



CincoE QueryMate: Acceso inteligente a registros de reporteria mediante lenguaje natural

Julian Andrés Quintero Restrepo

Informe del semestre de industria para optar al título de Ingeniero de
Telecomunicaciones otorgado por la Universidad de Antioquia

Asesor:
Erwin Alexánder Leal Piedrahita: PhD

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Electrónica y
Telecomunicaciones
Medellín 2024.

Cita	(Quintero Restrepo, 2024)
Referencia	Quintero Restrepo. J. A. (2024). <i>CincoE QueryMate: Acceso inteligente a registros de reporteria mediante lenguaje natural</i> . [Semestre de industria]. Universidad de Antioquia, Medellín, Colombia.
Estilo APA 7 (2020)	



Centro de Documentación Ingeniería (CENDOI)

Repositorio Institucional: <http://bibliotecadigital.udea.edu.co>

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda Céspedes.

Decano/Director: Julio Saldarriaga.

Jefe departamento: Eduard Rodriguez.

El contenido de esta obra corresponde al derecho de expresión de los autores y no compromete el pensamiento institucional de la Universidad de Antioquia ni desata su responsabilidad frente a terceros. Los autores asumen la responsabilidad por los derechos de autor y conexos.

Contenido

1.	Resumen.....	5
2.	Introducción.....	5
3.	Objetivos.....	6
3.1	Objetivo General	6
3.2	Objetivos Específicos:	6
4.	Marco Teórico.....	7
4.1	LLM	8
4.1.1	Cómo Operan y Funcionan los LLMs	8
4.1.2	Tipos de LLM.....	8
4.1.3	Implementaciones Actuales	9
4.1.6	Modos de Operación	10
4.1.7	Métodos de Procesamiento.....	10
4.2	Prompt.....	10
4.3	Documentos o fuente de información.....	11
4.4	LangChain.....	11
4.4.1	Agentes	11
4.4.2	Tools	12
4.4.3	Token	12
4.4.4	AgentExecutor	12
4.4.5	Chains.....	12
4.5	Sistema POS	13
5	Solución propuesta	13
5.1	Identificación de requisitos.	13
5.1.1	Puntos débiles en relación con las necesidades de los clientes.	13
5.1.2	Recomendaciones y acciones de mejora propuestas.....	14
5.2	Aspectos clave en la selección de los LLM	15
5.2.1	Características técnicas que deben cumplir los LLM	15
5.2.2	Información sobre los diferentes LLM disponibles en el mercado que se adaptan a las necesidades del proyecto.....	17
5.2.2.1	OpenAI	17

5.2.2.2	Google Gemini	19
5.2.2.3	Azure OpenAI.....	21
5.2.3	Elección de LLM para pruebas piloto.....	22
5.3	Pruebas piloto usando LangSmith para los LLM Google Gemini y GPT 4o	23
5.3.1	Prueba de Consulta Básica.....	23
5.3.2	Prueba de Consulta Avanzada	27
5.3.3	Tabla que permite la selección del LLM más adecuado para la solución a implementar.	31
5.4	Integración del LLM en la web principal	32
5.4.1	Vista lógica	33
5.4.2	Vista de desarrollo	34
5.4.2.1	<i>Frontend</i>	35
5.4.2.2	<i>Backend</i>	36
5.4.3	Vista de procesos	42
5.4.4	Vista física	43
5.4.5	Vista de escenarios	43
5.5	Evaluación del sistema integrado	44
6	Conclusiones.....	47
7	Poster	49
8	Referencias bibliográficas.....	50

1. Resumen

El proyecto "CincoE QueryMate" nace de la necesidad de mejorar la obtención de información específica en la empresa CincoE S.A.S., donde los clientes encontraban desafiante localizar datos relevantes dentro de informes extensos. Este proyecto se enfocó en desarrollar un sistema de interacción para propietarios de negocios utilizando inteligencia artificial y modelos de lenguaje grandes (LLM – Large Language Model). El sistema se implementó a través del análisis del historial de ventas en plataformas web corporativas, mejorando significativamente la experiencia del cliente al proporcionar respuestas personalizadas. Los LLM permiten a los clientes formular preguntas precisas sobre los informes, como datos de ventas específicos o tendencias recientes, facilitando una interacción más natural y eficiente con la información.

