya que normalmente implican operaciones de filtrado simples sobre los datos y no requieren procesos analíticos complejos ni la combinación de múltiples fuentes de información.

En el contexto de esta plataforma, las consultas de atributos suelen consistir en la recuperación directa de información asociada a un buque concreto o a un campo específico del *dataset*. Ejemplos típicos de este tipo de consultas incluyen la obtención de la velocidad media de un buque identificado por su MMSI, la consulta del tipo de embarcación o la recuperación de dimensiones como la eslora o el calado. Algunos ejemplos de estas consultas pueden observarse en la Figura 4-2.

Consulta de datos AIS

Consulta

cual es la eslora del barco con mmsi 356413000

Enviar Limpiar

Consulta: cual es la eslora del barco con mmsi 356413000

Respuesta: Eslora del barco con MMSI 356413000

Campo	Valor
Eslora (m)	200

La eslora del barco con MMSI 356413000 es de 200 metros.

Consulta de datos AIS

Consulta

cual es la mmsi del barco con mas velocidad del 7 de enero de 2023

Enviar Limpiar

Consulta: cual es la mmsi del barco con mas velocidad del 7 de enero de 2023

Respuesta: Tabla de resultados

MMSI	Velocidad máxima
368248520	40.0 nudos
503666200	40.0 nudos
338367159	40.0 nudos
265513020	40.0 nudos
770576272	40.0 nudos

La MMSI del barco con mayor velocidad el 7 de enero de 2023 es 368248520.

Figura 4-2 Ejemplos de consultas de atributos

4.3.2 Resultados

En todas las ejecuciones realizadas se obtuvieron puntuaciones entre los valores 9 y 10, con una media de 9,7, lo que indica que el sistema fue capaz de interpretar correctamente la consulta y recuperar el atributo solicitado en todos los casos. En cuanto al tiempo de respuesta, los valores registrados se sitúan en un rango comprendido aproximadamente entre 9,5 y 19 segundos, con una media de 14,5 segundos, como se puede comprobar en la Figura 4-3.

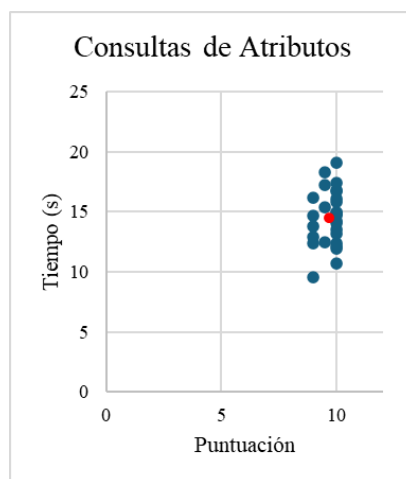


Figura 4-3 Gráfica de resultados de las consultas de atributos

4.3.3 Análisis

Los resultados obtenidos muestran un comportamiento muy estable y consistente del sistema en este tipo de consultas. La puntuación máxima obtenida en las ejecuciones confirma que la arquitectura es capaz de interpretar correctamente consultas simples en lenguaje natural y traducirlas en operaciones estructuradas sobre los datos. Desde el punto de vista del rendimiento, el tiempo medio de respuesta se mantiene en valores relativamente homogéneos entre las distintas ejecuciones.

En conjunto, estos resultados indican que el sistema presenta un alto grado de fiabilidad en consultas de baja complejidad, resolviendo correctamente todas las preguntas planteadas y manteniendo tiempos de respuesta consistentes.

4.4 Consultas de listados

4.4.1 Descripción de las consultas

El objetivo de este tipo de consultas es obtener conjuntos de elementos que cumplen determinadas condiciones dentro del *dataset*. A diferencia de las consultas de atributos, en las que se recupera un valor concreto asociado a un identificador específico, las consultas de listados requieren analizar un volumen mayor de datos para identificar todos los elementos que satisfacen los criterios definidos en la consulta.

En el contexto de esta plataforma, este tipo de preguntas suele implicar operaciones de filtrado, ordenación o agregación sobre el conjunto de datos, permitiendo identificar buques que cumplen determinadas características o comportamientos dentro del tráfico marítimo analizado. Algunos ejemplos habituales de este tipo de consultas incluyen la identificación de los buques más rápidos registrados en una determinada zona, la obtención de los barcos que superan una velocidad concreta o la generación de listados de embarcaciones que cumplen determinadas condiciones operativas. Algunos ejemplos de estas consultas pueden observarse en la Figura 4-4.

Consulta de datos AIS

Consulta

¿Cuáles son los 10 barcos con mayor eslora registrados en el Golfo de Vizcaya?

Enviar

Limpiar

Consulta: ¿Cuáles son los 10 barcos con mayor eslora registrados en el Golfo de Vizcaya?

Respuesta: Tabla de los 10 barcos con mayor eslora registrados en el Golfo de Vizcaya

MMSI	Eslora (m)	IMO	Tipo de barco
345020036	450	0	Remolcador
351382000	438	9159414	Carga
525777777	438	0	No disponible
367532850	428	8836663	SPARE
366945650	427	0	SPARE
366984320	426	0	SPARE
311001024	424	9831684	Otro tipo
367533050	422	8836649	Remolcador de remolque
352412000	421	0	Remolcador
366997280	418	0	SPARE

Respuesta: Los diez barcos con mayor eslora registrados en el Golfo de Vizcaya son los siguientes.

Consulta de datos AIS

Consulta

¿Cuáles son los 10 barcos más rápidos registrados en el Estrecho de Gibraltar en los últimos 3 días?

Enviar

Limpiar

Consulta: ¿Cuáles son los 10 barcos más rápidos registrados en el Estrecho de Gibraltar en los últimos 3 días?

Respuesta: Tabla de los 10 barcos más rápidos registrados en el Estrecho de Gibraltar en los últimos 3 días

MMSI	Velocidad (nudos)
242947000	40.0
224239000	39.5
224593000	39.5
224441790	39.5
242904000	39.0
242787300	39.0
224121250	39.0
242901000	38.5
242903000	38.5
224448000	38.0

Los barcos más rápidos registrados en el Estrecho de Gibraltar en los últimos 3 días son:

- El buque con MMSI 242947000, que alcanzó una velocidad máxima de 40 nudos.
- Los buques con MMSI 224239000, 224593000 y 224441790, que alcanzaron velocidades máximas de 39.5 nudos cada uno.

Figura 4-4 Ejemplos de consultas de listados

Desde el punto de vista del procesamiento, estas consultas presentan una complejidad mayor que las consultas de atributos, ya que requieren recorrer un mayor número de registros del *dataset* y aplicar operaciones de filtrado o clasificación antes de generar el resultado final.

4.4.2 Resultados

En cuanto a la puntuación, los resultados obtenidos muestran valores comprendidos entre 6 y 10, con una media de 8,5, lo que indica que, en la mayoría de los casos, el sistema fue capaz de interpretar correctamente la consulta y generar resultados adecuados.

Respecto al tiempo de respuesta, los valores registrados se sitúan aproximadamente entre 13,5 y 46 segundos, con una media de 31 segundos. Este incremento respecto a las consultas de atributos refleja el mayor volumen de datos que el sistema debe analizar para generar los listados solicitados.

Los resultados completos obtenidos durante las ejecuciones pueden observarse en la Figura 4-5.



Figura 4-5 Gráfica de resultados de las consultas de listados

4.4.3 Análisis

Los resultados obtenidos muestran que las consultas de listados presentan un comportamiento generalmente estable, aunque con una variabilidad mayor tanto en las puntuaciones como en los tiempos de respuesta en comparación con las consultas de atributos. La media de puntuación obtenida indica que el sistema es capaz de resolver correctamente la mayoría de las preguntas planteadas, aunque la mayor complejidad de estas consultas introduce un margen mayor de error en algunos casos.

Desde el punto de vista del rendimiento, se observa que los mayores tiempos de respuesta corresponden a las consultas en las que se imponen múltiples condiciones de filtrado. En estos casos, el sistema debe aplicar diferentes restricciones sobre el *dataset* antes de generar el listado solicitado, lo que incrementa el coste computacional del proceso.

Un ejemplo representativo de este tipo de consulta sería: *¿Cuáles son los 10 barcos más rápidos registrados en el Estrecho de Gibraltar en los últimos 3 días?* En este caso, el sistema debe aplicar simultáneamente un filtro por velocidad, un filtro espacial basado en la posición geográfica y un filtro temporal, lo que implica analizar un mayor volumen de registros y realizar más operaciones sobre los datos.

En conjunto, estos resultados indican que el sistema mantiene un buen nivel de precisión en consultas de complejidad intermedia, aunque el incremento en el número de condiciones de filtrado se refleja directamente en los tiempos de ejecución. Este comportamiento es coherente con el aumento de operaciones necesarias para procesar este tipo de consultas dentro del *dataset* analizado.

4.5 Consultas de fusión de datos

4.5.1 Descripción de las consultas

El objetivo de este tipo de consultas es obtener información que requiere combinar datos procedentes de distintas tablas del sistema. A diferencia de las consultas de atributos o de listados, en este caso la información solicitada no puede recuperarse directamente de una única fuente, sino que requiere integrar datos de naturaleza diferente.

En el contexto de esta plataforma, las consultas de fusión de datos implican generalmente la combinación de datos dinámicos de navegación con datos estáticos de los buques. Este proceso permite relacionar características estructurales del buque, como su tipo o dimensiones, con información operativa como la velocidad o la posición registrada en cada instante.

Ejemplos típicos de este tipo de consultas incluyen el cálculo de la velocidad media de un determinado tipo de buque, la identificación de los barcos más rápidos junto con sus características estructurales o el análisis de dimensiones asociadas a determinados patrones de navegación. Algunos ejemplos de estas consultas pueden observarse en la Figura 4-6.

