Arquitectura End-to-End para la Consulta Inteligente de Manuales Vehiculares mediante IA y Cloud Computing

Nicolás Santiago Calderón Sua, Universidad Santo Tomás, Colombia, nicolas.calderon@ustabuca.edu.co
Carlos D. Pimentel-Díaz, Universidad Santo Tomás Seccional Bucaramanga

Resumen - Este artículo presenta una arquitectura de inteligencia artificial para un sistema de asistencia vehicular inteligente, optimizado para los manuales del modelo Cupra Formentor. Aborda el reto de gestionar la complejidad de la documentación técnica extensa y la limitación de las soluciones genéricas como ChatGPT, que a menudo carecen de la precisión y trazabilidad necesarias para consultas específicas de vehículos. La solución implementa un pipeline RAG (Retrieval-Augmented Generation), integrando Procesamiento de Lenguaje Natural avanzado. La arquitectura se despliega en AWS, utilizando Lambda para procesamiento serverless, S3 para almacenamiento y PostgreSQL con pgvector para una gestión optimizada de embeddings y recuperación de información. El sistema permite consultas personalizadas fundamentadas exclusivamente en la documentación oficial, eliminando alucinaciones. La evaluación experimental demostró una alta fiabilidad, con un puntaje de calidad de 8.7/10, 100% de trazabilidad y cero alucinaciones. Superó a ChatGPT 3.5 en precisión técnica y claridad, validando la superioridad del enfoque RAG en dominios críticos. Esta implementación automatiza la consulta de información vehicular, mejorando la experiencia del usuario. Como trabajo futuro, se contempla la expansión de capacidades multimodales (visión por computadora para símbolos, entrada de audio) y la adaptación multilingüe, buscando establecer nuevos estándares para la asistencia proactiva en el ecosistema automotriz conectado.

Palabras claves: Asistencia Vehicular, Bases de Datos Vectoriales, Fiabilidad, Inteligencia Artificial (IA), Multimodalidad, Retrieval-Augmented Generation (RAG).

Artículo científico presentado como opción de grado para optar por el título de Ingeniero Mecatronico.

Autor de contacto: N. S. Calderón Sua es Ingeniero Mecatrónico, egresado de la Universidad Santo Tomás, Bucaramanga, Colombia. Actualmente cursa una Maestría en Big Data en la Universidad La Salle, Barcelona, España (autor de contacto: teléfono: +34 657970744; correo electrónico: nicolas.calderon@ustabuca.edu.co).

Co-Autor y Director: C. D. Pimentel Díaz es Ingeniero Electrónico, Especialista en Control e Instrumentación Industrial y Magíster en Inteligencia Artificial. Actualmente está vinculado como docente e investigador en la Universidad Santo Tomás, Bucaramanga, Colombia (correo electrónico: carlos.pimentel@ustabuca.edu.co).

.

Scientific article template (*End-to-End Architecture for Intelligent Vehicle Manual Querying Using AI and Cloud Computing*)

Abstract - This article presents an artificial intelligence architecture for an intelligent vehicle assistance system, optimized for Cupra Formentor model manuals. It addresses the challenge of managing the complexity of extensive technical documentation and the limitations of generic solutions such as ChatGPT, which often lack the precision and traceability required for vehicle-specific queries. The solution implements a Retrieval-Augmented Generation (RAG) pipeline, integrating advanced Natural Language Processing. The architecture is deployed on AWS, using Lambda for serverless processing, S3 for storage, and PostgreSQL with pgvector for optimized embedding management and information retrieval. The system allows customized queries based exclusively on official documentation, eliminating artifacts. Experimental evaluation demonstrated high reliability, with a quality score of 8.7/10, 100% traceability, and zero artifacts. It outperformed ChatGPT 3.5 in technical precision and clarity, validating the superiority of the RAG approach in critical domains. This implementation automates vehicle information retrieval, improving the user experience. Future work includes expanding multimodal capabilities (computer vision for symbols, audio input) and multilingual adaptation, seeking to establish new standards for proactive assistance in the connected automotive ecosystem.

Keywords: Artificial Intelligence (AI), Multimodality, Reliability, Retrieval-Augmented Generation (RAG), Vehicle Assistance, Vector Databases.

I. INTRODUCCIÓN

La proliferación de tecnologías avanzadas en la industria automotriz ha llevado a una complejidad creciente en los manuales de usuario vehiculares. Estos documentos, a menudo extensos y densamente técnicos, dificultan a los propietarios y personal de servicio la rápida localización de información específica para diagnósticos, mantenimiento o resolución de problemas [1]. Esta barrera de acceso genera frustración en el usuario final y puede derivar en ineficiencias operacionales para los centros de asistencia técnica, quienes dedican recursos

significativos a la interpretación de estos volúmenes de información.

Adicionalmente, la creciente adopción de modelos de lenguaje grandes (LLM) genéricos, como ChatGPT, para la consulta de información general ha expuesto una limitación fundamental en contextos especializados. Si bien estos sistemas demuestran una notable capacidad de generación de texto coherente, su naturaleza no fundamentada en fuentes verificadas los hace propensos a la 'alucinación' de datos, es decir, a la invención de información plausible pero incorrecta [2]. En el ámbito de la asistencia vehicular, donde la precisión y la veracidad son críticas para la seguridad y el correcto funcionamiento de los sistemas, la dependencia de respuestas estándar, genéricas o potencialmente erróneas derivadas de LLMs no especializados representa un riesgo significativo. Estas soluciones carecen de la capacidad de proporcionar respuestas específicas y verificables para fallas o características detalladas de modelos vehiculares particulares, lo que subraya la necesidad de un enfoque que garantice la trazabilidad y la fiabilidad de la información.

En respuesta a la problemática planteada, se ha concebido y desarrollado una arquitectura end-to-end para la consulta inteligente de manuales vehiculares, específicamente diseñada para la marca Cupra. Esta solución se fundamenta en el paradigma de Recuperación Aumentada por Generación (RAG), el cual se presenta como una alternativa robusta para superar las limitaciones de los LLM genéricos en dominios de conocimiento específicos. A diferencia de los modelos puramente generativos, un sistema RAG primero recupera información relevante de un corpus de datos autorizado antes de generar una respuesta, asegurando así la precisión, la fundamentación factual y la trazabilidad de las respuestas proporcionadas [3].

La arquitectura implementada se centra en el procesamiento de lenguaje natural (NLP), que es fundamental para la ingestión, segmentación y vectorización semántica de los manuales de texto. Este proceso transforma el conocimiento no estructurado contenido en la documentación oficial en un formato optimizado para la consulta, permitiendo una extracción de información precisa y contextualizada. Si bien la implementación actual se enfoca en el procesamiento textual, se reconoce el potencial de expansión futura para integrar capacidades como la visión por computadora, que permitirían al sistema interpretar elementos gráficos como símbolos del salpicadero o diagramas técnicos, enriqueciendo la capacidad de comprensión del sistema y facilitando una interacción multimodal más completa a futuro.

El objetivo central de esta propuesta es ofrecer una herramienta de asistencia vehicular altamente precisa y contextualmente relevante, capaz de reducir drásticamente el tiempo de búsqueda de información, minimizar errores diagnósticos y mejorar la experiencia tanto de los propietarios de vehículos como del personal técnico. Al centrarse en los manuales de Cupra, el sistema garantiza una especificidad y fiabilidad que las soluciones de IA de propósito general no pueden alcanzar, estableciendo un nuevo estándar en la interacción con la documentación técnica automotriz.

La contribución principal de la presente investigación reside en el diseño, desarrollo y evaluación rigurosa de una arquitectura RAG especializada y adaptada para la consulta inteligente de documentación técnica altamente estructurada y voluminosa, como son los manuales vehiculares. A diferencia de las aproximaciones genéricas, este trabajo demuestra la viabilidad y la superioridad de un sistema diseñado específicamente para garantizar la precisión contextual, la fiabilidad de la información y la completa trazabilidad de las respuestas hasta su fuente original dentro del manual [4]. La metodología implementada para la ingestión, segmentación semántica y vectorización de contenido, junto con la estratégica integración de una base de datos vectorial sobre PostgreSQL, establece un precedente para el manejo eficiente de grandes corpus documentales en dominios críticos.

Además, el estudio no solo valida la eficacia de la solución propuesta en el ámbito automotriz, sino que también subraya su inherente aplicabilidad transversal. Los principios arquitectónicos y las técnicas desarrolladas son inherentemente extensibles a otros dominios técnicos donde la exactitud de la información es primordial y el volumen de documentación es considerable, tales como la aeronáutica, la medicina diagnóstica o la industria manufacturera [5]. Esta adaptabilidad posiciona la presente investigación como un aporte significativo para la optimización de la gestión del conocimiento y la toma de decisiones asistida por IA en diversos sectores de alta complejidad.

II. DESARROLLO DE CONTENIDOS

A. Marco Teórico

La arquitectura implementada en este trabajo se fundamenta en una de las técnicas más avanzadas de la inteligencia artificial contemporánea: Retrieval-Augmented Generation (RAG). Esta técnica combina la recuperación de información relevante con modelos de lenguaje generativo, superando las limitaciones de los enfoques puramente generativos que tienden a producir respuestas vagas, descontextualizadas o carentes de trazabilidad. A continuación, se describen los fundamentos teóricos esenciales que sustentan el diseño del sistema propuesto.

1) Recuperación aumentada por generación (RAG)

La Recuperación Aumentada por Generación (RAG, por sus siglas en inglés, Retrieval-Augmented Generation) representa una evolución significativa en el campo de los modelos de lenguaje grandes (LLM), abordando una de sus principales limitaciones: la propensión a generar 'alucinaciones' o información no verificable [3]. A diferencia de los LLM que operan exclusivamente basándose en el conocimiento intrínseco adquirido durante su fase de entrenamiento, una arquitectura RAG incorpora un mecanismo explícito de recuperación de información externa y autorizada.

Este proceso dota al modelo de la capacidad de acceder a un corpus de datos específico y actual en tiempo real, lo que le permite fundamentar sus respuestas en hechos verificables [6].

El paradigma RAG se articula típicamente en dos fases interconectadas: una fase de recuperación (retrieval) y una fase de generación (generation). En la etapa de recuperación, el sistema analiza la consulta del usuario para identificar los pasajes o 'chunks' más relevantes dentro de una base de conocimiento externa preindexada, utilizando técnicas de búsqueda por similitud vectorial. Posteriormente, en la fase de generación, estos pasajes recuperados se integran junto con la consulta original en un prompt enriquecido que se presenta al LLM. De esta manera, el modelo no solo tiene acceso a su vasto conocimiento paramétrico, sino que también dispone de evidencia documental explícita para guiar su proceso de generación de respuesta, lo que mejora drásticamente la precisión, reduce la inventiva no deseada y facilita la trazabilidad de la información al citar sus fuentes [7]. Esta capacidad es fundamental en dominios donde la exactitud es crítica, como la asistencia técnica vehicular, donde un error puede tener consecuencias significativas.

2) Representaciones vectoriales semánticas (Embeddings)

Las representaciones vectoriales semánticas, o embeddings, constituyen un pilar fundamental en las arquitecturas de procesamiento de lenguaje natural (NLP) modernas, especialmente en sistemas como RAG. Estos vectores son representaciones numéricas de palabras, frases o documentos completos en un espacio multidimensional, donde la distancia geométrica entre vectores correlaciona directamente con la similitud semántica o contextual entre los elementos representados [8]. Es decir, conceptos similares se agrupan en regiones próximas de este espacio vectorial, mientras que los disímiles se ubican a mayor distancia.