Se destaca que la incorporación de LLM mejora notablemente la experiencia del cliente, permitiendo un acceso más fácil y rápido a la información relevante dentro de los informes. Las evaluaciones realizadas en un ambiente de producción mostraron resultados positivos, con el sistema proporcionando respuestas precisas y relevantes en un tiempo prudente.

2. Introducción

CincoE es una empresa pionera en innovación que ofrece soluciones tecnológicas integrales para optimizar la gestión de puntos de venta (POS – Point Of Sale). A través de su plataforma web y móvil, proporciona herramientas intuitivas y poderosas que se ajustan a las necesidades de una amplia gama de negocios, desde pequeñas tiendas hasta grandes cadenas comerciales. Además, ofrece soluciones personalizadas y escalables, respaldadas por un equipo de profesionales altamente cualificados y tecnología de última generación.

Actualmente, obtener informes específicos puede resultar abrumador para los clientes debido a la extensión de los documentos y la dificultad para encontrar la información relevante. En este contexto, los grandes modelos de lenguaje (LLM – Large Language Model) se destacan como herramientas innovadoras que facilitan una interacción más natural e intuitiva con la información. Con el uso de LLM, los clientes pueden formular preguntas específicas sobre los informes,

como: ¿cuál es el artículo más vendido?, ¿cuándo se registraron las mayores ventas?, ¿cuántas unidades de un producto específico se vendieron? o ¿cuál es la tendencia reciente de ventas? Los LLM pueden procesar y analizar el contenido del informe para responder a estas preguntas de manera precisa y concisa, eliminando la necesidad de revisar largas páginas de texto. Esto permite a los clientes obtener la información necesaria de manera rápida y eficiente, facilitando la toma de decisiones informadas y estratégicas, y ahorrando tiempo y esfuerzo en la búsqueda de datos.

Este trabajo se enfocó en desarrollar un sistema de interacción para propietarios de negocios, utilizando inteligencia artificial y modelos LLM. El sistema se implementó a través del análisis del historial de ventas en plataformas web corporativas, mejorando significativamente la experiencia del cliente al proporcionar respuestas personalizadas.

3. Objetivos

3.1 Objetivo General

Desarrollar e implementar un módulo de software para el acceso Inteligente a Registros de Reportería mediante Lenguaje Natural, CincoE QueryMate.

3.2 Objetivos Específicos:

- 1) Identificar los requisitos de los clientes a partir de la información que se desea consultar en la plataforma.
- 2) Determinar los LLM más adecuados para las necesidades de la empresa en base a por su precio, rendimiento y facilidad de uso.
- 3) Integrar el modelo en el API principal de la web corporativa de forma personalizada para los informes relacionados con los Puntos de Venta (POS) y otras fuentes de información.
- 4) Evaluar el sistema integrado con diferentes tipos de usuarios y escenarios para verificar que el cumplimiento de los objetivos de la implementación del LLM ha sido exitosa.

4. Marco Teórico

En toda solución de software que incorpore habilitadores tecnológicos como los grandes modelos de lenguaje (LLM – Large Language Model), se identifican los componentes de su ecosistema y se presenta una breve descripción de cada uno (ver Figura 1).