Consulta de datos AIS

Consulta

cual es la velocidad media y la eslora del barco con mmsi 356413000

Enviar Limpiar

Consulta: cual es la velocidad media y la eslora del barco con mmsi 356413000

Respuesta: Resultados

MMSI	Longitud (m)	Ancho (m)
356413000	200	32

La velocidad media del barco con MMSI 356413000 es de aproximadamente 1,5 nudos, con un rango de 0,5 a 12,3 nudos.

Consulta de datos AIS

Consulta

cual es la velocidad media de los remolcadores

Enviar Limpiar

Consulta: cual es la velocidad media de los remolcadores

Respuesta: Tabla de resultados

Parámetro	Valor
Media	5,26 nudos
Desviación estándar	3,19 nudos
Número de observaciones	561088

La velocidad media de los remolcadores es de aproximadamente 5,26 nudos.

Figura 4-6 Ejemplos de consultas de fusión de datos

Desde el punto de vista del procesamiento, este tipo de consultas introduce una complejidad adicional, ya que requiere realizar operaciones de unión entre tablas antes de aplicar los filtros o agregaciones necesarios para obtener el resultado final. Por este motivo, estas consultas constituyen un escenario adecuado para evaluar la capacidad del sistema para gestionar operaciones de integración de datos dentro del proceso de interpretación de consultas en lenguaje natural.

4.5.2 Resultados

En cuanto a la puntuación, los resultados obtenidos muestran una mayor variabilidad que en las consultas de atributos. Las puntuaciones registradas se sitúan entre 3,5 y 10, con una media de 7, lo que indica que, aunque el sistema fue capaz de resolver correctamente algunas consultas, en otros casos la interpretación o la ejecución de la consulta no produjo el resultado esperado.

Respecto al tiempo de respuesta, los valores registrados se sitúan aproximadamente entre 15 y 37 segundos, con una media de 26 segundos. Este incremento respecto a las consultas más simples refleja el mayor coste computacional asociado a las operaciones de combinación de datos necesarias para resolver este tipo de preguntas.

Los resultados completos obtenidos durante las ejecuciones pueden observarse en la Figura 4-7.

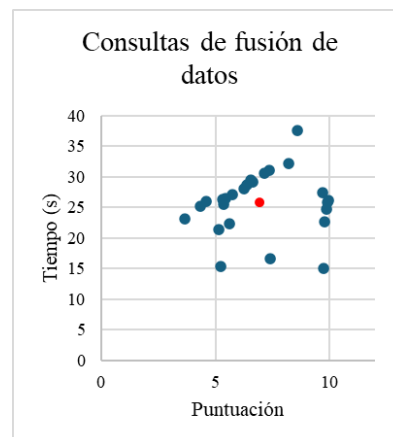


Figura 4-7 Gráfica de resultados de las consultas de fusión de datos

4.5.3 *Análisis*

Los resultados obtenidos muestran que las consultas de fusión de datos representan un escenario significativamente más exigente para el sistema que los tipos de consultas analizados previamente. La necesidad de integrar información procedente de diferentes tablas introduce una complejidad adicional tanto en la interpretación de la consulta como en la ejecución de las operaciones necesarias sobre los datos.

Desde el punto de vista de la precisión, la media de puntuación obtenida indica que el sistema es capaz de resolver correctamente una parte relevante de las consultas planteadas, aunque presenta dificultades en algunos casos. Estas dificultades están relacionadas con la correcta identificación de las relaciones entre los distintos conjuntos de datos y con la construcción adecuada de las operaciones necesarias para combinar la información.

En términos de rendimiento, el aumento del tiempo medio de respuesta respecto a las consultas más simples es coherente con el incremento de la complejidad del procesamiento requerido. La necesidad de realizar operaciones de unión entre tablas y posteriores agregaciones implica un mayor volumen de operaciones sobre el *dataset*, lo que se refleja en los tiempos registrados.

En conjunto, estos resultados ponen de manifiesto que las consultas de fusión de datos constituyen un escenario intermedio entre las consultas simples de recuperación de atributos y las consultas analíticas más complejas, permitiendo evaluar la capacidad del sistema para manejar operaciones de integración de información dentro de un entorno de consulta en lenguaje natural.

4.6 Consultas analíticas

4.6.1 *Descripción de las consultas*

Las consultas analíticas representan el nivel más alto de complejidad dentro de las pruebas realizadas. A diferencia de las consultas de atributos, listados o fusión de datos, cuyo objetivo principal es recuperar o combinar información existente, en este caso se busca extraer información interpretativa a partir de los datos disponibles, analizando comportamientos o relaciones entre distintos elementos del *dataset*.

En el contexto de esta plataforma, las consultas analíticas se centran en tres tipos de análisis: patrones de vida de los buques, interacción entre embarcaciones y detección de anomalías en el tráfico marítimo. Este tipo de preguntas requiere que el sistema no solo interprete correctamente la consulta en lenguaje natural, sino que también sea capaz de seleccionar las operaciones analíticas adecuadas para obtener la información solicitada.

Las consultas relacionadas con patrones de vida buscan identificar el comportamiento habitual de un buque, analizando variables como su velocidad media, rumbo predominante o zonas de operación más frecuentes. Por su parte, las consultas de interacción entre barcos se centran en el análisis de relaciones espaciales o temporales entre distintas embarcaciones, como el cálculo de distancias mínimas entre buques o la identificación de encuentros cercanos.

Finalmente, las consultas orientadas a detección de anomalías intentan identificar comportamientos poco habituales dentro del tráfico marítimo, como cambios bruscos de rumbo, velocidades inusuales o patrones de movimiento que se desvían de lo esperado. Algunos ejemplos de este tipo de consultas pueden observarse en la Figura 4-8.

Debido a la naturaleza interpretativa de estas preguntas, su resolución requiere combinar diferentes operaciones analíticas sobre los datos, lo que introduce una mayor complejidad tanto en la interpretación de la consulta como en la ejecución de las operaciones necesarias para generar la respuesta.

Consulta de datos AIS

Consulta

cual ha sido el CPA entre los barcos con mmsi 261148000 y mmsi 261182515

Enviar

Limpiar

Consulta: cual ha sido el CPA entre los barcos con mmsi 261148000 y mmsi 261182515

Respuesta: Análisis de CPA entre buques con MMSI 261148000 y MMSI 261182515

Resultados del análisis

Tabla de posiciones de los dos buques

Timestamp	MMSI	Latitud	Longitud	Velocidad (nudos)	Rumbo (grados)
1673823289	261148000	54.519222	18.558555	0,1	285,0
1673822929	261148000	54.519212	18.558535	0,1	95,8
...	...	...	...	...	...

Tabla de CPA entre los dos buques

Timestamp	MMSI 261148000	MMSI 261182515	CPA (millas náuticas)
1673823289	54.519222, 18.558555	54.520123, 18.559667	0,5
1673822929	54.519212, 18.558535	54.521012, 18.560767	0,6
...	...	...	...

Interpretación final

El análisis de CPA entre los buques con MMSI 261148000 y MMSI 261182515 muestra que la distancia más cercana entre ellos fue de aproximadamente 0,5 millas náuticas en el momento del timestamp 1673823289. Es importante tener en cuenta que esta distancia puede variar dependiendo de las posiciones y velocidades de los buques en cada momento.

Nota: La tabla de CPA se ha truncado para mantener la brevedad. Si es necesario, puedo proporcionar más información o detalles adicionales.

Consulta de datos AIS

Consulta

patron de vida en el Estrecho de Malaca

Enviar

Limpiar

Consulta: patron de vida en el Estrecho de Malaca

Respuesta: Análisis del patrón de vida en el Estrecho de Malaca

Resumen El análisis realizado muestra un patrón de vida complejo en el Estrecho de Malaca, con una gran variedad de buques y velocidades. A continuación, se presentan los resultados detallados.

Top 3 tipos de buques

Tipo de buque	Número de mensajes	Vehículos únicos	Velocidad promedio	Rumbo promedio
Desconocido	1011	468	10,78 nudos	281,53°
Carga	564	192	12,38 nudos	299,64°
Tanque	338	116	11,91 nudos	356,27°

Velocidad y rumbo promedio

- Velocidad promedio: 11,70 nudos
- Rumbo promedio: 289,60°

Análisis de anomalías

Se compararon los resultados con los promedios de su tipo. Los buques desconocidos tienen una velocidad promedio ligeramente más baja que la media, mientras que los buques de carga y tanque tienen velocidades promedio similares a las medias.

Interpretación final

El patrón de vida en el Estrecho de Malaca es complejo y variado. Los buques desconocidos son los más numerosos, pero también tienen una velocidad promedio ligeramente más baja que la media. Los buques de carga y tanque tienen velocidades promedio similares a las medias. Es importante tener en cuenta estas diferencias al analizar el tráfico marítimo en esta zona.

Figura 4-8 Ejemplos de consultas de análisis

4.6.2 Resultados

En cuanto a la puntuación, los resultados obtenidos se sitúan entre 3 y 10, con una media de aproximadamente 7,5, lo que indica que el sistema fue capaz de resolver correctamente una parte significativa de las consultas analíticas planteadas, aunque con una variabilidad mayor que en los tipos de consultas analizados anteriormente.

Respecto al tiempo de respuesta, los valores registrados se encuentran aproximadamente entre 20 y 100 segundos, con una media cercana a los 50 segundos. Este incremento respecto a los apartados anteriores refleja la mayor complejidad computacional asociada a este tipo de análisis, que requiere realizar múltiples operaciones sobre los datos antes de generar el resultado final.

Los resultados completos obtenidos durante las ejecuciones pueden observarse en la Figura 4-9.