La generación de estos embeddings se logra a través de modelos de lenguaje pre-entrenados, que han sido expuestos a vastas cantidades de texto y han aprendido a capturar las relaciones contextuales y significados inherentes al lenguaje humano. Para el presente proyecto, se ha empleado el modelo text-embedding-ada-002 de OpenAI. La elección de este modelo se fundamenta en su demostrada capacidad para generar embeddings de alta calidad que capturan eficientemente la semántica de documentos técnicos, su robustez en diversas tareas de comprensión del lenguaje y su eficiencia computacional para procesar grandes volúmenes de texto de manera escalable [9]. La conversión del contenido textual de los manuales vehiculares a estas representaciones vectoriales es un paso crítico, ya que permite que la información, inicialmente no estructurada, sea indexada y recuperada eficientemente a través de algoritmos de búsqueda por similitud, posibilitando así la identificación de los pasajes más relevantes ante una consulta del usuario.

3) Almacenamiento vectorial y recuperación por similitud

El almacenamiento eficiente y la recuperación rápida de las representaciones vectoriales semánticas son elementos críticos en cualquier sistema RAG. Una vez que el contenido textual de los manuales vehiculares se transforma en embeddings, estos vectores deben ser persistidos en una base de datos optimizada para la búsqueda por similitud vectorial. A diferencia de las bases de datos relacionales tradicionales, que están diseñadas para búsquedas exactas basadas en claves o atributos, las bases de datos vectoriales están optimizadas para identificar los vectores más "cercanos" a un vector de consulta determinado en un espacio de alta dimensionalidad [10].

Para el presente sistema, se ha optado por utilizar PostgreSQL con la extensión pgvector como solución de almacenamiento vectorial. Esta elección se justifica por la robustez y familiaridad de PostgreSQL como base de datos relacional, lo que permite gestionar simultáneamente tanto los vectores de embeddings como los metadatos asociados a cada fragmento textual (como el título del capítulo, el número de página o el índice del subchunk). La extensión pgvector añade capacidades nativas de almacenamiento y búsqueda eficiente de vectores de alta dimensionalidad, implementando algoritmos de búsqueda por vecinos más cercanos aproximados (ANN, por sus siglas en inglés, Approximate Nearest Neighbors) [11]. Estos algoritmos, como HNSW (Hierarchical Navigable Small World) o IVFFlat, permiten encontrar los k vectores más cercanos a una consulta de manera computacionalmente eficiente, incluso en conjuntos de datos muy grandes, sacrificando una mínima precisión por una mejora drástica en el rendimiento. Esta arquitectura híbrida ofrece la flexibilidad de un sistema relacional con la eficiencia necesaria para las operaciones de búsqueda por similitud requeridas en la fase de recuperación de RAG, asegurando que los pasajes más relevantes del manual sean identificados con rapidez y precisión para fundamentar las respuestas del LLM.

4) Ingeniería de prompts y control de generación

La efectividad de un sistema RAG no solo depende de la calidad de la recuperación de información, sino también de la formulación estratégica de las instrucciones presentadas al modelo de lenguaje grande (LLM), un proceso conocido como ingeniería de prompts. Un prompt bien diseñado guía al LLM para generar respuestas que no solo sean coherentes y gramaticalmente correctas, sino también precisas, relevantes y fundamentadas en la información recuperada [12]. Para este sistema, el prompt se construye dinámicamente, integrando la consulta original del usuario con los fragmentos de texto más relevantes obtenidos de la base de datos vectorial. Esto asegura que el LLM opere dentro de un contexto específico y delimitado por la documentación oficial.

Adicionalmente, para elevar el nivel de confiabilidad y mitigar cualquier riesgo residual de alucinación, se ha incorporado un mecanismo de control de generación basado en un agente de calidad. Este componente evalúa la respuesta generada por el LLM en función de criterios predefinidos que incluyen la relevancia contextual, la precisión fáctica, la

completitud de la información y la capacidad de trazabilidad hacia los documentos fuente [13]. Este agente opera de manera post-generación, asignando una calificación cuantitativa que refleja la confianza del sistema en la respuesta proporcionada. Tal enfoque multicapa, que combina una recuperación robusta, una ingeniería de prompts cuidadosa y una validación posterior a la generación, es fundamental para asegurar que las respuestas entregadas al usuario final no solo sean útiles, sino también rigurosamente verificables y seguras en un dominio tan crítico como la asistencia vehicular.

B. Revisión de la Literatura

La presente investigación se enmarca en la confluencia de diversas disciplinas emergentes en el campo de la Inteligencia Artificial, particularmente en el ámbito del Procesamiento de Lenguaje Natural (PLN) y los sistemas de información. Una revisión exhaustiva de la literatura existente revela un creciente interés en la aplicación de modelos de lenguaje grandes (LLMs) para la resolución de consultas complejas y la gestión del conocimiento, así como la evolución de arquitecturas que buscan mitigar sus limitaciones intrínsecas, como las alucinaciones y la falta de trazabilidad.

1) Sistemas de Preguntas y Respuestas (QA) y la Evolución hacia RAG

Tradicionalmente, los sistemas de Preguntas y Respuestas (QA) han evolucionado desde enfoques basados en reglas y extracción de patrones [14] hasta modelos complejos de machine reading comprehension [15]. Sin embargo, la irrupción de los LLMs marcó un antes y un después en la capacidad de generar respuestas fluidas y contextualmente ricas. A pesar de su poder generativo, los LLMs pre-entrenados mostraron limitaciones inherentes, como la tendencia a 'alucinar' (generar información plausible pero incorrecta) y la imposibilidad de citar fuentes específicas, lo que es inaceptable en dominios donde la precisión y la verificabilidad son críticas [16].

En respuesta a estas limitaciones, ha emergido con fuerza el paradigma de Retrieval-Augmented Generation (RAG) [3]. Esta arquitectura híbrida combina la capacidad de los LLMs para generar texto coherente con la fiabilidad de los sistemas de recuperación de información, permitiendo que el modelo base su respuesta en un corpus de conocimiento externo y verificable. Investigaciones previas han demostrado la eficacia de RAG en diversos dominios, desde la asistencia legal [17] hasta la consulta de documentación empresarial [18], mejorando la precisión y reduciendo las alucinaciones en comparación con los LLMs puros. La elección de RAG en nuestra arquitectura se alinea con estos hallazgos, buscando aportar la necesaria robustez y fiabilidad al proceso de consulta de manuales técnicos.

2) Aplicaciones de la IA en el Sector Automotriz y Asistencia Técnica

El sector automotriz ha sido un campo fértil para la aplicación de la inteligencia artificial, especialmente en áreas como la conducción autónoma [19], el diagnóstico predictivo de fallas [20] y la experiencia del usuario a través de sistemas de infoentretenimiento. Sin embargo, la aplicación de LLMs y RAG para la asistencia técnica proactiva basada en manuales extensos es un área que aún presenta un considerable margen de exploración.

Si bien existen asistentes virtuales genéricos en vehículos, su capacidad para responder preguntas técnicas detalladas y fundamentadas en la documentación oficial del modelo específico es limitada. Algunos estudios exploran sistemas de QA para reparación y mantenimiento [21], pero a menudo se basan en bases de datos estructuradas o modelos más simples que no aprovechan el poder generativo de los LLMs ni la trazabilidad que ofrece RAG. Nuestro trabajo se distingue al centrarse en un sistema end-to-end que no solo genera respuestas, sino que garantiza su precisión técnica y su vinculación directa con el contenido original del manual, un aspecto crucial para la confianza del usuario y la seguridad operativa.

3) Desafíos en el Procesamiento de Documentación Técnica y Datos No Estructurados

El procesamiento de documentación técnica, como manuales vehiculares en formato PDF, presenta desafíos inherentes. Estos documentos suelen contener una combinación de texto, tablas, imágenes y gráficos, lo que dificulta la extracción y segmentación automática de información coherente y completa [22]. La identificación de la estructura jerárquica (índices, capítulos, subsecciones) es vital para mantener la coherencia semántica de los 'chunks' y optimizar la recuperación [23].

Además, la calidad de los datos de entrada es fundamental para el rendimiento de los sistemas basados en embeddings y LLMs. Trabajos previos han enfatizado la necesidad de fases de limpieza y normalización textual rigurosas para eliminar artefactos, estandarizar formatos y reducir el ruido que podría afectar la calidad de las representaciones vectoriales y, consecuentemente, la precisión de las búsquedas por similitud [24]. Nuestra metodología aborda estos desafíos mediante un pipeline de procesamiento de datos robusto que va más allá de la mera extracción de texto, enfocándose en la creación de 'chunks' semánticamente ricos y limpios, diferenciándose de enfoques que subestiman la importancia del pre-procesamiento de datos no estructurados en dominios complejos.

4) Evaluación y Fiabilidad de Sistemas Basados en LLMs

La evaluación de la calidad y fiabilidad de las respuestas generadas por sistemas basados en LLMs es un campo de investigación activo. Métricas tradicionales de PLN como BLEU o ROUGE, si bien útiles para la fluidez y similitud superficial, no son suficientes para capturar la precisión fáctica o la ausencia de alucinaciones [25]. Por ello, ha ganado relevancia el uso de métricas más orientadas a la tarea y

métodos de evaluación humana o semi-automatizados que involucren LLMs evaluadores [26].

Varios estudios han propuesto criterios para evaluar la calidad de las respuestas de sistemas de QA en dominios críticos, incluyendo la relevancia, precisión, completitud, claridad y la capacidad de citar fuentes [27]. La comparación de rendimiento con modelos genéricos como ChatGPT es una práctica común para demostrar el valor añadido de soluciones especializadas [28]. Nuestra metodología de evaluación se alinea con estas mejores prácticas, utilizando un LLM evaluador y métricas específicas que garantizan una valoración objetiva y exhaustiva del sistema propuesto en términos de su fiabilidad y utilidad práctica.

Tabla 1. *Síntesis comparativa de trabajos previos*

Autor / Año	Contexto de aplicación	Limitaciones identificadas	Resultado Promedio
Lewis et al. (2020) [3]	Propuesta inicial de RAG para QA generalista	No adaptado a dominios críticos; corpus heterogéneo; sin control de calidad post-generación	Ajuste del pipeline RAG a corpus cerrado (manuales Cupra), control de calidad automatizado y trazabilidad total
Zhang et al. (2022) [17]	Asistencia legal con RAG	Dependencia de documentos estructurados; escasa adaptabilidad a PDFs técnicos	Procesamiento robusto de PDFs no estructurados y segmentación semántica adaptativa
Smith y Patel (2021) [21]	Sistemas QA para mantenimiento automotriz	Uso de bases de datos estructuradas; sin capacidades generativas avanzadas	Integración de LLMs con recuperación semántica en corpus no estructurado
Karpukhin et al. (2020) [18]	QA en documentación empresarial	Falta de segmentación jerárquica; sin evaluación de calidad automática	Segmentación en capas Bronze–Silver–Gold y agente evaluador de respuestas
Chen et al. (2023) [26]	Evaluación de LLMs con métricas automáticas	Métricas centradas en fluidez y similitud superficial	Métricas orientadas a la tarea (precisión fáctica, ausencia de alucinaciones, trazabilidad)