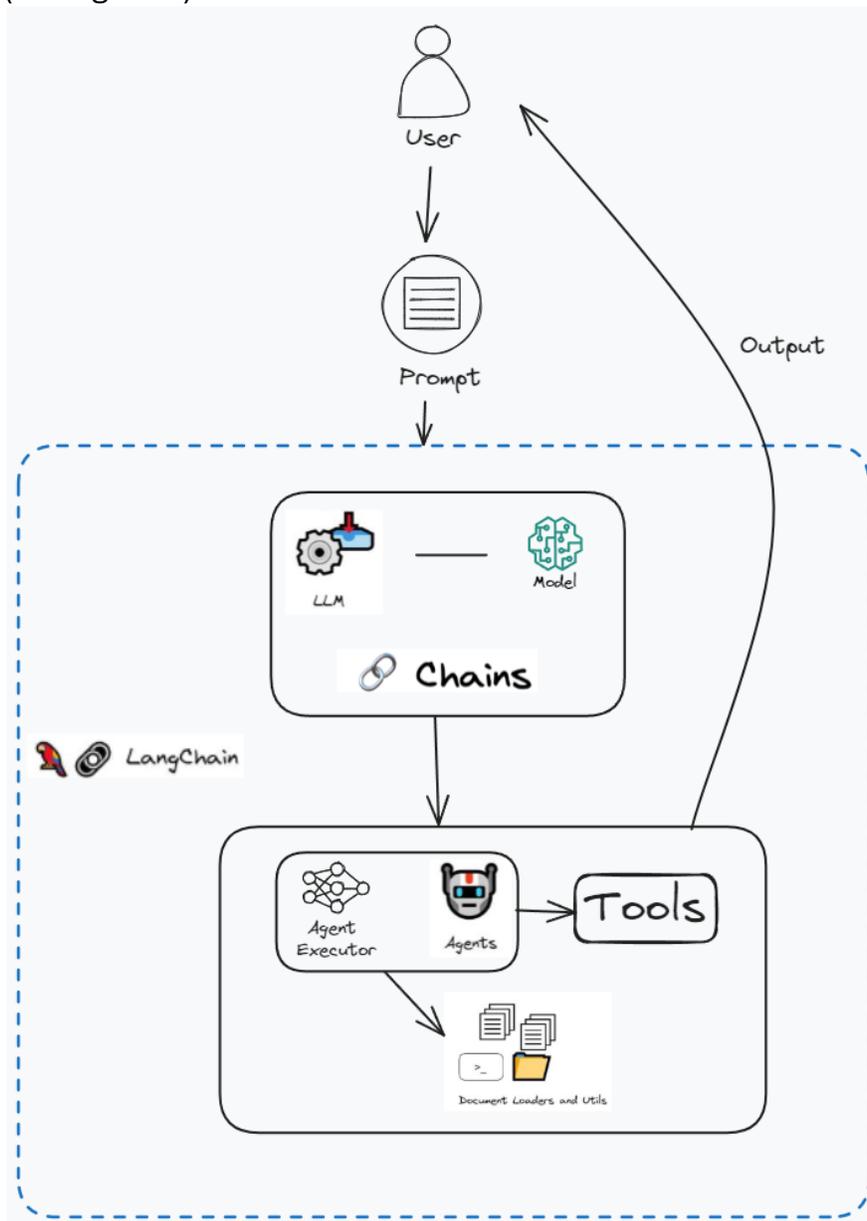


Figura 1. Elementos que integran una solución LLM

4.1 LLM

Los modelos de LLM son una rama de la inteligencia artificial (IA). Estos modelos se entrenan con grandes cantidades de datos de texto y son capaces de realizar una variedad de tareas, como la traducción automática, la generación de texto, la escritura creativa y la respuesta a preguntas [1]. En el proyecto, los LLM son la base con la cual se lograrán alcanzar los objetivos propuestos, ya que se encargan de encontrar los resultados esperados por el usuario, utilizando la respectiva API asignada.

4.1.1 Cómo Operan y Funcionan los LLMs

Entrenamiento: Los LLMs se entrenan en grandes conjuntos de datos textuales utilizando técnicas de aprendizaje profundo. Durante el entrenamiento, el modelo aprende patrones, estructuras gramaticales y el significado contextual de las palabras. El proceso de entrenamiento implica el ajuste de millones o incluso miles de millones de parámetros para minimizar el error en la predicción del siguiente token en una secuencia de texto [1].

Arquitectura: La mayoría de los LLMs modernos están basados en la arquitectura Transformer, introducida por Vaswani et al. en 2017. Los Transformers utilizan mecanismos de atención para procesar las palabras en una secuencia de texto de manera más eficiente y capturar dependencias a largo plazo. Esta arquitectura permite paralelizar el entrenamiento y generar texto de manera más rápida y precisa [1].