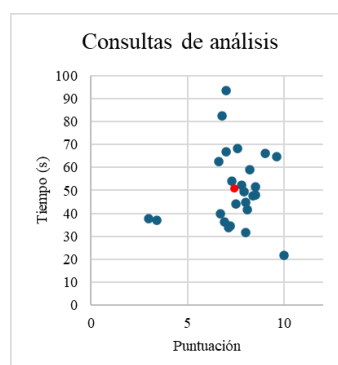


Figura 4-9 Gráfica de resultados de las consultas de análisis

4.6.3 Análisis

Los resultados obtenidos muestran que las consultas analíticas constituyen el escenario más exigente para el sistema, tanto desde el punto de vista de la interpretación de la consulta como de la ejecución de las operaciones necesarias para generar la respuesta.

En primer lugar, se observa que las consultas relacionadas con patrones de vida de los buques presentan un comportamiento especialmente satisfactorio. Este resultado es coherente con el diseño del sistema, ya que este tipo de análisis es el que se encuentra más desarrollado dentro del documento *analysis.md*, donde se han definido con mayor detalle las operaciones necesarias para caracterizar el comportamiento habitual de una embarcación. Como consecuencia, el sistema es capaz de identificar con mayor precisión las variables relevantes y generar resultados consistentes en este tipo de preguntas.

Las consultas relacionadas con la interacción entre barcos también presentan, en general, resultados bastante acertados. No obstante, se observa que la calidad de las respuestas tiende a disminuir cuando las preguntas planteadas dejan un mayor grado de libertad en la interpretación del análisis. Cuando los parámetros de la consulta están claramente acotados, el sistema genera resultados más precisos, mientras que las consultas más abiertas introducen ambigüedad en el proceso de interpretación.

Los peores resultados se observan en las consultas relacionadas con la detección de anomalías. Este comportamiento resulta coherente con el diseño actual del sistema, ya que este apartado es el que se encuentra menos desarrollado dentro del documento *analysis.md*, lo que limita la capacidad del sistema para determinar con claridad qué tipo de análisis debe aplicar en cada caso.

En algunos casos también se observan valores que se desvían del comportamiento general del conjunto de consultas. En particular, aparecen ejecuciones con tiempos de respuesta significativamente superiores al resto. Este fenómeno está relacionado con el grado de ambigüedad de la pregunta formulada. A diferencia de otros tipos de consultas, en las consultas analíticas las preguntas más abiertas tienden a incrementar el tiempo de ejecución.

Por ejemplo, ante una pregunta como: *¿Hay anomalías entre los datos?*, el sistema no dispone de ningún tipo de restricción que limite el ámbito del análisis. Como consecuencia, se ve obligado a aplicar diferentes operaciones analíticas sobre el conjunto completo de datos disponibles, lo que incrementa considerablemente el coste computacional. En cambio, cuando la consulta incluye restricciones adicionales, por ejemplo, *¿Algún pesquero del Estrecho de Gibraltar del último día presenta alguna anomalía?*, el sistema puede limitar el análisis a un subconjunto mucho más reducido de registros, reduciendo así el tiempo de respuesta.

Por último, algunas consultas presentan puntuaciones inferiores debido principalmente a la forma en que están formuladas. Preguntas como: *¿Ha ocurrido algo raro en el Estrecho de Malaca el último día?* no definen con precisión el tipo de análisis esperado. En estos casos el sistema debe inferir posibles interpretaciones, lo que puede conducir a respuestas menos ajustadas a la pregunta planteada o con información adicional no estrictamente necesaria.

En conjunto, estos resultados muestran que el sistema es capaz de abordar consultas analíticas con un nivel razonable de precisión cuando las operaciones de análisis están bien definidas y la consulta incluye restricciones claras. Sin embargo, también ponen de manifiesto la importancia de seguir ampliando y refinando las reglas de análisis del sistema para mejorar su comportamiento en consultas más abiertas o ambiguas.

4.7 Comparación entre tipos de consulta

En este apartado se comparan los resultados obtenidos en las distintas categorías de consulta analizadas a lo largo del capítulo. Se evalúan dos aspectos fundamentales del comportamiento del sistema: el tiempo de respuesta necesario para resolver cada consulta y la calidad de la respuesta generada, medida a través de la puntuación media obtenida en cada categoría. Esta comparación permite

observar cómo varía el rendimiento del sistema a medida que aumenta la complejidad de las consultas planteadas.

Al comparar los resultados obtenidos en las distintas categorías de consulta, se observa que los tiempos de respuesta, mostrados en la Figura 4-10, están principalmente determinados por la cantidad de datos que el sistema debe analizar y por el volumen de información que debe procesar para generar la respuesta final.

En este sentido, las consultas de atributos presentan los tiempos más reducidos, con una media de 14,50 segundos, ya que normalmente implican operaciones simples de filtrado o agregación sobre un conjunto limitado de datos. Las consultas de listados muestran un incremento significativo en el tiempo medio de respuesta, alcanzando 31 segundos de media, debido a que requieren recorrer un volumen mayor de registros y generar resultados que incluyen múltiples elementos. Este tipo de consultas, pese a ser conceptualmente más sencillas que las de fusión de datos, presentan un mayor tiempo de ejecución (26 segundos en este último caso). Esto se debe a que las consultas de listados suelen implicar la generación de conjuntos más amplios de resultados, lo que incrementa el coste de procesamiento y presentación de la información.

Finalmente, las consultas analíticas son las que presentan el mayor tiempo medio de respuesta, con 50,79 segundos, lo cual resulta coherente con su mayor complejidad, ya que requieren realizar múltiples operaciones de análisis antes de generar la respuesta.

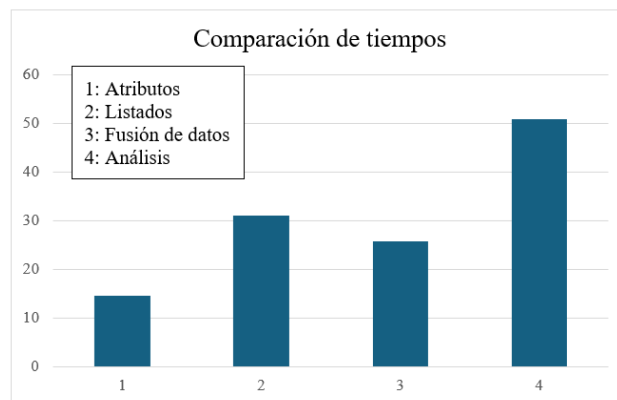


Figura 4-10 Gráfica de comparación de tiempos

Además, realizando un desglose del tiempo invertido en las distintas fases del flujo de procesamiento de las consultas (mostrado en Figura 4-11), se confirma lo siguiente:

En las consultas deterministas, la mayor parte del tiempo se concentra en las fases asociadas al modelo de lenguaje, principalmente en la generación del plan de ejecución y en la redacción final de la respuesta, mientras que la fase de ejecución de la herramienta representa una fracción muy reducida del tiempo total.

En el caso de las consultas analíticas se observa un comportamiento similar, aunque con una distribución más equilibrada entre etapas. En estas consultas, además de la interpretación de la pregunta y la generación de la respuesta, aparece un coste adicional asociado a la recuperación de contexto

mediante el sistema RAG durante la fase de análisis. En conjunto, estos resultados muestran que el coste computacional del sistema está dominado por los procesos de interpretación y generación realizados por el modelo de lenguaje, mientras que las operaciones deterministas sobre los datos representan una parte relativamente pequeña del tiempo total de procesamiento.

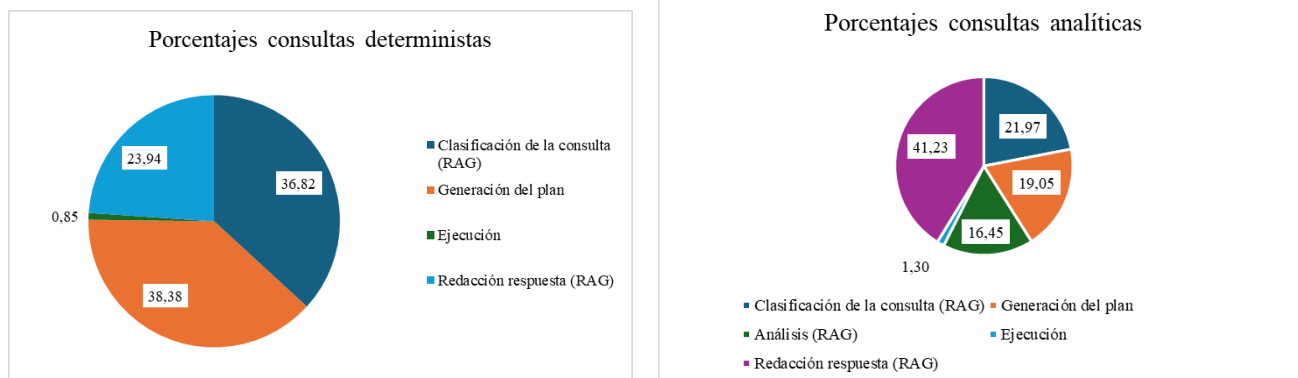


Figura 4-11 Gráficas de desglose de los tiempos empleados

Con estas gráficas se puede concluir que el coste computacional asociado a las consultas está dominado por el LLM y el RAG, lo que refuerza el uso de una plataforma que use únicamente estos elementos para la contextualización y no para la recuperación de los datos.

En cuanto a la calidad de las respuestas, véase la Figura 4-12, medida a través de la puntuación media obtenida en cada categoría, se observa que las consultas de atributos presentan el mejor rendimiento con una puntuación media de 9,7, lo que confirma la elevada fiabilidad del sistema en preguntas de baja complejidad. Las consultas de listados obtienen también resultados satisfactorios, con una puntuación media de 8,53.