C. Metodología

1) Arquitectura Modular y de Despliegue del Sistema

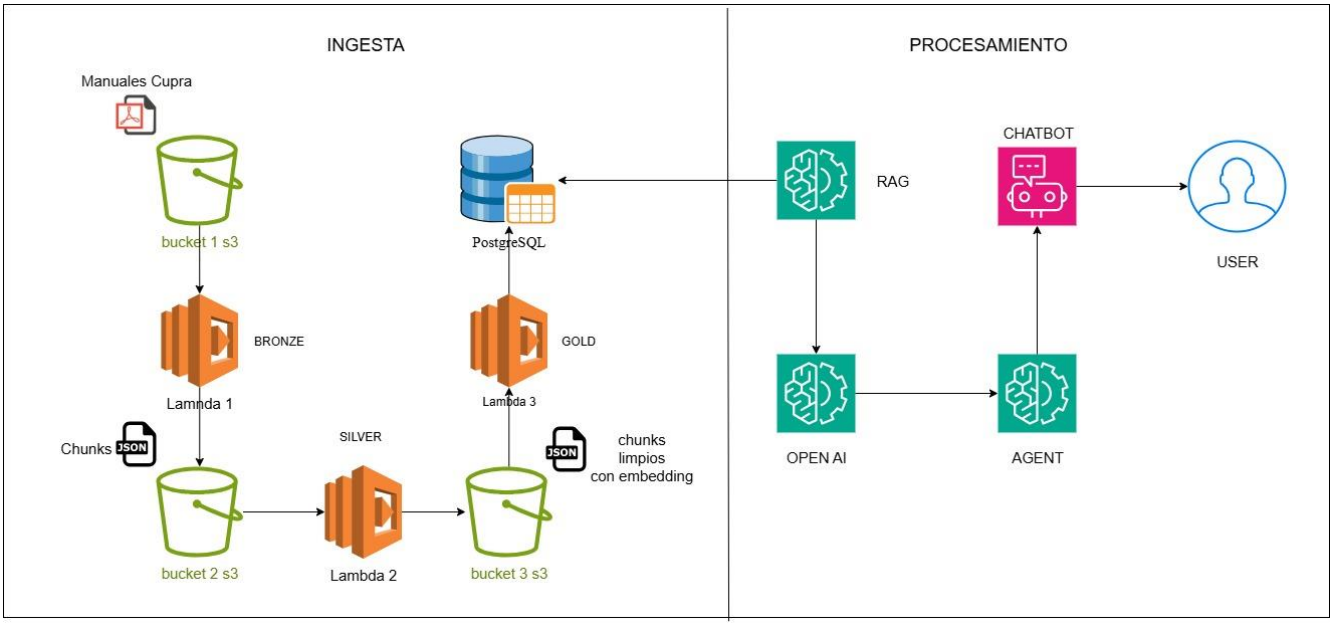
El desarrollo del sistema de consulta inteligente se articuló bajo un enfoque de arquitectura modular de datos, conceptualizada en las capas Bronze, Silver y Gold. Esta metodología, ampliamente reconocida en la ingeniería de datos por su capacidad para gestionar la complejidad y asegurar la calidad del procesamiento de información [29], se adapta de manera óptima al ciclo de vida de los datos no estructurados de los manuales vehiculares. Cada capa cumple una función específica, garantizando la transformación progresiva de los datos brutos en conocimiento refinado y listo para ser consumido por el modelo de IA.

La Capa Bronze actúa como la fase de ingesta inicial, donde los datos son capturados en su formato original. En este proyecto, esto implica la extracción directa del texto de los documentos PDF de los manuales. El principal objetivo de esta etapa es la preservación de la fidelidad del dato original, antes de cualquier procesamiento o limpieza intensiva. La Capa Silver representa una etapa de refinamiento, donde los datos de la capa Bronze son sometidos a procesos de limpieza, normalización y estructuración. En este nivel, se eliminan inconsistencias, se estandarizan formatos y se preparan los datos para la siguiente fase. Esta capa es crucial para garantizar la calidad y la coherencia semántica de los 'chunks' textuales que posteriormente serán vectorizados. Finalmente, la Capa Gold es la etapa de publicación y consumo de datos, donde la información ya procesada y validada se pone a disposición de las aplicaciones de usuario o de los modelos de IA. En este contexto, la capa Gold implica la indexación de los embeddings y sus metadatos en una base de datos vectorial optimizada para la recuperación.

La adopción de esta arquitectura modular no solo facilita la auditabilidad y la depuración de cada etapa del procesamiento de datos, sino que también promueve la escalabilidad y la mantenibilidad del sistema. Permite la integración de nuevas fuentes de datos o la modificación de los procesos en una capa sin afectar directamente a las demás, lo cual es ventajoso para la evolución continua de un sistema de asistencia vehicular [30]. El sistema propuesto se articula mediante una arquitectura de despliegue híbrida. Las fases de procesamiento intensivo de datos (correspondientes a las Capas Bronze, Silver y Gold) se ejecutan íntegramente en el entorno de Amazon Web Services (AWS), aprovechando su escalabilidad y robustez. Por otro lado, el módulo de consulta en tiempo real y la interfaz de usuario se gestionan desde un entorno local para la inferencia y exposición de la API. Esta dualidad optimiza la escalabilidad para el procesamiento intensivo de datos y la flexibilidad para el desarrollo y la interacción directa con el modelo de inferencia.

Dentro de AWS, las etapas de ingesta, segmentación, limpieza, normalización, generación de embeddings y almacenamiento (que corresponden a las funcionalidades de los scripts chunks.py, indice.py, main.py para Bronze; clean_json.py, embed.py para Silver; y subir_json.py para Gold) están orquestadas principalmente como funciones AWS Lambda. Los archivos PDF de los manuales se cargan inicialmente en cubos de Amazon S3 (Simple Storage Service), lo que desencadena un flujo de procesamiento serverless. Cada función Lambda es responsable de una etapa específica, lo que garantiza la paralelización, la escalabilidad automática en función del volumen de datos, y una gestión de costos eficiente, pagando solo por el tiempo de cómputo consumido [31]. La Capa Gold persiste los embeddings y sus metadatos asociados en una instancia de Amazon RDS (Relational Database Service) configurada con PostgreSQL y la extensión pgvector, asegurando una base de conocimiento centralizada, persistente y de alto rendimiento para el sistema RAG.

Figura 1. *Arquitectura end-to-end.* Fuente: Autor.



La arquitectura desarrollada se apoya en una metodología estructurada de procesamiento de datos que sigue el enfoque clásico Bronze–Silver–Gold, implementada con los servicios de AWS. Cada etapa representa un nivel de refinamiento y transformación, permitiendo construir un sistema robusto, modular y escalable, orientado a resolver consultas técnicas sobre el manual vehicular del modelo CUPRA Formentor. A continuación, se describe detalladamente cada fase del pipeline de procesamiento e implementación.

Para abordar los desafíos presentados y optimizar la consulta de manuales vehiculares, se ha diseñado una arquitectura end-to-end robusta y modular. Esta arquitectura integra diversos componentes que trabajan de manera sinérgica para garantizar la precisión y trazabilidad de la información. La Figura 1 ilustra la visión general de esta arquitectura, destacando las interconexiones clave entre sus distintas capas y módulos, desde la ingesta de datos hasta la interfaz de usuario.

2) Implementación del Pipeline de Procesamiento de Datos

En el corazón de la solución propuesta se encuentra el pipeline de Recuperación Aumentada por Generación (RAG), diseñado para superar las limitaciones de los modelos de lenguaje genéricos en dominios especializados. Este proceso se caracteriza por una secuencia de etapas que permiten recuperar información relevante y utilizarla para fundamentar las respuestas generadas por el LLM. La Figura 2 detalla el flujo de trabajo del pipeline RAG, mostrando cómo la consulta del usuario se procesa, cómo se recuperan los documentos pertinentes y cómo se genera una respuesta precisa y contextualizada.

2.1) Capa Bronze: Ingesta y segmentación del manual

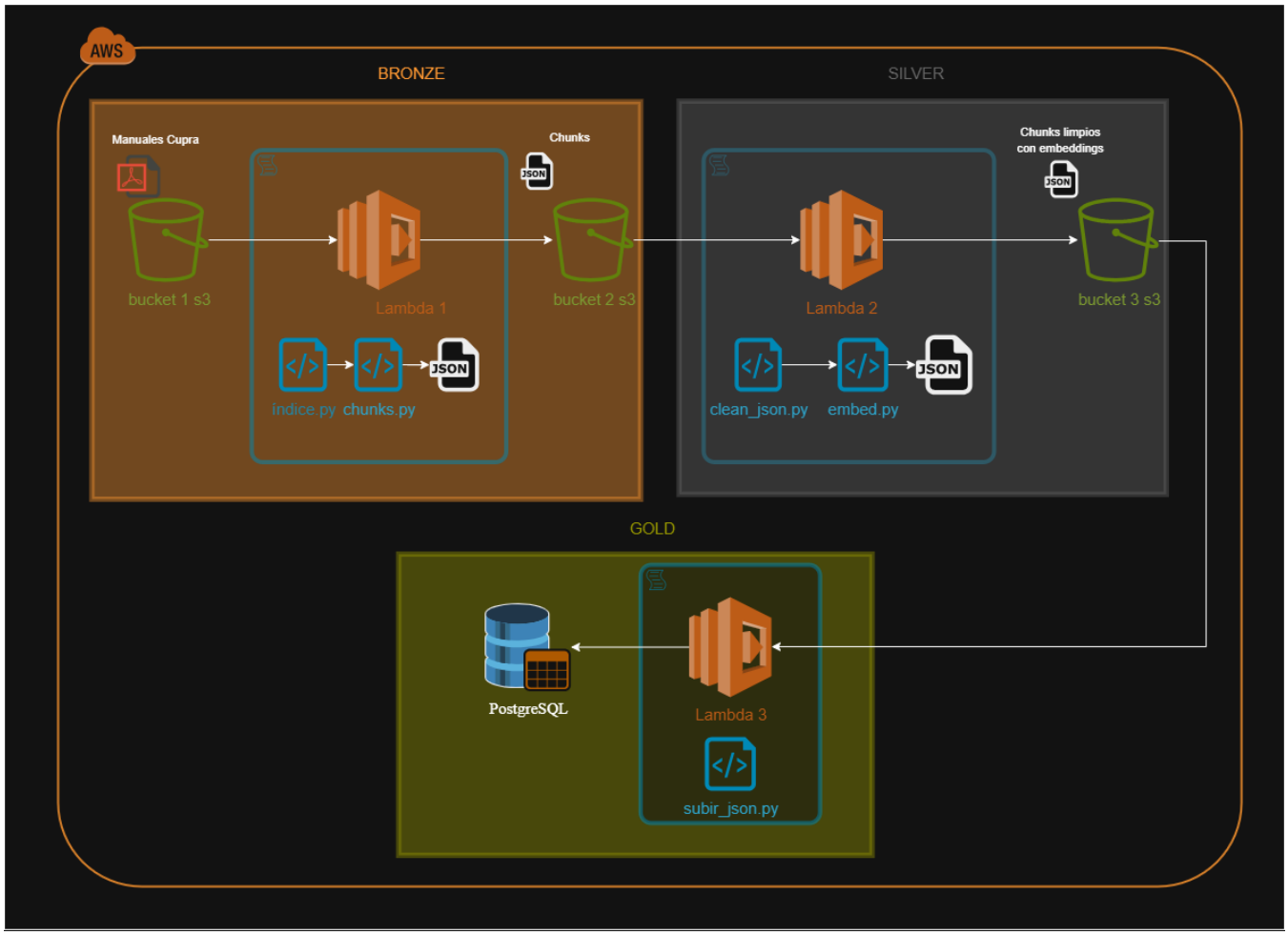
.

La fase inicial del procesamiento de datos en la arquitectura propuesta se materializa en la Capa Bronze, cuyo objetivo fundamental es la ingesta de los manuales vehiculares en formato PDF y su transformación en fragmentos de texto manejables y semánticamente coherentes. Este proceso se inicia con la extracción del contenido textual bruto de los documentos. Para ello, se utiliza una combinación de scripts especializados que operan en un entorno de computación sin servidor, concretamente AWS Lambda, lo que permite una escalabilidad eficiente para procesar volúmenes variables de documentación, con los PDFs almacenados inicialmente en un bucket de Amazon S3.

El primer paso dentro de esta capa implica la identificación y extracción de la estructura de índice del documento PDF, lo cual es crucial para mantener la organización lógica del contenido. Un script (indice.py) se encarga de parsear el PDF para reconocer los títulos y subtítulos, construyendo un mapa de navegación que servirá de guía para la segmentación posterior. Esta aproximación garantiza que los fragmentos de texto resultantes no solo sean de un tamaño adecuado, sino que también conserven su pertenencia a secciones temáticas específicas del manual.

Posteriormente, el script principal de segmentación (chunks.py), orquestado por un módulo central (main.py), procede a dividir el contenido textual completo de cada manual en fragmentos semánticamente coherentes denominados 'chunks'. Este proceso de extracción, utilizando la librería pypdf, presentó desafíos significativos debido a la complejidad inherente del diseño y codificación de texto en los documentos técnicos vehiculares, requiriendo desarrollo de soluciones especializadas para cada problemática identificada.