Un resultado interesante se observa al comparar las consultas de fusión de datos (7) y las consultas analíticas (7,5), donde estas últimas presentan una puntuación ligeramente superior a pesar de su mayor complejidad. Este comportamiento puede explicarse por el uso del documento *analysis.md*, que proporciona al sistema una estructura más definida para planificar este tipo de análisis. Gracias a este soporte documental, el sistema es capaz de elaborar un plan de ejecución más sólido que cuando debe interpretar la consulta únicamente a partir del lenguaje natural.

Por el contrario, las consultas de fusión de datos dependen principalmente de la correcta identificación de relaciones entre tablas, lo que introduce dificultades adicionales en el proceso de interpretación, especialmente en el momento de realizar los saltos entre distintos conjuntos de datos.

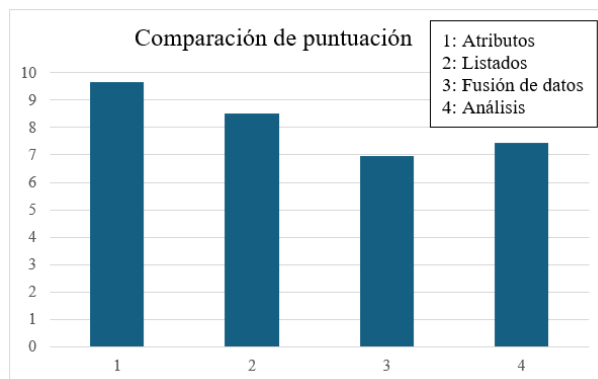


Figura 4-12 Gráfica de comparación de puntuación de calidad de las respuestas

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

El objetivo principal de este Trabajo de Fin de Grado ha sido el diseño e implementación de una plataforma capaz de realizar consultas y análisis sobre datos de tráfico marítimo AIS mediante el uso de modelos de lenguaje y técnicas de recuperación de información. En particular, se ha buscado desarrollar un sistema que permita a los usuarios interactuar con los datos utilizando lenguaje natural, simplificando el acceso a información compleja sin necesidad de conocimientos técnicos sobre bases de datos o herramientas de análisis.

Para alcanzar este objetivo, se ha diseñado una arquitectura basada en el uso de un modelo de lenguaje integrado con herramientas específicas de análisis de datos AIS. Esta arquitectura permite transformar las consultas formuladas por el usuario en lenguaje natural en una secuencia de operaciones que posteriormente son ejecutadas sobre los datos disponibles. De esta forma, el sistema actúa como una interfaz inteligente entre el usuario y la base de datos, facilitando la obtención de información relevante sobre el tráfico marítimo.

Durante el desarrollo del trabajo se ha implementado un prototipo funcional capaz de realizar distintos tipos de consultas, que incluyen desde la obtención de atributos específicos de buques hasta análisis más complejos que requieren combinar diferentes conjuntos de datos o realizar cálculos sobre las trayectorias de los buques. Los resultados obtenidos durante la fase de evaluación muestran que el sistema es capaz de interpretar correctamente la mayoría de las consultas planteadas y generar respuestas coherentes a partir de los datos AIS disponibles.

Asimismo, durante el desarrollo del sistema se ha podido comprobar la necesidad de combinar las capacidades de interpretación del lenguaje natural de los modelos de lenguaje con herramientas externas especializadas en el procesamiento de datos.

En este sentido, el trabajo realizado se ha apoyado en investigaciones recientes sobre el uso de LLM para el análisis de datos AIS [28], donde se concluye que, aunque estos modelos presentan un gran potencial para interpretar consultas en lenguaje natural, muestran limitaciones cuando deben operar directamente sobre grandes volúmenes de datos o realizar operaciones analíticas complejas.

Por este motivo, una arquitectura mixta que combine el razonamiento del modelo de lenguaje con herramientas de procesamiento de datos resulta una solución especialmente adecuada, permitiendo aprovechar las fortalezas de ambos enfoques y mejorar la fiabilidad y eficiencia del sistema.

En conjunto, el trabajo realizado demuestra la viabilidad de integrar modelos de lenguaje con sistemas de análisis de datos marítimos para facilitar el acceso y explotación de información procedente

del sistema AIS. La plataforma desarrollada constituye un primer paso hacia herramientas más avanzadas de apoyo al análisis del tráfico marítimo y a la toma de decisiones en entornos donde la información disponible es abundante y compleja.

5.2 Líneas futuras

A partir de los resultados obtenidos y de las limitaciones identificadas durante el desarrollo del sistema, se proponen varias líneas de trabajo que podrían explorarse en futuras investigaciones.

5.2.1 *Optimización del rendimiento y escalabilidad del sistema*

Una posible línea futura consiste en mejorar la eficiencia del sistema cuando se trabaja con conjuntos de datos de mayor tamaño. El análisis de datos AIS puede implicar volúmenes muy elevados de información, por lo que sería interesante estudiar técnicas de optimización que permitan mejorar los tiempos de respuesta y la escalabilidad de la plataforma.

Asimismo, una posible línea de evolución del sistema sería la incorporación de arquitecturas más avanzadas, como enfoques de RAG agéntico o el uso de modelos con mayor capacidad de razonamiento e iteración. Estas aproximaciones permitirían mejorar el proceso de interpretación de las consultas y la explotación de la información recuperada. Esta línea de trabajo se encuadra dentro de la optimización del sistema, ya que, como se ha señalado a lo largo de la memoria, los modelos empleados pueden presentar limitaciones a la hora de manejar la complejidad y el volumen de los datos analizados.

5.2.2 *Ampliación de las capacidades de análisis*

Otra posible mejora sería ampliar las funcionalidades analíticas del sistema, incorporando nuevos tipos de consultas relacionadas con el análisis de patrones de tráfico, detección de anomalías o identificación de comportamientos sospechosos en la navegación de los buques. Esto permitiría aumentar el valor operativo de la plataforma en aplicaciones relacionadas con la vigilancia marítima.

5.2.3 *Mejora del uso de modelos de lenguaje especializados*

Otra línea de trabajo interesante sería explorar el uso de modelos de lenguaje más especializados o adaptados al dominio marítimo mediante técnicas como el *fine-tuning* o el uso de bases de conocimiento específicas. Esto podría mejorar la capacidad del sistema para interpretar consultas complejas y generar respuestas más precisas en el contexto del análisis de datos AIS.

5.2.4 *Integración con otras fuentes de información marítima*

Finalmente, el sistema podría enriquecerse mediante la integración de otras fuentes de datos adicionales, como información meteorológica, registros portuarios, imágenes satelitales o datos procedentes de sistemas radar. La combinación de distintas fuentes permitiría realizar análisis más completos y mejorar la capacidad de interpretación de las situaciones marítimas.

6 BIBLIOGRAFÍA

- [1] V. Martínez Martínez, «Los orígenes y la importancia del Big Data,» Universidad Isabel I, 09 septiembre 2024. [En línea]. Available: <https://www.ui1.es/blog-ui1/los-origenes-y-la-importancia-del-big-data>. [Último acceso: 12 enero 2026].
- [2] T. Mucci y C. Styker, «*What is big data analytics?*,» 5 abril 2024. [En línea]. Available: <https://www.ibm.com/think/topics/big-data-analytics>. [Último acceso: 12 enero 2026].
- [3] G. Pulido Pulido, «La Batalla Multidominio y el campo de batalla futuro,» 08 diciembre 2018. [En línea]. Available: <https://www.revistaejercitos.com/articulos/batalla-multidominio/>. [Último acceso: 14 enero 2026].
- [4] M. Rodelgo Lacruz, B. Barragáns Martínez, N. Fernández García, G. Higgins, P. Sendín Raña y A. Suárez García, «*Real-time AIS data analysis for anomaly detection in the maritime environment*,» *17th NATO Operations Research and Analysis (OR&A) Conference*, Laurel, Maryland (USA), 2023.
- [5] E. Flores, M. Ibarra, L. Moreno, G. Sidorov y P. Rosso, «Modelos de recuperación de información basados en n-gramas aplicados a la reutilización de código fuente,» Universitat Politècnica de València, España; Centro de Investigación en Computación, Instituto Politécnico Nacional, México, 2014.
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser y I. Polosukhin, «*Attention is all you need*,» Google, 2017.
- [7] S. Altman, «ChatGPT - Prompt usado: "Crea una imagen a modo de esquema de la evolución de los modelos de lenguaje hasta los actuales LLMs.",» [En línea]. Available: <https://chatgpt.com/>.
- [8] S. Altman, «OpenAI,» [En línea]. Available: <https://openai.com/about/>.
- [9] E. Rodríguez de Luis, «ChatGPT-3.5 vs GPT-4: las principales diferencias entre los modelos de OpenAI, explicadas,» 01 febrero 2024. [En línea]. Available: <https://www.genbeta.com/comparativa/chatgpt-3-5-vs-gpt-4-principales-diferencias-explicadas>. [Último acceso: 21 enero 2026].

- [10] Google, «*PaLM 2 Technical Report*,» 2023.
- [11] G. Gemini Team, «*Gemini: A Family of Highly Capable*,» Google, 2025.
- [12] «Introducción a Microsoft 365 Copilot,» Microsoft, 06 marzo 2026. [En línea]. Available: <https://learn.microsoft.com/es-es/copilot/microsoft-365/microsoft-365-copilot-overview>. [Último acceso: 17 marzo 2026].
- [13] P. P. Dakle, S. Rallabandi y P. Raghavan, «*Understanding BLOOM: An empirical study on diverse NLP tasks*,» AI Center of Excellence, Fidelity Investments, Boston, Massachusetts (USA), 2023.
- [14] Y.-F. Huang, C.-W. Kuo y H.-C. Tsai, «*Comparative Analysis of Automatic Literature Review Using Mistral Large Language Model and Human Reviewers*,» Research Square, 2024.
- [15] T. F. L. Team, «*The Falcon Series of Open Language Models*,» Technology Innovation Institute, Abu Dhabi, 2023.
- [16] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave y G. Lample, «*LLaMA: Open and Efficient Foundation Language Models*,» 2023.
- [17] M. Zuckerberg, «Meta AI,» [En línea]. Available: <https://ai.meta.com/meta-ai/>.
- [18] Y. Hu y Y. Lu, «*RAG and RAU: A Survey on Retrieval-Augmented Language Model in NLP*,» East China University of Science and Technology, Peking University, 2025.
- [19] «¿Qué es RAG (generación aumentada por recuperación)?,» Amazon, [En línea]. Available: <https://aws.amazon.com/es/what-is/retrieval-augmented-generation/>. [Último acceso: 17 marzo 2026].
- [20] «*Why use LangChain?*,» [En línea]. Available: <https://www.langchain.com/langchain>. [Último acceso: 17 enero 2026].
- [21] «*Welcome to LlamaIndex*,» LlamaIndex, [En línea]. Available: <https://developers.llamaindex.ai/python/framework/>. [Último acceso: 20 enero 2026].
- [22] «*International Maritime Organization*,» [En línea]. Available: <https://www.imo.org/>. [Último acceso: 21 enero 2026].
- [23] «*AIS Dynamic data and Static Information*,» IMARINEX, [En línea]. Available: <https://www.imarinex.com/ais-dynamic-data-and-static-information/>. [Último acceso: 21 enero 2026].
- [24] O. Ukolova, «*AIS Density Map to watch intensity of global vessel traffic*,» LinkedIn, 09 febrero 2016. [En línea]. Available: <https://www.linkedin.com/pulse/ais-density-map-watch-intensity-global-vessel-traffic-olga-ukolova>. [Último acceso: 22 enero 2026].
- [25] «Bases de datos estáticas. ¿Qué son? Ejemplos,» ayudaley, [En línea]. Available: <https://ayudaleyprotecciondatos.es/bases-de-datos/estaticas/>. [Último acceso: 22 enero 2026].
- [26] T. Mucci, «*What is data-driven decision-making?*,» IBM, 23 julio 2024. [En línea]. Available: <https://www.ibm.com/think/topics/data-driven-decision-making>. [Último acceso: 22 enero 2026].

- [27] H. Park, J. Jung, M. Seo, H. Choi, D. Cho, S. Park y D.-G. Choi, «*AIS-LLM: A Unified Framework for Maritime Trajectory Prediction, Anomaly Detection, and Collision Risk Assessment with Explainable Forecasting*,» 2025.
- [28] G. Merten, G. Dejaegere y M. Sakr, «*Using LLMs for Analyzing AIS Data*,» Université libre de Bruxelles, Bruselas, 2025.
- [29] P. Herrezuelo de la Fuente, «TFM - Democratización de los Datos con Inteligencia Artificial: Un Asistente Virtual basado en LLM,» Universidad de Oviedo, 2025.
- [30] B. Clark, «¿Qué es LangGraph?,» IBM, 10 enero 2025. [En línea]. Available: <https://www.ibm.com/es-es/think/topics/langgraph>. [Último acceso: 27 enero 2026].
- [31] J. León Martón y Á. J. Álvarez Miguel, «Técnicas de lenguaje natural aplicadas a consultas documentales,» Universidad de Alcalá, 2025.
- [32] R. Ong, «¿Qué es Faiss (Facebook AI Similarity Search)?,» Datacamp, 29 julio 2024. [En línea]. Available: <https://www.datacamp.com/es/blog/faiss-facebook-ai-similarity-search>. [Último acceso: 11 febrero 2026].
- [33] «*Install Docker Engine on Ubuntu*,» DockerDocs, [En línea]. Available: <https://docs.docker.com/engine/install/ubuntu/>. [Último acceso: 28 enero 2026].
- [34] «*Installing the NVIDIA Container Toolkit*,» Nvidia, [En línea]. Available: <https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html#installation>. [Último acceso: 28 enero 2026].
- [35] «*Ollama is now available as an official Docker image*,» Ollama, [En línea]. Available: <https://ollama.com/blog/ollama-is-now-available-as-an-official-docker-image>. [Último acceso: 28 enero 2026].
- [36] «*Open WebUI - Quick start with Docker*,» Open WebUI, [En línea]. Available: <https://docs.openwebui.com/getting-started/quick-start/>. [Último acceso: 29 enero 2026].
- [37] A. Treuille, T. Teixeira y A. Kelly, «*Streamlit - A faster way to build and share data apps*,» [En línea]. Available: <https://streamlit.io/>. [Último acceso: 11 febrero 2026].
- [38] M. Raasveldt y H. Mühleisen, «*DuckDB - An in-process SQL OLAP database management system*,» [En línea]. Available: <https://duckdb.org/>. [Último acceso: 11 febrero 2026].
- [39] L. Roque, «*Master Semantic Search at Scale: Index Millions of Documents with Lightning-Fast Inference Times using FAISS*,» Medium, 31 marzo 2023. [En línea]. Available: <https://medium.com/data-science/master-semantic-search-at-scale-index-millions-of-documents-with-lightning-fast-inference-times-fa395e4efd88>. [Último acceso: 18 marzo 2026].
- [40] «*Introducing JSON*,» [En línea]. Available: <https://www.json.org/json-en.html>. [Último acceso: 22 febrero 2026].
- [41] «*Polars - DataFrames for the new era*,» [En línea]. Available: <https://pola.rs/>. [Último acceso: 22 febrero 2026].
- [42] L. Siddiqui y T. Barrera, «Explicación de Apache Parquet: Guía para profesionales de los datos,» Datacamp, 11 febrero 2025. [En línea]. Available: <https://www.datacamp.com/es/tutorial/apache-parquet>. [Último acceso: 03 febrero 2026].

- [43] «Rust - El lenguaje que empodera a todos para construir software fiable y eficiente.,» [En línea]. Available: <https://rust-lang.org/es/>. [Último acceso: 23 febrero 2026].
- [44] «*Apache Arrow - The universal columnar format and multi-language toolbox for fast data interchange and in-memory analytics*,» [En línea]. Available: <https://arrow.apache.org/>. [Último acceso: 23 febrero 2026].

ANEXO I: IMPLICACIONES SOCIALES, ECONÓMICAS Y AMBIENTALES

Siguiendo los criterios del Sello Internacional de Calidad EUR-ACE, este anexo recoge un análisis de los impactos sociales, económicos y ambientales vinculados a este TFG. El objetivo de este anexo es sintetizar dichas consideraciones y destacar la influencia que este trabajo puede tener en distintos ámbitos.

Desde una perspectiva social, el desarrollo de herramientas capaces de analizar grandes volúmenes de datos de tráfico marítimo puede contribuir a mejorar la seguridad y la vigilancia en el entorno marítimo. La capacidad de detectar patrones de comportamiento anómalos en los movimientos de los buques permite identificar situaciones potencialmente sospechosas que podrían pasar desapercibidas mediante métodos tradicionales de análisis. De este modo, plataformas como la desarrollada en este trabajo pueden apoyar a los organismos responsables de la vigilancia marítima proporcionando herramientas que faciliten el análisis de la información disponible y la detección temprana de situaciones de riesgo.

En particular, la identificación de patrones incongruentes en las trayectorias de los buques puede ayudar a detectar actividades ilícitas en el mar. Un ejemplo de ello es el fenómeno conocido como *loitering*, que consiste en la permanencia prolongada de una embarcación en una zona concreta sin una justificación aparente. Este tipo de comportamientos puede estar relacionado con actividades ilegales como el tráfico ilícito, la pesca ilegal o la explotación no autorizada de restos arqueológicos submarinos. La detección temprana de estos comportamientos puede contribuir a la protección del patrimonio subacuático y a la preservación de los ecosistemas marinos.

Desde el punto de vista económico, el uso de sistemas de análisis automatizado basados en inteligencia artificial puede mejorar la eficiencia en la gestión del tráfico marítimo y en las tareas de supervisión y control. La automatización de determinadas tareas de análisis permite reducir el tiempo necesario para procesar grandes volúmenes de datos AIS, facilitando la toma de decisiones por parte de los operadores. Esto puede traducirse en una mayor eficiencia en las operaciones portuarias, una mejor planificación de rutas marítimas y una optimización del uso de los recursos disponibles en el ámbito logístico y del transporte marítimo.

Por otro lado, desde la perspectiva ambiental, es importante considerar el impacto energético asociado al uso de tecnologías basadas en inteligencia artificial. El funcionamiento de modelos de lenguaje y sistemas de análisis de datos requiere el uso de infraestructuras computacionales con una elevada capacidad de procesamiento, como servidores especializados o unidades de procesamiento gráfico (GPU). Estas infraestructuras consumen cantidades significativas de energía, especialmente cuando se procesan grandes volúmenes de datos o se ejecutan modelos complejos. Este consumo energético puede traducirse en una mayor huella de carbono si la electricidad utilizada procede de fuentes no renovables, por lo que resulta importante tener en cuenta este aspecto en el desarrollo y uso de sistemas de inteligencia artificial.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

El desarrollo de tecnologías basadas en inteligencia artificial aplicadas al análisis del tráfico marítimo plantea diversas cuestiones éticas y sociales que deben abordarse de forma responsable. En particular, el uso de modelos capaces de analizar grandes volúmenes de datos y generar respuestas automatizadas introduce nuevos retos relacionados con la transparencia, la fiabilidad de los resultados y el uso adecuado de estas herramientas en entornos operativos.