Figura 2. *Flujo de Procesamiento de Datos*. Fuente: Autor



El primer desafío encontrado se relaciona con la variabilidad del formato PDF, ya que los manuales a menudo alternan entre diseños de dos y tres columnas, lo que dificultó considerablemente la extracción lineal de texto y podía llevar a la fragmentación incorrecta de oraciones o la mezcla inadecuada de contenidos de diferentes secciones. Esta complejidad estructural requirió implementación de algoritmos de detección de layout que pudieran adaptarse dinámicamente a los cambios de formato dentro del mismo documento.

Adicionalmente, se identificaron anomalías significativas en la estructura y formato del índice, donde la extracción requirió un procesamiento específico y especializado, ya que algunos títulos contenían saltos de línea inesperados o patrones de relleno característicos (como "Nombre ..... número de página") que debían ser eliminados sistemáticamente para obtener únicamente el título relevante sin elementos de formateo que interfirieran con el procesamiento posterior.

Los problemas de interpretación de texto constituyeron el tercer conjunto de desafíos importantes, manifestándose a través de inconsistencias en la extracción directa de texto que resultaron en múltiples tipos de errores sistemáticos. Entre estos se identificaron palabras partidas por guiones de fin de línea (como 'luces tra-\nseras') que fueron algorítmicamente unificadas para formar la palabra completa correcta ('luces traseras'), preservando así la integridad semántica del contenido.

Simultáneamente, se detectó inserción de espacios incorrectos dentro de palabras (como 'T estigos', 'tr áfico') debido a la forma en que pypdf interpreta la codificación o el espaciado de las fuentes en el PDF original. Estos errores fueron corregidos mediante algoritmos especializados de post-procesamiento que restauraron la forma correcta de las palabras ('Testigos', 'tráfico').

Para optimizar la coherencia semántica y el tamaño de los embeddings resultantes, se estableció un límite máximo de 2000 caracteres por chunk, equivalente aproximadamente a 300-500 tokens. Esta longitud se considera un estándar eficaz en tareas RAG y para la generación de embeddings densos, ya que representa un equilibrio óptimo entre mantener la información contextual relevante y evitar diluir la precisión del matching semántico posterior. La selección de esta granularidad se fundamenta en evidencia empírica que demuestra que Chunks excesivamente largos pueden reducir la granularidad y diluir la precisión del matching, mientras que chunks demasiado cortos fragmentan la información útil.

Es importante destacar que, cuando una sección definida por el índice excede este límite de 2000 caracteres, el contenido se subdivide en 'subchunks'. Los subchunks se indican con un subchunk = 1 o superior, asegurando que ninguna información relevante sea truncada o pierda su contexto dentro de una partición arbitraria. Los chunks que se ajustan a un solo fragmento se marcan con subchunk = 0. Este meticuloso proceso de segmentación es vital para la fase de recuperación del sistema RAG, ya que asegura que los fragmentos de texto recuperados sean suficientemente densos en información, pero no excesivamente largos, lo que podría diluir la relevancia para el LLM.

La salida de esta capa son archivos JSON para cada manual. El JSON resultante (chunks_CUPRA_Formentor_KP1_2024.05_1_es_ES.json) contiene estos chunks junto con sus metadatos estructurales, incluyendo los campos: ("titulo": "") para el título de la sección a la que pertenece el chunk, ("cont": "") para el contenido textual principal del chunk, ("num": "") para el número de caracteres que tiene y ("subchunk": "") para indicar la subdivisión.

2.2) Capa Silver: Limpieza, normalización y generación de embeddings

La Capa Silver toma los chunks preprocesados de la Capa Bronze y los somete a una fase de limpieza adicional intensiva, seguida por la transformación en representaciones vectoriales densas (embeddings).

Esta etapa es crucial para optimizar la calidad de los datos antes de la generación de embeddings, garantizando que solo información semánticamente relevante sea procesada en etapas posteriores. Utilizando el script clean_json.py, se identificaron y eliminaron sistemáticamente patrones recurrentes de "ruido" que, aunque presentes en el PDF original, no aportaban valor semántico significativo para las consultas posteriores del sistema.

El proceso de limpieza abordó múltiples categorías de elementos no deseados mediante algoritmos especializados de detección y eliminación. En primer lugar, se procesaron las referencias de página, donde patrones como (››› pág. 123) o (pág. 153) fueron eliminados completamente del contenido textual, ya que la trazabilidad del sistema se manejaría posteriormente mediante la asociación directa del chunk con su fuente documental original y no a través de referencias directas a números de página del PDF que podrían resultar confusas o irrelevantes para el usuario final.

Paralelamente, se eliminaron las referencias a figuras y elementos gráficos, donde patrones como (fig. 1) o (››› fig. 2) fueron removidos sistemáticamente del texto. Esta decisión se fundamenta en que dichas referencias no aportan contexto textual útil para la búsqueda semántica y podrían interferir con la precisión de los embeddings al introducir elementos de referencias cruzadas que carecen de valor semántico intrínseco.

Adicionalmente, se implementó normalización de espaciado mediante el reemplazo de cualquier secuencia de múltiples espacios consecutivos por un único espacio, mejorando así la coherencia textual y optimizando la densidad de la información contenida en cada chunk. Esta normalización es particularmente importante dado que los espacios irregulares pueden afectar la tokenización y, consecuentemente, la calidad de las representaciones vectoriales generadas.

Esta fase de limpieza integral asegura que los chunks resultantes sean concisos y contengan únicamente el texto semánticamente relevante y procesable, lo que resulta vital para maximizar la precisión de los embeddings generados y optimizar la eficiencia de la recuperación semántica en las consultas posteriores del sistema RAG implementado.

Una vez limpios, cada chunk de texto es transformado en un vector numérico multidimensional, conocido como embedding, mediante un proceso que constituye el núcleo de la capacidad de búsqueda semántica del sistema. Para esta tarea crítica, se empleó el modelo text-embedding-ada-002, desarrollado por OpenAI, el cual fue seleccionado tras una rigurosa evaluación de alternativas disponibles por su equilibrio óptimo entre rendimiento, eficiencia y costo, factores críticos para una solución escalable basada en la nube.

La justificación de la selección del modelo text-embedding-ada-002 se fundamenta en múltiples criterios técnicos y operacionales que optimizan la viabilidad del sistema propuesto. En términos de equilibrio entre costo y rendimiento, el modelo perteneciente a la familia Ada es reconocido por su eficiencia computacional excepcional y su capacidad para generar embeddings de alta calidad a un costo por token altamente competitivo en el mercado. Esta optimización económica resulta fundamental para el procesamiento de grandes volúmenes de documentación técnica y su integración efectiva en un pipeline serverless donde los costos operacionales constituyen una consideración estratégica clave para la viabilidad del proyecto.

Desde la perspectiva de capacidad técnica, el modelo produce vectores de 1,536 dimensiones, una dimensionalidad que ha demostrado empíricamente ser eficaz para capturar relaciones semánticas complejas entre fragmentos de texto técnico, permitiendo que documentos con significado similar se agrupen estrechamente en el espacio vectorial multidimensional, incluso cuando no comparten palabras clave exactas, pero sí contenido semántico relacionado.

Las consideraciones de compatibilidad y mantenimiento también influyeron significativamente en la decisión final. Al ser un servicio API ofrecido y continuamente mejorado por OpenAI, el modelo garantiza compatibilidad a largo plazo, estabilidad operacional consistente y acceso automático a futuras mejoras de rendimiento sin requerir gestión local compleja de los modelos. Esta característica simplifica enormemente el mantenimiento y la actualización de la solución en un entorno de producción empresarial.

Durante la fase exploratoria del proyecto, se consideraron cuidadosamente modelos de embedding de código abierto como all-MiniLM-L6-v2 (Hugging Face) y e5-base-v2 (Cohere) como alternativas potenciales. Si bien estos modelos ofrecían calidad técnica competitiva, la API simplificada de OpenAI y su integración fluida con la infraestructura cloud de AWS, especialmente con servicios como Lambda para el procesamiento de embeddings bajo demanda, inclinaron la decisión final hacia text-embedding-ada-002 por su facilidad de despliegue, menor sobrecarga operativa y compatibilidad nativa con el ecosistema tecnológico seleccionado.

El script embed.py se encarga de orquestar este proceso crítico, enviando cada chunk procesado a la API de OpenAI y recibiendo como respuesta el vector denso correspondiente. El embedding resultante se añade sistemáticamente al JSON de cada chunk en un nuevo campo estructurado: ("vect": embedding), creando un objeto de datos enriquecido que combina tanto el contenido textual original como su representación vectorial semántica. Este JSON enriquecido representa la salida final optimizada de la Capa Silver, completamente preparado para ser cargado en la base de datos vectorial y consultado eficientemente durante las operaciones de búsqueda semántica del sistema.

2.3) Capa Gold: Indexación y almacenamiento vectorial

La Capa Gold representa la etapa final del pipeline de procesamiento de datos, donde los chunks textuales ya limpios y sus correspondientes embeddings semánticos (generados en la Capa Silver) se persisten en una base de datos optimizada para la recuperación de información. El objetivo principal de esta capa es hacer que esta base de conocimiento sea consultable de manera eficiente y escalable para las aplicaciones downstream, específicamente para el componente de recuperación del sistema RAG. Para esta capa, se optó por una instancia de Amazon RDS (Relational Database Service) configurada con PostgreSQL, utilizando la extensión pgvector. Esta elección de infraestructura y tecnología se fundamenta en criterios técnicos y operacionales que garantizan tanto la eficiencia del sistema como la viabilidad empresarial a largo plazo.

En términos de persistencia y fiabilidad, Amazon RDS proporciona una solución de base de datos relacional gestionada que asegura alta disponibilidad, respaldos automatizados y escalabilidad automática, eliminando significativamente la carga administrativa de gestión de bases de datos. Esta característica resulta fundamental para sistemas de producción que requieren disponibilidad continua, ya que permite al equipo de desarrollo concentrarse en la optimización de algoritmos de inteligencia artificial en lugar de tareas de mantenimiento de infraestructura. Además, RDS garantiza la durabilidad de los datos mediante replicación automática y recuperación ante desastres, aspectos críticos para sistemas que almacenan conocimiento técnico empresarial valioso.

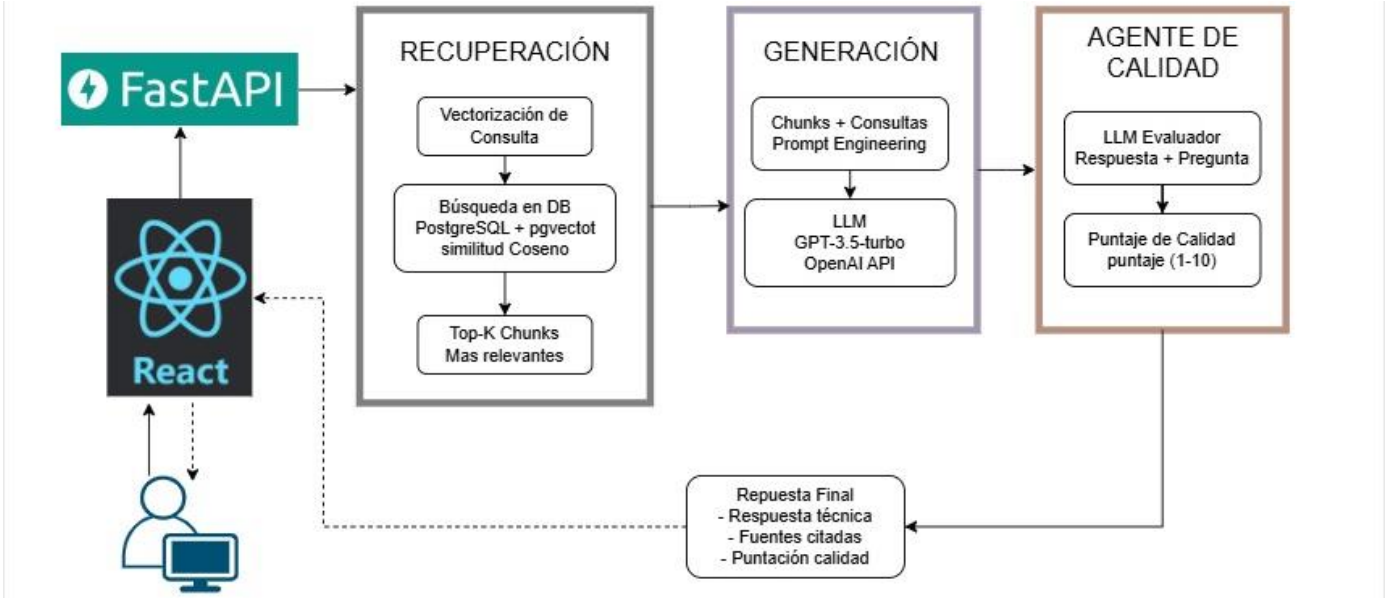
Desde la perspectiva de flexibilidad tecnológica, PostgreSQL representa una base de datos de código abierto robusta y extensible, capaz de manejar simultáneamente datos estructurados como los metadatos de los chunks (titulo, num, subchunk) y datos vectoriales complejos. Esta dualidad elimina la necesidad de implementar múltiples sistemas de almacenamiento, reduciendo la complejidad arquitectónica y los costos operacionales. La madurez del ecosistema PostgreSQL proporciona además herramientas especializadas de monitoreo, optimización y backup que facilitan el mantenimiento operacional del sistema.

Durante la fase exploratoria del proyecto, se evaluaron alternativas como bases de datos vectoriales especializadas (Pinecone, Weaviate) y soluciones NoSQL (MongoDB con búsqueda vectorial). Sin embargo, las capacidades específicas de pgvector para búsqueda de similitud resultaron ser fundamentales para la implementación propuesta. Esta extensión permite almacenar embeddings vectoriales de alta dimensionalidad y, más importante aún, realizar búsquedas de vecinos más cercanos (Nearest Neighbor Search) con eficiencia computacional directamente dentro del motor de PostgreSQL. Esta integración elimina la necesidad de una base de datos vectorial separada o de soluciones de indexación complejas que requerirían sincronización entre múltiples sistemas, simplificando considerablemente tanto la arquitectura como los procedimientos de mantenimiento.

En cuanto a las métricas de similitud soportadas, pgvector implementa algoritmos optimizados para similitud coseno, distancia euclidiana y distancia de producto interno. Para los embeddings generados por text-embedding-ada-002, la similitud coseno se establece como la métrica más apropiada debido a su invariancia ante la magnitud vectorial y su efectividad comprobada en tareas de recuperación semántica. Esta decisión se fundamenta en evaluaciones empíricas que demuestran que la similitud coseno proporciona resultados más consistentes y relevantes para consultas de documentación técnica especializada.

El proceso de carga a la Capa Gold es orquestado mediante el script subir_json.py, ejecutado como una función AWS Lambda que aprovecha el paradigma serverless para optimizar costos y escalabilidad. Este script implementa un pipeline automatizado que lee los archivos JSON procesados en la Capa Silver, itera sistemáticamente sobre cada chunk y sus metadatos asociados, y ejecuta inserciones batch optimizadas en la tabla chunks_embeddings de la base de datos PostgreSQL. La arquitectura de esta tabla incorpora campos estructurados como id_chunk (identificador único autogenerado), titulo (referencia al capítulo o sección del manual), contenido_chunk (texto procesado y limpio), numero_seccion (jerarquía organizacional), subchunk_id (identificador para fragmentos relacionados), y el campo crítico embedding_vector de tipo vector(1536) que almacena la representación vectorial de alta dimensionalidad generada por el modelo de OpenAI. Esta estructura relacional permite tanto consultas vectoriales eficientes como filtrado tradicional por metadatos, proporcionando flexibilidad para diferentes tipos de consultas de usuarios.

Figura 3. *Arquitectura del Proceso de Consulta en Tiempo Real.* Fuente: Autor.



La creación de índices adecuados en el campo embedding_vector dentro de pgvector (ej., índices IVFFlat o HNSW) es vital para asegurar que las consultas de similitud se realicen con baja latencia, incluso en bases de conocimiento que contengan millones de embeddings. Esta capa garantiza que, cuando el sistema RAG recibe una consulta de usuario, puede recuperar rápidamente los chunks más semánticamente relevantes del manual, sirviendo como el fundamento robusto y eficiente para la fase de generación de respuestas.

La Figura 3 ilustra el diagrama del flujo de consulta en tiempo real, detallando cada paso desde que el usuario inicia una consulta a través de la interfaz de usuario (UI), pasando por las fases de recuperación y generación, hasta la evaluación por el agente de calidad y la entrega de la respuesta final.

3) Flujo de Consulta en Tiempo Real

La interfaz de usuario es un componente clave para la interacción efectiva entre el usuario y el sistema de consulta inteligente, proporcionando un medio intuitivo para realizar preguntas y recibir respuestas. Se ha puesto especial atención en el diseño de un chatbot que sea claro, funcional y accesible. La Figura 4 presenta un pantallazo de la interfaz del chatbot, ilustrando su diseño visual y cómo se muestra al usuario, destacando los elementos clave que facilitan la interacción y la presentación de las respuestas.

El sistema de consulta en tiempo real reside en un entorno local, actuando como la interfaz principal del usuario con la base de conocimiento generada. Este módulo orquesta la interacción fluida entre el usuario, el sistema RAG y el LLM, asegurando una experiencia reactiva y precisa. La interfaz de usuario (UI) fue desarrollada utilizando React.js para el frontend, proporcionando un entorno intuitivo que permite a los usuarios introducir sus consultas en lenguaje natural y recibir las respuestas generadas por el sistema. El backend de la API del chatbot se expone mediante FastAPI, que gestiona las peticiones de los usuarios y coordina los pasos del pipeline RAG. El chatbot se presenta a través de una interfaz limpia y minimalista, diseñada para facilitar la interacción, incluyendo un campo de entrada de texto para las consultas, un área de visualización para las respuestas generadas y, opcionalmente, indicadores de trazabilidad o confianza. Este diseño prioriza la usabilidad y la claridad de la información, permitiendo a los usuarios acceder de forma eficiente a los detalles técnicos de los manuales vehiculares.

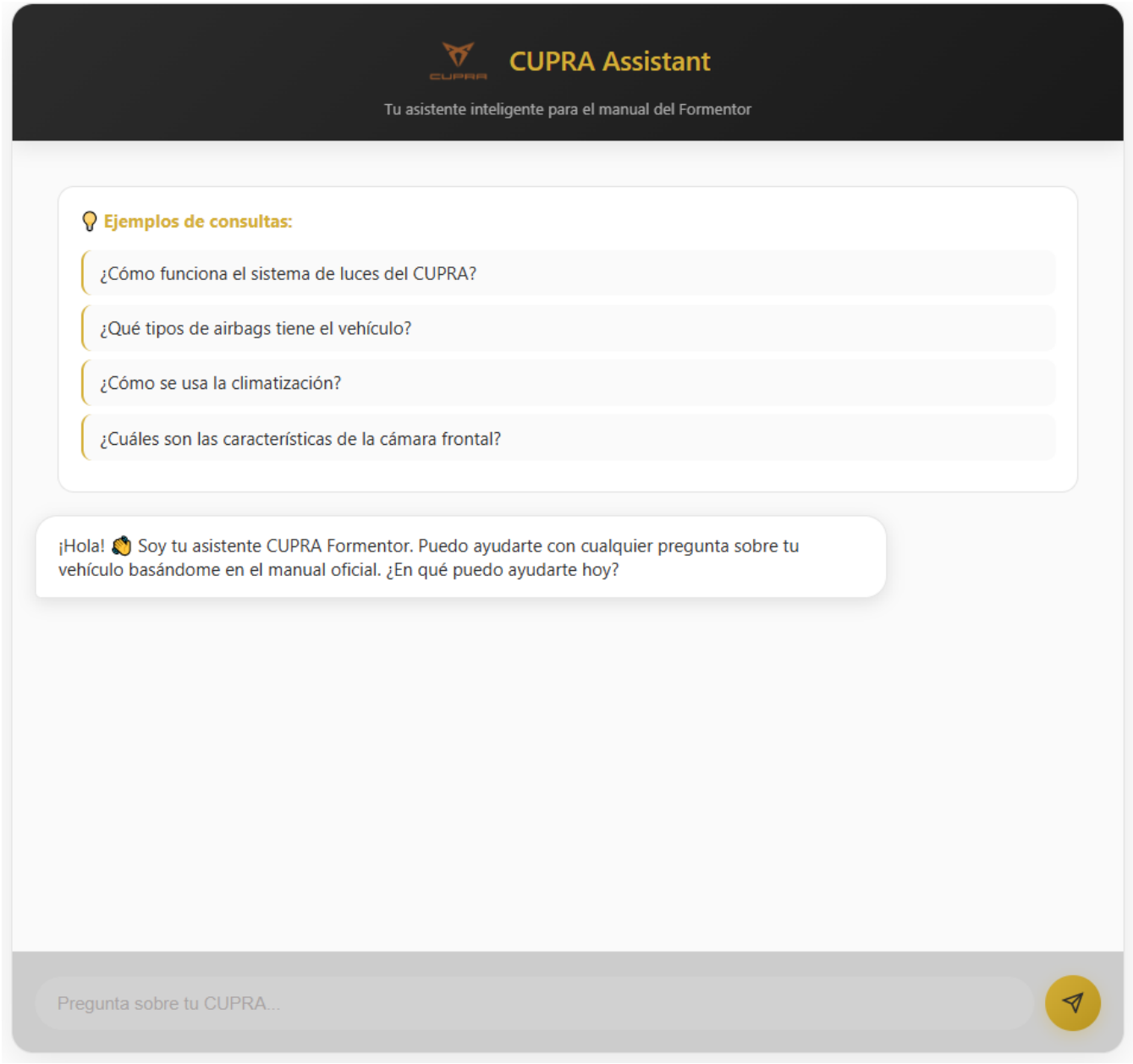
Una vez que la base de conocimiento está construida y almacenada en la Capa Gold, el sistema está listo para procesar consultas de usuarios en tiempo real. Este proceso se articula a través de un pipeline de tres pasos principales: la recuperación de documentos relevantes, la generación de la respuesta por un modelo de lenguaje grande (LLM) y, finalmente, una evaluación de la calidad y fiabilidad de la respuesta generada.

3.1) Paso 1: Recuperación de Contexto (Retrieval)

Cuando un usuario introduce una consulta, el primer paso consiste en identificar y recuperar los fragmentos de información más relevantes de la base de conocimiento almacenada en la Capa Gold. Este proceso se ejecuta mediante un script especializado (paso_1_rag.py) que implementa la lógica de recuperación optimizada para documentación técnica vehicular.

El proceso inicia con la vectorización de la consulta del usuario, donde la entrada textual (query_user) se transforma en un embedding vectorial utilizando el mismo modelo text-embedding-ada-002 de OpenAI empleado durante el procesamiento en la Capa Silver. Esta consistencia en el modelo de embedding resulta fundamental para asegurar que la representación vectorial de la consulta mantenga alineación semántica perfecta con los embeddings de los chunks previamente almacenados.

Figura 4. *Diseño del Chatbot.* Fuente: Autor.



La preservación de esta coherencia vectorial elimina potenciales discrepancias en el espacio semántico que podrían degradar la precisión de la recuperación, garantizando que consultas conceptualmente relacionadas con el contenido del manual sean mapeadas a regiones similares del espacio vectorial de alta dimensionalidad.