Uno de los principales aspectos éticos a considerar es la necesidad de garantizar que los sistemas de inteligencia artificial sean comprensibles y transparentes para los usuarios que interactúan con ellos. Dado que estas tecnologías pueden influir en la interpretación de la actividad marítima o en la toma de decisiones operativas, es importante que los resultados proporcionados por el sistema puedan ser evaluados críticamente por los analistas humanos. En este sentido, la inteligencia artificial debe entenderse como una herramienta de apoyo a la toma de decisiones y no como un sustituto del juicio humano.

Asimismo, el uso de grandes cantidades de datos para entrenar o alimentar estos sistemas plantea cuestiones relacionadas con la protección de la información y el tratamiento responsable de los datos. En el ámbito marítimo, los datos AIS constituyen una fuente fundamental para el análisis del tráfico, pero su utilización debe realizarse respetando los marcos regulatorios existentes y evitando usos indebidos de la información recopilada.

Desde una perspectiva social, el desarrollo de herramientas capaces de analizar grandes volúmenes de información puede aportar importantes beneficios para mejorar la seguridad y la vigilancia marítima. Sistemas de este tipo permiten identificar patrones de comportamiento en el tráfico de buques, detectar posibles anomalías o facilitar el análisis de grandes conjuntos de datos que, de otro modo, resultarían difíciles de interpretar mediante métodos tradicionales.

Por tanto, el empleo de inteligencia artificial en el ámbito marítimo debe ir acompañado de un enfoque ético y responsable que garantice que estas tecnologías se utilicen para reforzar las capacidades de análisis y vigilancia sin comprometer principios fundamentales como la transparencia, la seguridad o el uso adecuado de la información. De esta forma, la inteligencia artificial puede convertirse en una herramienta valiosa para mejorar el conocimiento del entorno marítimo y apoyar la toma de decisiones en este dominio.

ANEXO III: EXPLICACIÓN GENERAL DEL CÓDIGO

En este anexo se presenta una descripción general del código desarrollado para la implementación de la plataforma propuesta en este Trabajo de Fin de Grado. El objetivo de este apartado es proporcionar una visión global de la estructura del sistema, así como de los principales módulos que lo componen y su funcionamiento dentro de la arquitectura descrita en los capítulos anteriores. No se pretende realizar una explicación detallada de cada línea de código, sino ofrecer una referencia que facilite la comprensión del funcionamiento interno de la aplicación y de las decisiones de diseño adoptadas durante su desarrollo.

A3.1. Detalle de las bibliotecas empleadas

Al principio del archivo se importan todas las dependencias necesarias. Cada grupo corresponde a una parte concreta del sistema.

```
# =====  
# Imports  
# =====  
  
import os  
import re  
import json  
import time  
import math  
import difflib  
import unicodedata  
from typing import Any, Dict, Optional, Set, List, Callable, Tuple  
  
import streamlit as st  
import polars as pl  
  
from llama_index.embeddings.ollama import OllamaEmbedding  
from llama_index.core import StorageContext, load_index_from_storage  
from llama_index.vector_stores.faiss import FaissVectorStore  
from llama_index.core import Settings  
from llama_index.llms.ollama import Ollama
```

Figura A3-1 Listado de las bibliotecas empleadas

A3.2 Bloque de configuración

Después de los *imports* aparecen varias constantes. Este bloque fija los parámetros generales con los que funcionará la aplicación, incluyendo los datos de las zonas geográficas precargadas.

```
# =====  
# Configuración  
# =====  
OLLAMA_BASE_URL = os.getenv("OLLAMA_BASE_URL", "http://ollama:11434")  
LLM_MODEL = os.getenv("LLM_MODEL", "llama3.1:8b")  
EMBED_MODEL = os.getenv("EMBED_MODEL", "nomic-embed-text")  
  
RAG_PERSIST_DIR = "/app/storage/rag_faiss"  
RAG_DATA_DIR = "/tgf-rag/data/rag" # Directorio con documentos para RAG  
PARQUET_DIR = "/app/storage/parquet"  
  
MAX_LIMIT = 50  
DEFAULT_DAYS_IF_NO_TIME = 7  
  
# QC velocidad (kn)  
MIN_MOVING_SPEED_KN = 0.5  
MAX_VALID_SPEED_KN = 40.0  
  
BBOXES: Dict[str, List[float]] = {  
    "estrecho de gibraltar": [-6.15, 35.8, -5.05, 36.15],  
    "golfo de vizcaya": [-2.5, 43.3, -1.3, 44.0],  
    "islas baleares": [1.15, 38.7, 3.3, 40.0],  
    "canal de la mancha": [-3.0, 49.8, 2.0, 51.5],  
    "costa de galicia": [-9.5, 41.8, -7.5, 43.8],  
    "cabo finisterre": [-10.0, 42.5, -8.5, 44.0],  
    "mar de alboran": [-5.5, 35.5, -1.5, 37.5],  
    "costa de portugal": [-10.5, 36.8, -8.5, 41.5],  
    "golfo de cadiz": [-8.0, 35.8, -5.5, 37.2],  
    "estrecho de malaca": [98.5, 1.0, 104.5, 6.0],  
    "estrecho de hormuz": [55.0, 24.0, 57.5, 27.0],  
    "estrecho de bab el mandeb": [42.0, 11.0, 44.5, 14.5],  
    "estrecho de bosforo": [28.9, 40.9, 29.2, 41.3],  
    "estrecho de dardanelos": [26.0, 39.9, 26.8, 40.6],  
    "estrecho de bering": [-170.0, 65.5, -167.0, 67.5],  
    "estrecho de magallanes": [-75.5, -54.8, -69.0, -52.5],  
}
```

Figura A3-2 Configuración

A3.3 Columnas y *helpers* básicos

Se definen las columnas que pertenecen a los datos dinámicos y a los datos estáticos.

```
# =====
# Columnas
# =====
DYNAMIC_COLS: Set[str] = {"timestamp", "mmsi", "lat", "lon", "speed", "course", "heading"}
STATIC_COLS_RAW: Set[str] = {
    "timestamp", "MMSI", "IMO", "to_bow", "to_stern", "to_port", "to_starboard", "draught", "shiptype"
}
STATIC_COLS_NORM: Set[str] = {
    "timestamp", "mmsi", "IMO", "to_bow", "to_stern", "to_port", "to_starboard", "draught", "shiptype"
}
DERIVED_STATIC_COLS: Set[str] = {"length_m", "beam_m"}
```

Figura A3-3 Detalle de las columnas definidas

Luego aparecen varios *helpers* pequeños: funciones para limpiar prefijos, convertir valores a entero o flotante, limitar rangos, eliminar duplicados o formatear números. Son piezas simples, pero muy útiles porque se reutilizan constantemente en el resto del archivo.

```
# =====
# Helpers básicos
# =====
def _strip_prefix(col: str) -> str:
    if isinstance(col, str) and "." in col:
        return col.split(".", 1)[1]
    return col

def _coerce_value(v: Any) -> Any:
    if isinstance(v, (int, float, bool)):
        return v
    if v is None:
        return None
    if isinstance(v, str):
        s = v.strip()
        if s == "":
            return None
        if re.fullmatch(r"-?\d+", s):
            try:
                return int(s)
            except Exception:
                return s
        if re.fullmatch(r"-?\d+(\.\d+)?", s):
            try:
                return float(s)
            except Exception:
                return s
        if s.lower() in ("true", "false"):
            return s.lower() == "true"
        return s
    return v

def _safe_int(x: Any, default: int) -> int:
    try:
        return int(_coerce_value(x))
    except Exception:
        return default

def _safe_float(x: Any, default: float) -> float:
    try:
        return float(_coerce_value(x))
    except Exception:
        return default

def _unique_cols(cols: List[str]) -> List[str]:
    seen = set()
    out = []
    for c in cols:
        if c not in seen:
            out.append(c)
            seen.add(c)
    return out

def _clamp(n: int, lo: int, hi: int) -> int:
    return max(lo, min(hi, n))

def _fmt_num(x: Any, nd: int = 2) -> str:
    """Formatea número para display."""
    try:
        xf = float(x)
        if xf.is_integer():
            return str(int(xf))
        return f"{xf:.{nd}f}"
    except Exception:
        return str(x)
```