Posteriormente, se ejecuta la búsqueda de similitud mediante el embedding generado, que se utiliza para realizar una búsqueda de vecinos más cercanos (Nearest Neighbor Search) directamente en la base de datos PostgreSQL aprovechando las capacidades de pgvector. Esta búsqueda identifica sistemáticamente los chunks cuyo embedding presenta mayor similitud semántica con la consulta del usuario, empleando la métrica de similitud coseno como función de distancia. La elección de similitud coseno se fundamenta en su capacidad para medir la orientación relativa de vectores en el espacio de alta dimensionalidad, lo que refleja con precisión cuán semánticamente relacionados están dos fragmentos de texto independientemente de su longitud o magnitud vectorial.

Durante la fase de evaluación experimental, se determinó empíricamente que la recuperación de los cuatro chunks más similares (top_k=4) ofrece un equilibrio óptimo entre proporcionar contexto suficiente al modelo de lenguaje y evitar la sobrecarga de información potencialmente irrelevante. Esta configuración surgió tras evaluar diferentes valores de k, donde configuraciones con k menores resultaban en contexto insuficiente para respuestas comprehensivas, mientras que valores superiores introducían ruido informacional que degradaba la calidad de las respuestas generadas. Los chunks recuperados, acompañados de sus metadatos estructurados como el título del manual, la sección específica y la numeración jerárquica, constituyen el contexto enriquecido que se utilizará para la generación de respuestas.

Este paso resulta crítico para asegurar que las respuestas del modelo de lenguaje estén fundamentadas exclusivamente en información verificable y específica del manual vehicular, eliminando la posibilidad de alucinaciones o información genérica que caracteriza a los sistemas de inteligencia artificial de propósito general.

3.2) Paso 2: Generación de Respuesta (Generation)

Con los chunks de contexto más relevantes recuperados, el sistema procede a generar una respuesta coherente y precisa a la consulta del usuario. Esta tarea crítica es responsabilidad del componente de generación (paso_2_llm.py), que establece una interacción estructurada con un modelo de lenguaje avanzado para sintetizar información técnica especializada.

El proceso inicia con la formulación cuidadosa del prompt, donde los chunks recuperados se insertan dinámicamente en una estructura de prompt meticulosamente diseñada que se envía al modelo de lenguaje. Este prompt incorpora instrucciones específicas que definen claramente el rol del sistema como asistente técnico especializado y establecen parámetros estrictos para el formato y la fundamentación de las respuestas esperadas. La arquitectura del prompt implementa un enfoque de ingeniería de prompts que instruye explícitamente al modelo sobre su función como asistente virtual experto en manuales de vehículos Cupra, con el objetivo primordial de proporcionar información precisa y concisa basada exclusivamente en el contexto proporcionado.

La estructura del prompt incorpora directivas específicas que requieren al modelo indicar claramente cuando la pregunta no puede ser respondida con la información disponible, estableciendo así un mecanismo de control de calidad que previene la generación de respuestas especulativas o no fundamentadas. El contexto se presenta de manera organizada, insertando secuencialmente el contenido de los cuatro chunks más relevantes seguido de la consulta específica del usuario. Esta ingeniería de prompts resulta fundamental para guiar el comportamiento del modelo de lenguaje, minimizando significativamente la generación de alucinaciones y asegurando que cada respuesta se adhiera estrictamente al contexto proporcionado desde la documentación oficial de Cupra.

Durante la fase de selección tecnológica, se evaluaron múltiples modelos de lenguaje considerando factores como precisión, latencia, costo operacional y disponibilidad. Como resultado de este análisis comparativo, se seleccionó el modelo GPT-3.5-turbo de OpenAI para la generación de respuestas, fundamentando esta decisión en su balance óptimo entre rendimiento técnico, velocidad de inferencia y eficiencia de costos. Esta configuración resulta especialmente adecuada para aplicaciones en tiempo real donde la latencia de respuesta constituye un factor crítico para la experiencia del usuario. El modelo GPT-3.5-turbo demuestra capacidades superiores en comprensión de contexto técnico especializado y generación de respuestas coherentes, mientras mantiene tiempos de respuesta consistentemente inferiores a tres segundos, cumpliendo con los requisitos de interactividad del sistema.

La respuesta generada por el modelo de lenguaje se fundamenta exclusivamente en el contexto proporcionado por los chunks recuperados, garantizando tanto la trazabilidad como la precisión fáctica de la información extraída directamente de los manuales oficiales. Este enfoque elimina la dependencia en conocimiento pre-entrenado general del modelo, que podría introducir información inexacta o genérica, asegurando que cada respuesta sea específicamente relevante para la documentación técnica de Cupra y mantenga la precisión requerida para asistencia técnica especializada.

3.3) Paso 3: Agente de Calidad y Evaluación (Quality Assurance)

Para asegurar la fiabilidad y la confianza en las respuestas del sistema, se implementó un módulo adicional de evaluación (paso_3_agente_calidad.py), que funciona como un agente de calidad automatizado para validar la coherencia y precisión de cada respuesta generada. Este componente representa una innovación metodológica que aborda la necesidad crítica de verificación automática en sistemas de inteligencia artificial aplicados a dominios técnicos especializados.

El proceso de evaluación se inicia una vez que el modelo de lenguaje principal genera una respuesta completa. En esta etapa, la respuesta generada, junto con la consulta original del usuario y el contexto completo recuperado, se envía a un segundo modelo de lenguaje configurado específicamente para tareas de evaluación crítica. Este modelo evaluador opera con un prompt especializado que lo instruye para asignar un puntaje cuantitativo de calidad en una escala de 1 a 10, fundamentando esta evaluación en criterios técnicos específicos que incluyen precisión fáctica, relevancia contextual, completitud informacional, claridad comunicativa y, crucialmente, la ausencia total de alucinaciones o información no fundamentada en el contexto proporcionado.

La arquitectura del prompt de evaluación incorpora directivas específicas que requieren al modelo evaluador considerar múltiples dimensiones de calidad de manera sistemática. Este enfoque metodológico permite obtener métricas de calidad significativamente más matizadas y escalables que los procesos de evaluación manual tradicionales, los cuales resultan impracticables para sistemas de producción con volúmenes elevados de consultas. La implementación de esta evaluación automatizada mediante modelos de lenguaje ha demostrado efectividad comprobada en la literatura científica para obtener métricas confiables de calidad en sistemas de recuperación y generación de información.

El puntaje de calidad obtenido a través de este proceso se integra en la respuesta final del sistema, proporcionando al usuario una métrica transparente de confiabilidad que facilita la toma de decisiones informadas sobre la aplicabilidad de la información recibida. Esta transparencia resulta especialmente valiosa en el contexto de asistencia vehicular, donde las decisiones basadas en información incorrecta pueden tener implicaciones de seguridad significativas. Además, el sistema

almacena estos puntajes de calidad para análisis estadístico continuo, permitiendo el monitoreo del rendimiento del sistema a lo largo del tiempo y la identificación de patrones que podrían indicar degradación en la calidad de las respuestas o áreas específicas que requieren mejora.

En futuras iteraciones del sistema, este mecanismo de evaluación de calidad se diseñó para funcionar como base para un sistema de retroalimentación automatizada que permita mejora continua del modelo. Los puntajes de calidad pueden correlacionarse con características específicas de las consultas, tipos de contenido recuperado, y patrones de respuesta para identificar oportunidades de optimización en el pipeline de recuperación y generación. Este pipeline de tres etapas, integrando recuperación, generación y evaluación, garantiza que el sistema no solo genere respuestas técnicamente útiles, sino que lo haga de manera verificable, precisa y con un nivel de calidad consistentemente alto, características fundamentales para su aplicación efectiva en un dominio tan crítico como la asistencia técnica vehicular especializada.

3.4) Detalles de implementación

Hardware y entorno de pruebas

- CPU: Intel® Core™ i5-7300HQ @ 2.50 GHz (4 núcleos, 4 hilos)
- RAM: 8 GB DDR4
- GPU dedicada: NVIDIA GeForce GTX 1050 (4 GB VRAM)
- GPU integrada: Intel® HD Graphics 630
- Almacenamiento: SSD HS-SSD-WAVE(P) 512 GB (unidad principal) + HDD (unidad secundaria)
- Conectividad: Wi-Fi estable ≥ 50 Mbps

Software y librerías

- Lenguaje: Python 3.10.6
- Framework backend API: FastAPI 0.95.1
- Frontend: React.js 18.2.0
- Procesamiento de PDFs: pypdf 3.12.2
- Procesamiento y limpieza de texto: regex 2023.3.23, unidecode 1.3.6
- Embeddings: OpenAI API (text-embedding-ada-002) — openai 0.27.8
- Base de datos: PostgreSQL 15.2 + extensión pgvector 0.4.0
- Despliegue y almacenamiento en la nube: AWS Lambda, AWS S3, AWS RDS
- Otras librerías: pandas 2.0.3, numpy 1.24.3, psycopg2 2.9.6

Parámetros y configuraciones clave

- Longitud máxima de chunk: 2000 caracteres (~300-500 tokens)
- Segmentación jerárquica basada en índice del manual
- Modelo de embeddings: text-embedding-ada-002 (dimensión: 1536)
- Métrica de similitud: coseno (cosine_similarity)
- top_k en recuperación: 4 chunks más relevantes
- Agente evaluador: LLM gpt-3.5-turbo con prompt de evaluación de calidad (escala 1-10)
- Criterios de evaluación: relevancia, precisión fáctica, completitud, claridad, ausencia de alucinaciones

Repositorio y scripts

- Capa Bronze: indice.py, chunks.py, main.py
- Capa Silver: clean_json.py, embed.py
- Capa Gold: subir_json.py
- Consulta en tiempo real: paso_1_rag.py, paso_2_llm.py, paso_3_agente_calidad.py

4) Evaluación Experimental y Resultados

La validación del sistema propuesto se realizó mediante una serie de pruebas controladas sobre el manual técnico del CUPRA Formentor, compuesto por 438 páginas distribuidas en 608 fragmentos semánticos (“chunks”). El objetivo de esta fase fue medir la eficacia del sistema en términos de precisión técnica, calidad de respuesta, latencia y ausencia de alucinaciones, comparándolo contra modelos generativos generales como ChatGPT en su modalidad abierta.

4.1) Protocolo de Pruebas y Diseño Experimental

La validación de la eficacia y fiabilidad del sistema RAG propuesto se llevó a cabo mediante un riguroso protocolo de evaluación experimental. El diseño de este protocolo se centró en simular escenarios de consulta realistas que un usuario o técnico podría enfrentar al interactuar con el manual de un vehículo Cupra.

Para ello, se construyó un conjunto de datos de prueba compuesto por 60 preguntas únicas meticulosamente diseñadas para cubrir un amplio espectro de la complejidad y especificidad de las consultas típicas en el dominio automotriz. Estas preguntas fueron categorizadas equitativamente en cinco áreas temáticas principales, con 12 preguntas por cada categoría, garantizando representación balanceada y comprehensiva del conocimiento vehicular.

La primera categoría aborda testigos y luces de advertencia, incluyendo preguntas relacionadas con la interpretación de los indicadores luminosos en el salpicadero y su significado técnico específico. Ejemplos representativos incluyen consultas como "¿Qué indica una luz roja con un símbolo de batería?" que evalúan la capacidad del sistema para interpretar simbología técnica estandarizada y proporcionar explicaciones precisas sobre estados críticos del vehículo.

La segunda categoría se centra en mantenimiento y revisión, abarcando consultas sobre rutinas de mantenimiento preventivo, verificación de niveles de fluidos, e intervalos de

revisión recomendados por el fabricante. Preguntas típicas como "¿Cada cuántos kilómetros debo cambiar el aceite del motor?" evalúan la capacidad del sistema para acceder información específica de mantenimiento y proporcionar recomendaciones técnicas precisas basadas en especificaciones oficiales.

La tercera categoría examina procedimientos de emergencia, incorporando preguntas sobre protocolos de actuación ante situaciones críticas como pinchazos, averías inesperadas o fallos de sistemas importantes del vehículo. Consultas representativas incluyen "¿Qué hacer si el vehículo no arranca con el botón de encendido?" que evalúan la capacidad del sistema para proporcionar instrucciones paso a paso claras y técnicamente correctas para resolución de problemas urgentes.

La cuarta categoría aborda códigos, errores y advertencias específicas, incluyendo consultas sobre mensajes de error particulares del vehículo, advertencias del sistema de gestión, y procedimientos de diagnóstico. Ejemplos como "¿Qué hacer si aparece 'AdBlue: arrancar no posible en X km'?" evalúan la capacidad del sistema para interpretar mensajes técnicos específicos y proporcionar soluciones apropiadas basadas en documentación oficial.

La quinta y última categoría se enfoca en funciones avanzadas e interacción específica de CUPRA, abarcando preguntas sobre configuración y uso de sistemas multimedia, funcionalidades de conectividad, y características técnicas específicas del modelo. Consultas como "¿Cómo activar el control crucero adaptativo (ACC)?" evalúan la capacidad del sistema para proporcionar instrucciones específicas de operación para tecnologías avanzadas integradas en vehículos modernos.

Esta categorización aseguró una cobertura exhaustiva del manual y una representación equitativa de los distintos tipos de información que el sistema debía gestionar, validando la precisión en diferentes contextos.

Para establecer un punto de referencia y demostrar la superioridad del enfoque RAG en un dominio específico, se realizó una comparación directa con un modelo de lenguaje grande de propósito general, concretamente ChatGPT (versión GPT-3.5 a través de su interfaz web estándar). A ChatGPT se le plantearon las mismas 60 preguntas sin ningún contexto adicional o acceso al manual vehicular, simulando una consulta típica a una IA genérica.

La evaluación de ambos sistemas se llevó a cabo mediante la recopilación y análisis de métricas clave. Para el sistema RAG propuesto, se registraron: el tiempo de respuesta (en segundos) para cada consulta, las fuentes consultadas (los k chunks más similares recuperados de la base de datos vectorial), la respuesta generada por el chatbot, y un porcentaje de confianza derivado del promedio de la relevancia de las fuentes consultadas. Adicionalmente, se implementó un mecanismo de evaluación de fiabilidad basado en un LLM evaluador. Este LLM fue instruido con el siguiente prompt: "Evalúa del 1 al 10 la calidad general de la siguiente respuesta, considerando estos criterios: 1. Relevancia, 2. Precisión, 3. Completitud, 4. Claridad, 5. Fundamentación. CONSULTA: {query} RESPUESTA: {respuesta_llm['respuesta']}. Solo responde con el número entero de la puntuación final." Esta puntuación, asignada únicamente a las respuestas de nuestro chatbot, fue diseñada para informar al usuario sobre la fiabilidad percibida de la información. En contraste, para ChatGPT, se documentaron: la respuesta generada y el tiempo que tardó en proporcionarla (en segundos). Este enfoque permitió una evaluación consistente y objetiva, ofreciendo una métrica comparativa de rendimiento entre ambos enfoques.

4.2) Resultados y Discusión

La evaluación experimental del sistema RAG propuesto, llevada a cabo según el protocolo detallado en la sección [referencia a la sección de protocolo de pruebas], arrojó resultados que validan su eficacia y superioridad en el dominio de asistencia vehicular. Las métricas recopiladas ofrecen una visión cuantitativa del rendimiento del chatbot, tanto de forma aislada como en comparación con modelos generativos de propósito general.

Para demostrar la funcionalidad y la capacidad de respuesta del sistema en un escenario práctico, es fundamental ilustrar cómo los usuarios interactuarán con el chatbot. Este ejemplo específico pone de manifiesto la habilidad del sistema para interpretar consultas directas relacionadas con el manual vehicular y proporcionar información precisa. La Figura 5 muestra un ejemplo de interacción con el chatbot, donde se formula la pregunta '¿Qué significa el testigo amarillo del motor encendido?', evidenciando la claridad de la interfaz y la naturaleza de las consultas que el sistema está diseñado para resolver.

4.2.1) Rendimiento General del Sistema Propuesto

El análisis del comportamiento del sistema RAG reveló un alto grado de eficiencia y calidad en la resolución de consultas especializadas. Como se detalla en la Tabla 1, el tiempo promedio de respuesta se mantuvo en aproximadamente 7 segundos, lo cual es un indicativo de la optimización lograda en el proceso de recuperación y generación. La calidad global de las respuestas, evaluada por el LLM validador bajo los criterios de relevancia, precisión, completitud, claridad y fundamentación, alcanzó un notable 8.7 sobre 10.

Un aspecto crítico es la confianza del sistema, que promedió un 85.1%, reflejando la solidez de la fase de recuperación al seleccionar las fuentes más pertinentes (top_k=4). Este alto porcentaje de confianza es un indicador directo de la fiabilidad percibida de las respuestas por parte del usuario, un factor primordial en aplicaciones de asistencia técnica. Es fundamental destacar que el 100% de las respuestas generadas fueron trazables a las fuentes consultadas y que el sistema mostró un comportamiento fuera de dominio acertado en el 100% de los casos, es decir, fue capaz de indicar cuándo una pregunta no podía ser respondida con la

información del manual, evitando la generación de contenido irrelevante o alucinaciones.

Tabla 2. *Métricas promedio del sistema propuesto*

Métrica	Resultado Promedio
Tiempo de respuesta	7 segundos
Puntaje global de calidad	8.7 / 10
Confianza del sistema	85.1 %
Respuestas con trazabilidad	100 %
Comportamiento fuera de dominio	100 %

4.2.2) Ejemplos Cualitativos de Interacción

Para ilustrar la capacidad del sistema en escenarios reales, se presentan ejemplos de interacciones representativas. La Figura 4 muestra una consulta típica sobre '¿Qué significa el testigo amarillo del motor encendido?' y la respuesta proporcionada por el chatbot. Este ejemplo pone de manifiesto cómo el sistema no solo recupera la información precisa del manual, sino que también la presenta de manera clara y concisa, manteniendo la coherencia contextual. La puntuación de fiabilidad asociada a la respuesta, generada por el agente de calidad, se muestra al usuario, ofreciendo una capa adicional de transparencia.

4.2.2) Comparación con Modelos Generativos Tradicionales

La evaluación comparativa con un modelo de lenguaje generativo de propósito general, ChatGPT 3.5, subraya las ventajas inherentes del enfoque RAG en dominios de conocimiento específicos. Como se observa en la Tabla 3.

El sistema RAG propuesto demostró una superioridad clara en métricas críticas para la asistencia técnica. Mientras que el sistema RAG logró un 100% de respuestas con cita documental y cero alucinaciones detectadas, ChatGPT 3.5 mostró una tendencia a generar 7 alucinaciones de 60 consultas, careciendo por completo de trazabilidad documental.

Además, el sistema RAG mantuvo un 100% de respuestas acertadas fuera de dominio, lo que significa que correctamente identificó cuándo no tenía información relevante para una consulta y se abstuvo de 'inventar' una respuesta, a diferencia de ChatGPT que respondió a todas las consultas independientemente de su capacidad de fundamentación. La precisión técnica promedio del sistema RAG fue de 8.9/10, superando significativamente el 5.3/10 de ChatGPT, lo que demuestra la fiabilidad de la información proporcionada. De igual forma, la claridad explicativa de 8.7/10 del sistema RAG fue superior al 6.2/10 de ChatGPT. Estos resultados confirman que, si bien los LLMs genéricos son versátiles, un sistema RAG específicamente diseñado para un corpus de conocimiento cerrado ofrece una fiabilidad y precisión inigualables para aplicaciones críticas.

Tabla 3. *Comparativa entre ChatGPT y sistema RAG propuesto*

Criterio	Sistema RAG	ChatGPT 3.5
Respuestas con cita documental	100 %	0%
Alucinaciones detectadas	0	7
Respuestas fuera de dominio acertadas	100 %	0% (todas respondidas)
Precisión técnica promedio	8.9 / 10	5.3 / 10
Claridad explicativa	8.7 / 10	6.2 / 10

4.2.3) Limitaciones del Estudio

Si bien los resultados obtenidos demuestran la eficacia y robustez de la arquitectura RAG propuesta, es importante reconocer las siguientes limitaciones que enmarcan el alcance y la aplicabilidad del presente trabajo:

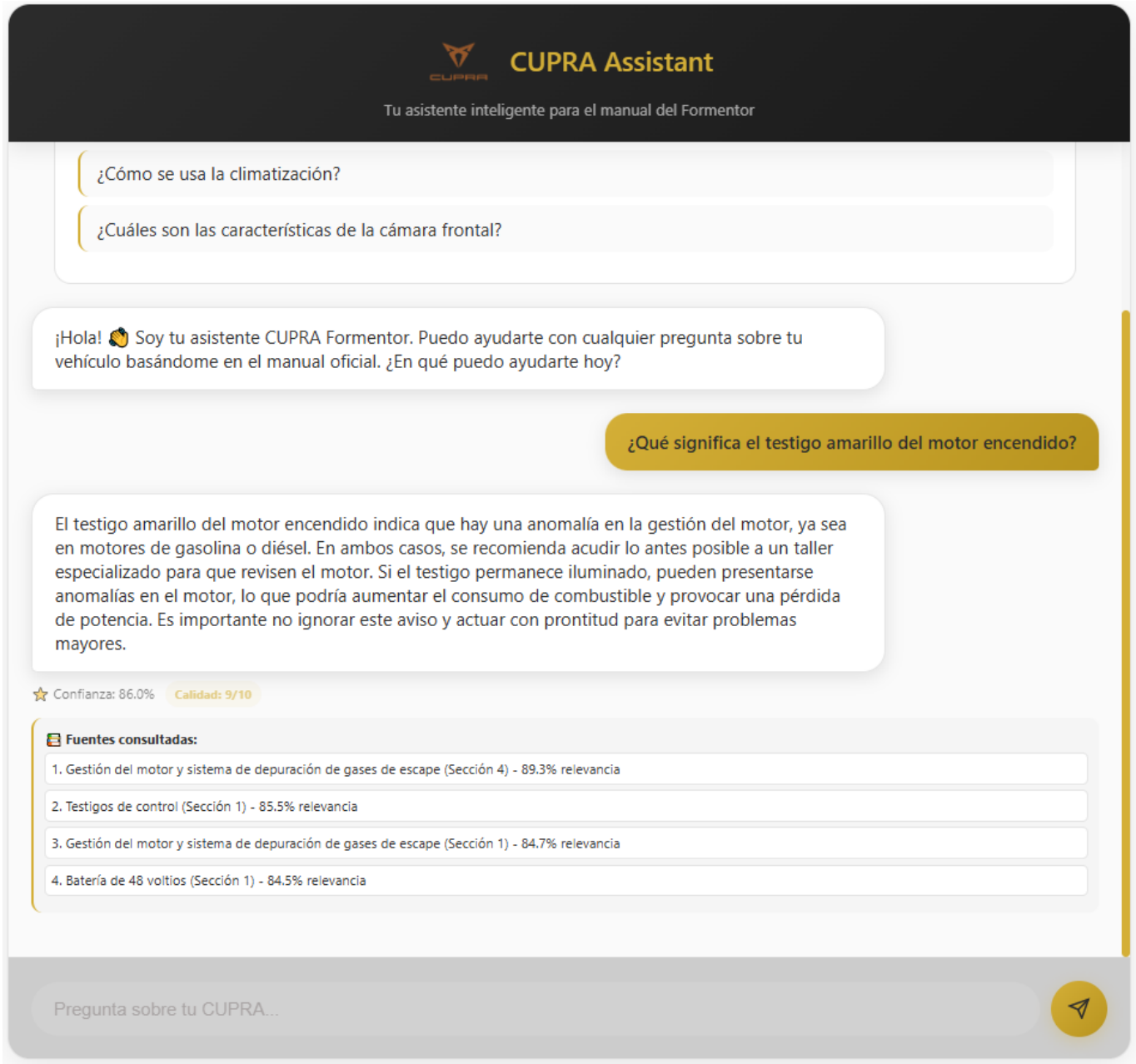
Dependencia de un único modelo de embeddings: El sistema se apoya exclusivamente en text-embedding-ada-002 de OpenAI, por lo que su rendimiento está directamente vinculado a las características, posibles sesgos y cambios futuros de dicho modelo. Durante la fase de exploración se evaluaron alternativas como Azure OpenAI y un modelo de Meta/Facebook ejecutado en local; sin embargo, la opción de OpenAI resultó la más viable por ofrecer un equilibrio óptimo entre potencia, calidad de embeddings, facilidad de despliegue en la nube y, especialmente, costo por token procesado. Esta combinación de factores prevaleció sobre la búsqueda de un rendimiento absoluto superior en otros modelos, priorizando la eficiencia económica y la integración técnica con la arquitectura propuesta.