Figura A3-4 *Helpers* básicos

A3.4 Extracción de parámetros desde la pregunta

Este bloque intenta obtener información directamente del texto escrito por el usuario, sin depender todavía del LLM.

```
# =====
# Extractores de parámetros
# =====

def extract_mmsi_from_text(q: str) -> Optional[int]:
    q1 = (q or "").lower()
    m = re.search(r"\bmmsi\D*(\d{9})\b", q1)
    if m:
        return int(m.group(1))
    m2 = re.search(r"\b(\d{9})\b", q or "")
    if m2:
        return int(m2.group(1))
    return None

def extract_imo_from_text(q: str) -> Optional[int]:
    q1 = (q or "").lower()
    m = re.search(r"\bimo\D*(\d{7})\b", q1)
    if m:
        return int(m.group(1))
    return None

def extract_days_window(q: str) -> Optional[int]:
    """Extrae ventana temporal con múltiples patrones (word boundaries)."""
    q1 = (q or "").lower()

    patterns = [
        r"(?:de\s+)?(?:los\s+)?(?:últim|últim|last)\w*\s+(\d+)\s+(?:día|dias|días|day|days)\b",
        r"\ben\s+(?:los\s+)?(\d+)\s+(?:día|dias|días|day|days)\b",
        r"\bhace\s+(\d+)\s+(?:día|dias|días|day|days)\b",
        r"(\d+)\s+(?:día|dias|días|day|days)\s+(?:atrás|atras|back|ago)\b",
    ]

    for pat in patterns:
        m = re.search(pat, q1)
        if m:
            try:
                return int(m.group(1))
            except Exception:
                pass
    return None

def extract_k_from_text(q: str) -> Optional[int]:
    q1 = (q or "").lower()
    m = re.search(r"\btóp\s+(\d+)\b", q1)
    if m:
        return int(m.group(1))
    m2 = re.search(r"\blos\s+(\d+)\b", q1)
    if m2:
        return int(m2.group(1))
    return None

def _note_default_window(days: int) -> str:
    return f"Ventana temporal: últimos {days} días (por defecto al no especificar)."
```

```
def get_lazyframe(source: str) -> pl.LazyFrame:
    source = (source or "").lower().strip()
    if source == "dynamic":
        path = f"{PARQUET_DIR}/dynamic/*.parquet"
    elif source == "static":
        path = f"{PARQUET_DIR}/static/*.parquet"
    else:
        path = f"{PARQUET_DIR}/other/*.parquet"
    return pl.scan_parquet(path)

def _prepare_static(static_lf: pl.LazyFrame) -> pl.LazyFrame:
    cols = static_lf.columns
    if "MMSI" in cols and "mmsi" not in cols:
        static_lf = static_lf.rename({"MMSI": "mmsi"})
    return static_lf

def _add_static_derived(lf: pl.LazyFrame) -> pl.LazyFrame:
    cols = set(lf.columns)
    if {"to_bow", "to_stern"}.issubset(cols):
        lf = lf.with_columns([pl.col("to_bow") + pl.col("to_stern").alias("length_m")])
    if {"to_port", "to_starboard"}.issubset(cols):
        lf = lf.with_columns([pl.col("to_port") + pl.col("to_starboard").alias("beam_m")])
    return lf

def _latest_static(static_lf: pl.LazyFrame) -> pl.LazyFrame:
    static_lf = _prepare_static(static_lf)
    if "timestamp" in static_lf.columns:
        latest = (
            static_lf
            .sort(["mmsi", "timestamp"])
            .group_by("mmsi")
            .agg(pl.all().last())
        )
    else:
        latest = static_lf.group_by("mmsi").agg(pl.all().first())
    return _add_static_derived(latest)

def apply_bbox_filter(lf: pl.LazyFrame, bbox: Optional[List[float]]) -> pl.LazyFrame:
    if not bbox or not isinstance(bbox, list) or len(bbox) != 4:
        return lf
    if not {"lat", "lon"}.issubset(set(lf.columns)):
        return lf
    try:
        minlon, minlat, maxlon, maxlat = map(float, bbox)
    except Exception:
        return lf
    return lf.filter(
        (pl.col("lon") >= minlon) & (pl.col("lon") <= maxlon) &
        (pl.col("lat") >= minlat) & (pl.col("lat") <= maxlat)
    )
```

Figura A3-5 Extractores de parámetros

Función	Qué detecta
extract_mmsi_from_text	Busca un MMSI en la pregunta.
extract_imo_from_text	Busca un IMO.
extract_days_window	Detecta expresiones como “últimos 7 días”.

Tabla A3-1 Ejemplos de los extractores

La idea es que, si el usuario ya da un dato clave, el sistema pueda aprovecharlo directamente y construir una ruta más fiable.

A3.5 Acceso a datos y caché

Prepara recursos costosos y los guarda en caché con Streamlit.

```
# =====
# Cache: LLM + RAG + Static + Anchor temporal
# =====
@st.cache_resource(show_spinner=False)
def get_llm_cached() -> Ollama:
    Settings.llm = Ollama(
        model=LLM_MODEL,
        base_url=OLLAMA_BASE_URL,
        request_timeout=180.0,
        temperature=0.0,
    )
    Settings.embed_model = OllamaEmbedding(model_name=EMBED_MODEL, base_url=OLLAMA_BASE_URL)
    return Settings.llm

@st.cache_resource(show_spinner=False)
def get_rag_query_engine_cached(similarity_top_k: int = 2):
    """Carga RAG usando rag_manager (inteligente)."""
    try:
        from rag_manager import get_rag_query_engine
        return get_rag_query_engine(similarity_top_k=similarity_top_k)
    except ImportError:
        # Fallback si no existe rag_manager
        try:
            vector_store = FaissVectorStore.from_persist_dir(persist_dir=RAG_PERSIST_DIR)
            storage_context = StorageContext.from_defaults(vector_store=vector_store, persist_dir=RAG_PERSIST_DIR)
            index = load_index_from_storage(storage_context)
            return index.as_query_engine(similarity_top_k=similarity_top_k)
        except Exception as e:
            if st.sidebar.checkbox("Mostrar errores RAG", value=False):
                st.warning(f'RAG no disponible: {e}')
            return None
```

```
@st.cache_resource(show_spinner=False)
def get_static_latest_df() -> pl.DataFrame:
    lf = pl.scan_parquet(f'{PARQUET_DIR}/static/*.parquet')
    lf = lf.filter(lf.timestamp() >= get_dynamic_max_timestamp())
    return lf.collect(streaming=True)

@st.cache_resource(show_spinner=False)
def get_static_by_mmsi() -> Dict[int, Dict[str, Any]]:
    df = get_static_latest_df()
    out: Dict[int, Dict[str, Any]] = {}
    for r in df.to_dicts():
        m = r.get("mmsi")
        if isinstance(m, int):
            out[m] = r
        elif isinstance(m, str) and m.isdigit():
            out[int(m)] = r
    return out

@st.cache_resource(show_spinner=False)
def get_dynamic_max_timestamp() -> int:
    lf = pl.scan_parquet(f'{PARQUET_DIR}/dynamic/*.parquet')
    if "timestamp" not in lf.columns:
        return int(time.time())
    try:
        mx = lf.select(pl.col("timestamp").max().alias("mx")).collect(streaming=True)["mx"][0]
        return int(mx) if mx is not None else int(time.time())
    except Exception:
        return int(time.time())
```

```
def default_time_filters(days: int) -> Dict[str, int]:
    anchor = get_dynamic_max_timestamp()
    from_ts = int(anchor - days * 86400)
    return {"time_from": from_ts, "time_to": int(anchor)}

def resolve_mmsi_from_imo(imo: int) -> Optional[int]:
    try:
        imo_int = int(_coerce_value(imo))
    except Exception:
        return None
    df = get_static_latest_df()
    if "IMO" not in df.columns or "mmsi" not in df.columns:
        return None
    hit = df.filter(pl.col("IMO") == imo_int).select(["mmsi"]).head(1)
    if hit.height == 0:
        return None
    try:
        return int(hit["mmsi"][0])
    except Exception:
        return None
```

Figura A3-6 Acceso a los datos y caché

Función	Qué hace
<code>get_llm_cached()</code>	Crea y deja listo el modelo Ollama y el modelo de <i>embeddings</i> .
<code>get_static_latest_df()</code> <code>get_static_by_mmsi()</code>	Preparan una versión limpia y accesible de los datos estáticos.
<code>get_dynamic_max_timestamp()</code>	Busca el último <i>timestamp</i> disponible en los datos dinámicos para anclar la ventana temporal por defecto.

Tabla A3-2 Ejemplos de las funciones

Gracias a estas funciones, la aplicación no repite trabajo pesado en cada consulta y responde más rápido.

A3.6 Clasificación

Este es uno de los bloques más importantes. El clasificador decide cómo interpretar la pregunta y qué herramienta o secuencia de herramientas debe ejecutarse.

- Primero intenta resolver casos muy concretos de forma determinista.
 - E.g: Si detecta un MMSI y la pregunta es claramente sobre velocidad, puede saltarse parte del razonamiento y construir un plan directo.
- Si no basta con reglas simples, construye un *prompt* para el LLM con ayuda del contexto RAG de *routing.md*.
- El resultado esperado del clasificador es un JSON con la ruta, la herramienta y los argumentos.

Además, el archivo define validadores por herramienta. Su función es revisar y normalizar los argumentos que ha producido el clasificador antes de ejecutar nada. Esto reduce mucho los fallos por valores mal formados.

```
# =====
# RAG retrieval (routing + intent + context)
# =====
def retrieve_routing_context(qe, question: str, max_chars: int = 800) -> str:
    """
    PUNTO 1: Recupera contexto de 'routing.md' para ayudar al LLM a enrutar.

    Documento usado: routing.md (rol: "routing")
    Contenido: Reglas para decidir entre ruta determinista y analítica

    Máximo: 800 caracteres
    Cuándo: SIEMPRE (antes de enrutamiento)
    """
    if not qe:
        return ""
    try:
        resp = qe.query(question)
        nodes = getattr(resp, "source_nodes", None)
        if not nodes:
            return str(resp)[:max_chars]
        parts: List[str] = []
        for i, sn in enumerate(nodes[2], start=1):
            text = (getattr(sn, "text", "") or "").strip()
            if text:
                # Filtrar solo documentos con rol "routing"
                meta = getattr(sn, "metadata", {}) or {}
                roles = meta.get("roles", ["routing", "analysis", "writing"])
                if "routing" not in roles:
                    continue # Saltear si no tiene rol routing
                parts.append(text[:500])
        return "\n".join(parts).strip()[:max_chars]
    except Exception:
        return ""

def retrieve_context_for_analysis(qe, question: str, max_chars: int = 1200) -> str:
    """
    PUNTO 2: Recupera contexto de 'analysis.md' para análisis complejos.

    Documento usado: analysis.md (rol: "analysis")
    Contenido: Reglas para análisis (proximidad, patrones, anomalías, zonas)

    Máximo: 1200 caracteres
    Cuándo: SOLO en análisis queries (después de enrutamiento)
    """
    if not qe:
        return ""
    try:
        resp = qe.query(question)
        nodes = getattr(resp, "source_nodes", None)
        if not nodes:
            return str(resp)[:max_chars]
        parts: List[str] = []
        budget = max_chars
        for i, sn in enumerate(nodes, start=1):
            text = (getattr(sn, "text", "") or "").strip()
            if not text:
                continue
            # Filtrar solo documentos con rol "analysis"
            meta = getattr(sn, "metadata", {}) or {}
            roles = meta.get("roles", ["routing", "analysis", "writing"])
            if "analysis" not in roles:
                continue # Saltear si no tiene rol analysis
            src = meta.get("file_path") or meta.get("source") or meta.get("filename") or ""
            header = f"[RAG {i}] {src}".strip()
            chunk = header + "\n" + text
            if len(chunk) > budget:
                chunk = chunk[:budget]
            parts.append(chunk)
            budget -= len(chunk)
            if budget <= 0:
                break
        return "\n".join(parts).strip()
    except Exception:
        return ""
```