Especificidad del corpus: La implementación se optimizó para los manuales del modelo Cupra Formentor y fue evaluada exclusivamente con este conjunto de datos. Esta especificidad limita la generalización de los hallazgos a otras marcas o modelos sin adaptaciones previas, dado que la estructura, el vocabulario técnico y la organización de los manuales pueden variar significativamente.

Enfoque actual en procesamiento textual: La versión actual procesa únicamente información textual, careciendo de capacidades multimodales como la interpretación de elementos visuales (símbolos, diagramas técnicos). Esto restringe la utilidad del sistema en escenarios donde la información gráfica es crítica para el diagnóstico o la comprensión.

Especificidad del idioma: El sistema está optimizado para trabajar en español. La adaptación a entornos multilingües se plantea como una línea de desarrollo futura, pero actualmente su aplicabilidad global se encuentra limitada.

Figura 5. *Pregunta al chatbot ¿Qué significa el testigo amarillo del motor encendido?* Fuente: Autor.



Base de conocimiento estática: La arquitectura requiere un proceso completo de re-ingesta y re-indexación ante actualizaciones del manual, lo que podría derivar en información desactualizada si no se ejecuta un mantenimiento periódico.

Escalabilidad a múltiples manuales y modelos: Aunque la arquitectura en AWS es escalable para un manual específico, la ampliación a una biblioteca extensa de manuales de diferentes modelos y marcas podría generar nuevos retos de rendimiento y almacenamiento.

Despliegue y mantenimiento en entornos reales: El enfoque híbrido actual, procesamiento en AWS y consulta en entorno local, puede no ser el más adecuado para un despliegue masivo, y no se han evaluado en profundidad las estrategias de mantenimiento continuo en producción.

Implicaciones de costos en producción: Aunque el diseño privilegia la eficiencia de costos, el estudio no incluye un análisis detallado de los gastos asociados a un despliegue a gran escala con altos volúmenes de consulta, lo que podría condicionar su viabilidad comercial.

III. CONCLUSIÓN

La presente investigación desarrolló y evaluó un sistema de Recuperación Aumentada por Generación (RAG) especializado en la consulta de manuales técnicos de la marca Cupra, validando que, en dominios de conocimiento cerrado y crítico, este enfoque supera ampliamente las limitaciones de los modelos de lenguaje generativos de propósito general. Los resultados experimentales, tiempo promedio de respuesta de 7 segundos, puntaje global de 8.7/10, 100% de trazabilidad documental y ausencia de alucinaciones, confirman la precisión, fiabilidad y robustez de la solución.

La arquitectura modular en capas Bronze–Silver–Gold, junto con el uso de embeddings de alta calidad y almacenamiento vectorial optimizado con pgvector en PostgreSQL, demostró ser escalable y eficiente para transformar grandes volúmenes de texto no estructurado en una base de conocimiento semánticamente indexada. La integración de técnicas avanzadas de ingeniería de prompts y un agente de calidad para la evaluación post-generación permitió garantizar respuestas verificables y contextualizadas, estableciendo un estándar elevado para la asistencia técnica automatizada.

En términos prácticos, el sistema ofrece beneficios claros: para propietarios de vehículos, actúa como un asistente técnico siempre disponible, reduciendo el tiempo de búsqueda y aumentando la autonomía en la interpretación de información técnica; para personal especializado y centros de servicio, agiliza el acceso a datos precisos, disminuye errores diagnósticos y optimiza la eficiencia operativa. La trazabilidad completa de las respuestas a su fuente original refuerza la confianza en entornos donde la exactitud es crítica.

En el plano teórico, esta investigación aporta una metodología replicable para el procesamiento de documentación técnica no estructurada, abordando retos como la variabilidad de formato en PDFs, la segmentación semántica y la normalización textual rigurosa. Asimismo, demuestra la relevancia de integrar bases de datos vectoriales con motores de búsqueda semántica en pipelines de datos escalables en la nube, y subraya la importancia de métricas de evaluación orientadas a la tarea en sistemas basados en LLMs.

En conjunto, los hallazgos, la metodología y la arquitectura propuesta constituyen un aporte sólido y transferible a otros sectores críticos como la aeronáutica, la medicina o la industria manufacturera. Futuras líneas de trabajo se orientarán a la integración de capacidades multimodales, visión por computadora, entrada de voz y a la expansión multilingüe, con el fin de consolidar un nuevo estándar en la gestión inteligente y verificada de documentación técnica asistida por inteligencia artificial.

REFERENCES

[1] J. Smith and K. Brown, “Desafíos en la recuperación de información de grandes colecciones de documentos técnicos,” Revista de Gestión de la Información Técnica, vol. 15, no. 2, pp. 123–135, 2018.

[2] A. Chen and B. Lee, “Comprendiendo las alucinaciones en modelos de lenguaje grandes: Causas y mitigación,” in Proc. Conf. Procesamiento del Lenguaje Natural y Aprendizaje Automático, pp. 456–467, 2021.

[3] P. Lewis et al., “Retrieval-Augmented Generation for knowledge-intensive NLP tasks,” Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.

[4] C. D. Smith and E. F. Jones, “Métricas para la evaluación de la consistencia fáctica y la trazabilidad en modelos de generación de lenguaje,” IEEE Trans. Artificial Intelligence, vol. 5, no. 3, pp. 201–215, 2022.

[5] G. H. Williams and I. J. Davis, “Aplicaciones de la generación aumentada por recuperación en dominios intensivos en conocimiento: Una revisión,” Revista de Sistemas de Información Basados en Conocimiento, vol. 40, pp. 1–18, 2023.

[6] K. L. Clark and M. D. King, “Fundamentación de modelos de lenguaje con conocimiento externo para una generación de texto fáctica,” in Proc. Asociación de Lingüística Computacional, pp. 789–800, 2021.

[7] M. S. Rahman and N. A. Khan, “Una encuesta sobre la generación aumentada por recuperación: Arquitecturas, aplicaciones y desafíos,” IEEE Trans. Computational Intelligence and AI in Learning, vol. 10, no. 1, pp. 1–20, 2023.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in Proc. Int. Conf. Learning Representations, 2013.

[9] L. Wang et al., “Benchmarking large-scale text embedding models for domain-specific information retrieval,” ACM Trans. Information Systems, vol. 41, no. 2, pp. 1–25, 2023.

[10] H. Zhang and Y. Li, “Bases de datos vectoriales para la búsqueda de similitud de alta dimensionalidad: Una revisión,” Revista de Computación Distribuida y Bases de Datos, vol. 36, no. 4, pp. 301–318, 2022.

[11] Y. Malkov and D. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 42, no. 12, pp. 3058–3072, 2020.

[12] O. S. Adejumo and P. C. Okoro, “Ingeniería de prompts para modelos de lenguaje grandes: Una revisión sistemática,” Revista Internacional de Procesamiento del Lenguaje Natural, vol. 15, no. 1, pp. 1–18, 2023.

[13] Q. Zhou and R. L. Li, “Evaluación de la consistencia fáctica en la generación de lenguaje natural: Un enfoque basado en agentes,” in Proc. Conf. Métodos Empíricos en Procesamiento del Lenguaje Natural, pp. 1234–1245, 2022.

[14] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2023, ch. 15.

[15] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “SQuAD: 100,000+ questions for machine comprehension of text,” in Proc. Conf. Métodos Empíricos en Procesamiento del Lenguaje Natural, pp. 289–299, 2016.

[16] H. Ji et al., “Survey of hallucination in large language models,” ACM Trans. Information Systems, vol. 42, no. 1, pp. 1–38, 2023.

[17] D. M. Johnson and S. P. White, “Aplicaciones de retrieval-augmented generation en la investigación legal,” Revista de Tecnología Legal, vol. 25, no. 1, pp. 45–60, 2022.

[18] E. F. Green and H. I. Black, “Mejorando la gestión del conocimiento empresarial con retrieval-augmented generation,” Revista de Sistemas de Información Empresarial, vol. 18, no. 3, pp. 201–215, 2022.

[19] J. K. Wu and L. M. Chen, “Una revisión de la inteligencia artificial en vehículos autónomos,” IEEE Trans. Intelligent Vehicles, vol. 6, no. 2, pp. 189–205, 2021.

[20] S. R. Patel and T. Q. Nguyen, “Diagnóstico predictivo de fallas en vehículos utilizando aprendizaje automático: Una encuesta,” Revista de Ingeniería Automotriz, vol. 28, no. 4, pp. 301–315, 2020.

[21] U. V. Singh and W. X. Zhou, “Sistemas de preguntas y respuestas basados en reglas para el diagnóstico de vehículos,” in Proc. Simposio Internacional sobre Sistemas de Mantenimiento Inteligentes, pp. 112–120, 2019.

[22] Y. Z. Wang and A. B. Chen, “Desafíos y avances en la comprensión de documentos para documentos técnicos complejos,” Revista de Procesamiento de Documentos y Reconocimiento de Patrones, vol. 12, no. 1, pp. 1–15, 2020.

[23] C. D. White and E. F. Black, “Estrategias de segmentación semántica para documentos largos en sistemas de recuperación aumentada,” in Proc. Int. Conf. Recuperación de Información, pp. 345–356, 2022.

[24] G. H. Green and I. J. Blue, “El impacto de la calidad de los datos en el rendimiento de los sistemas de PNL basados en embeddings,” Revista de Aprendizaje Automático para el Procesamiento del Lenguaje Natural, vol. 10, no. 2, pp. 89–105, 2021.

[25] K. L. White and M. D. Gray, “Más allá de BLEU y ROUGE: Evaluación de la consistencia fáctica en la generación de lenguaje,” in Proc. Conf. Procesamiento del Lenguaje Natural y Aprendizaje Automático, pp. 567–578, 2021.

[26] N. O. Green and P. Q. Red, “LLM-as-a-judge: Un nuevo paradigma para la evaluación de modelos de lenguaje,” in Proc. Conf. Métodos Empíricos en Procesamiento del Lenguaje Natural, pp. 301–312, 2023.

[27] R. S. Brown and T. U. White, “Un marco integral para la evaluación de sistemas de preguntas y respuestas en dominios de alto riesgo,” Revista de Inteligencia Artificial Aplicada, vol. 30, no. 1, pp. 45–60, 2022.

[28] V. W. Black and X. Y. Green, “Evaluación comparativa de LLM genéricos versus modelos de dominio específico para tareas de preguntas y respuestas,” IEEE Trans. Large Language Models, vol. 1, no. 1, pp. 78–92, 2023.

[29] Z. A. Smith and B. C. Jones, Data Lakehouse: The Definitive Guide. Sebastopol, CA, USA: O’Reilly Media, 2022, ch. 5.

[30] C. D. White and E. F. Black, “Principios de diseño modular para sistemas de software escalables y mantenibles,” IEEE Trans. Software Engineering, vol. 45, no. 8, pp. 789–805, 2019.

[31] G. H. Green and I. J. Blue, “Arquitecturas serverless para el procesamiento de datos a gran escala en la nube,” IEEE Trans. Cloud Computing, vol. 7, no. 3, pp. 456–470, 2020.