Figura A3-7 Recuperación de contexto para clasificación

```
# =====
# JSON extraction + repair
# =====
def extract_json(text: Any) -> Any:
    if not isinstance(text, str):
        text = str(text)
    s = text.strip()

    fence = re.search(r"```(?:json)?(.*?)```", s, flags=re.DOTALL | re.IGNORECASE)
    if fence:
        return json.loads(fence.group(1).strip())

    try:
        return json.loads(s)
    except Exception:
        pass

    starts = [i for i in (s.find("{"), s.find("[")) if i != -1]
    if not starts:
        raise ValueError("No se encontró JSON en la respuesta del modelo.")
    starts.sort()

    for start in starts:
        sub = s[start:]
        for end in range(len(sub), 1, -1):
            if sub[end - 1] not in ("}", "]", "\n"):
                continue
            candidate = sub[:end].strip()
            try:
                return json.loads(candidate)
            except Exception:
                continue
    raise ValueError("No se encontró ningún bloque JSON válido en la respuesta del modelo.")

def repair_json_with_llm(llm: Ollama, raw_text: str, schema_hint: str) -> Any:
    prompt = f"""
    Repara y devuelve SOLO JSON válido (sin texto extra).

    Esquema esperado:
    {schema_hint}

    Texto:
    {raw_text}

    JSON:
    """
    resp = llm.complete(prompt)
    return extract_json(str(resp))
```

Figura A3-8 Validación de la ruta escogida

A3.7 Herramientas deterministas: trabajo real sobre los datos

Las herramientas son funciones que consultan y procesan los datos AIS usando *Polars*. Cada una hace una tarea concreta.

```
def tool_dimensions(mmsi: Optional[int] = None,
                   imo: Optional[int] = None,
                   mmsi_list: Optional[List[int]] = None) -> Dict[str, Any]:
    if mmsi is None and imo is not None:
        mmsi = resolve_mmsi_from_imo(int(imo))
        if mmsi is None:
            return {"kind": "error", "message": f"No se encontró MMSI asociado al IMO {imo}."}
    if mmsi_list is None and mmsi is not None:
        mmsi_list = [int(mmsi)]

    if not mmsi_list:
        df = get_static_latest_df()
    else:
        df = get_static_latest_df().filter(pl.col("mmsi").is_in([int(x) for x in mmsi_list]))

    cols = [c for c in ["mmsi", "length_m", "beam_m"] if c in df.columns]
    if not cols:
        return {"kind": "error", "message": "No se pudieron calcular dimensiones (faltan columnas)."}
    out = df.select(cols)
    return {
        "kind": "table",
        "rows": out.height,
        "data": out.to_dicts(),
        "meta": {"tool": "dimensions", "timestamp": time.time()}
    }
```

Figura A3-9 Ejemplo del código de una herramienta

Herramienta	Función general
tool_static_profile	Devuelve perfil estático de uno o varios barcos.
tool_dimensions	Calcula o devuelve dimensiones como eslora y manga.
tool_track	Extrae una trayectoria o conjunto de posiciones de un MMSI.
tool_aggregate_dynamic	Hace agregaciones sobre datos dinámicos, como conteos o velocidades medias.
tool_aggregate_static	Hace agregaciones sobre campos estáticos.
tool_rank	Ordena resultados por una métrica, por ejemplo, velocidad o número de mensajes.
tool_topk_static	Obtiene los barcos con valores estáticos más altos.
tool_join_static_dynamic	Cruza información estática con una lista de MMSI.
tool_sanity_report	Ejecuta controles de calidad sobre los datos usados.

Tabla A3-3 Otras herramientas utilizadas

Todas estas herramientas devuelven resultados en una estructura bastante uniforme: tipo de salida, datos y metadatos. Eso hace más fácil tratarlas después de forma común.

A3.8 Redacción de la respuesta

Una vez ejecutadas las herramientas, el archivo todavía tiene que convertir el resultado técnico en una respuesta entendible para el usuario.

```
def write_deterministic_answer_llm(llm: Ollama, question: str, exec_result: Dict[str, Any], rag_context: str = "") -> str:
    prompt = build_deterministic_writer_prompt(question, exec_result, rag_context)
    return str(llm.complete(prompt)).strip()

def write_analysis_answer_llm(llm: Ollama, question: str, exec_result: Dict[str, Any], rag_context: str = "") -> str:
    prompt = build_analysis_writer_prompt(question, exec_result, rag_context)
    return str(llm.complete(prompt)).strip()
```

Figura A3-10 Ejemplo del código de una llamada al modelo para la generación de una respuesta

Función	Qué hace
<code>build_deterministic_writer_prompt()</code>	Construye el <i>prompt</i> para que el LLM redacte una respuesta breve a partir de un resultado determinista.
<code>build_analysis_writer_prompt()</code>	Crea un <i>prompt</i> más elaborado para consultas analíticas.
<code>write_deterministic_answer_llm()</code> <code>write_analysis_answer_llm()</code>	Llaman al modelo y devuelven el texto final.

Tabla A3-4 Otras funciones utilizadas

ANEXO IV: GLOSARIO DE SIGLAS

En este anexo se presenta un listado de las siglas y acrónimos utilizados a lo largo de la memoria. Su inclusión tiene como objetivo facilitar la comprensión del documento, proporcionando el significado de aquellos términos abreviados que aparecen en los distintos apartados del trabajo.

- AIS: *Automatic Identification System*
- API: *Application Programming Interface*
- CPU: *Central Processing Unit*
- CSV: *Comma-Separated Values*
- FAISS: *Facebook AI Similarity Search*
- GMT: *Greenwich Mean Time*
- GPU: *Graphics Processing Unit*
- IBM: *International Business Machines*
- IMO: *International Maritime Organization*
- JSON: *JavaScript Object Notation*
- LLM: *Large Language Model*
- LTS: *Long-Term Support*
- MMSI: *Maritime Mobile Service Identity*
- NATO: *North Atlantic Treaty Organization*
- NLIDB: *Natural Language Interface to Databases*
- OLAP: *Online Analytical Processing*
- PDF: *Portable Document Format*
- RAG: *Retrieval-Augmented Generation*
- RAM: *Random Access Memory*
- RTX: *Ray Tracing Texel eXtreme* (arquitectura de GPU de NVIDIA)
- SOLAS: *Safety of Life at Sea*
- SQL: *Structured Query Language*
- ZSA: *Zero-Shot Analysis*

ANEXO V: GUÍA DE INSTALACIÓN Y CONFIGURACIÓN DEL ENTORNO DE EJECUCIÓN

Instalación del *Docker*

```
# Add Docker's official GPG key:
sudo apt update
sudo apt install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg
-o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
sudo tee /etc/apt/sources.list.d/docker.sources <<EOF
Types: deb
URIs: https://download.docker.com/linux/ubuntu
Suites: $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF
sudo apt update
```

Nvidia Container Toolkit

Prerrequisitos

```
sudo apt-get update && sudo apt-get install -y --no-install-recommends \curl\ gnupg2
```

Configuración Repositorio

```
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor
-o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
    && curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-
container-toolkit.list | \
    sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-
toolkit-keyring.gpg] https://#g' | \
    sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
```

Actualizar paquetes de la lista del repositorio

```
sudo apt-get update
```

Instalar paquetes Nvidia container toolkit

```
export NVIDIA_CONTAINER_TOOLKIT_VERSION=1.18.2-1
sudo apt-get install -y \
    nvidia-container-toolkit=${NVIDIA_CONTAINER_TOOLKIT_VERSION} \
    nvidia-container-toolkit-base=${NVIDIA_CONTAINER_TOOLKIT_VERSION} \
```

```
libnvidia-container-tools=${NVIDIA_CONTAINER_TOOLKIT_VERSION} \  
libnvidia-container1=${NVIDIA_CONTAINER_TOOLKIT_VERSION}
```

Configurar Docker

```
sudo nvidia-ctk runtime configure --runtime=docker
```

Reiniciar Docker Daemon

```
sudo systemctl restart docker
```

Ejecutar Ollama dentro de un Contenedor *Docker*

```
docker run -d --gpus=all -v ollama:/root/.ollama -p 11434:11434 --name ollama_carlos  
ollama/ollama
```

Descargar modelos

```
docker exec -it ollama_carlos ollama pull llama3.1 (Latest)  
docker exec -it ollama_carlos ollama pull llama3.1:8b  
docker exec -it ollama_carlos ollama pull nomic-embed-text
```

WebUI on *Docker*

Ejecutar el último Open WebUI Docker image desde GitHub Container Registry

```
docker pull ghcr.io/open-webui/open-webui:main
```

Conectar WebUI a ollama en otro servidor

```
sudo docker run -d -p 3000:8080 -e OLLAMA_BASE_URL=http://193.146.212.201:11434/ -  
v open-webui:/app/backend/data --name webui_carlos_ollama --restart always  
ghcr.io/open-webui/open-webui:main
```