Funcionamiento: Un LLM genera texto prediciendo la siguiente palabra o token en una secuencia, basándose en el contexto previo. Esto se logra mediante el uso de técnicas de inferencia, donde el modelo utiliza sus parámetros entrenados para hacer predicciones probabilísticas sobre la siguiente palabra en una oración [1].

4.1.2 Tipos de LLM

Existen varios tipos de modelos de lenguaje, cada uno diseñado para tareas específicas, entre ellos se encuentran:

- **Modelos Generativos:** Estos modelos están diseñados para generar texto de manera autónoma. Ejemplos incluyen GPT-3 y GPT-4 de OpenAI, que pueden escribir ensayos, responder preguntas y mantener conversaciones [2].
- **Modelos de Comprensión:** Estos modelos se centran en comprender y analizar el texto en lugar de generarlo, es un ejemplo de un modelo de comprensión de lenguaje, utilizado para tareas como análisis de sentimiento y clasificación de texto. En el proyecto utilizamos modelos generativos debido a que se adaptan mejor a nuestras necesidades [2].

4.1.3 Implementaciones Actuales

Los modelos de lenguaje han sido implementados por diversas empresas líderes en tecnología, cada una aportando sus propias innovaciones y enfoques para mejorar la capacidad y la utilidad de estos modelos [3]. A continuación, se presentan algunas de las implementaciones más destacadas:

- OpenAI ha desarrollado algunos de los modelos de lenguaje más avanzados, como GPT (Generative Pre-trained Transformer), incluyendo versiones como GPT-3 y GPT-4. Estos modelos son conocidos por su capacidad para generar texto coherente y contextual, y se utilizan en una variedad de aplicaciones, desde chatbots hasta generación de contenido [3].
- GEMINI, un modelo desarrollado por Google, se enfoca en mejorar la interacción con los usuarios y proporcionar respuestas más precisas. Este modelo se utiliza en diversos productos de Google para mejorar la experiencia del usuario, demostrando la capacidad de los modelos de lenguaje para adaptarse a diferentes necesidades y aplicaciones [3].
- Azure AI de Microsoft también ofrece modelos de lenguaje avanzados como parte de su suite de servicios de inteligencia artificial. Azure AI proporciona herramientas y APIs para integrar modelos de lenguaje en aplicaciones empresariales, facilitando tareas como la generación de texto, el análisis de sentimientos y la comprensión del lenguaje natural. Estos modelos se destacan por su escalabilidad y capacidad de integración con otros servicios en la nube de Microsoft [3].

4.1.6 Modos de Operación

- **Síncrono:** En el modo síncrono, el modelo procesa y responde a las solicitudes en tiempo real. Este enfoque es adecuado para aplicaciones donde la velocidad de respuesta es crucial, como chatbots y asistentes virtuales [4].
- **Asíncrono:** En el modo asíncrono, las solicitudes se procesan en segundo plano, permitiendo que múltiples tareas se manejen simultáneamente. Este modo es útil para tareas que no requieren una respuesta inmediata, como el análisis de grandes volúmenes de texto [4].

4.1.7 Métodos de Procesamiento

- **Streaming:** El procesamiento en streaming implica el análisis continuo de datos en tiempo real. En el contexto de los LLMs, esto puede significar la generación de texto en tiempo real a medida que se reciben nuevos datos [4].
- **Lotes:** El procesamiento por lotes implica el manejo de grandes cantidades de datos en bloques. Esto es común en el entrenamiento de modelos y en tareas de procesamiento donde se requiere analizar o generar texto en grandes volúmenes [4].

4.2 Prompt

Un prompt es una instrucción o conjunto de instrucciones proporcionadas a un modelo de lenguaje para guiar su generación de texto. Sirve como punto de partida para la respuesta del modelo, determinando el contexto y el enfoque del contenido generado. Los prompts pueden variar en longitud y complejidad, desde simples frases hasta descripciones detalladas, y son fundamentales para obtener resultados específicos y relevantes del modelo [5]. En el proyecto, el prompt se captura vía web a través de un componente que permite la escritura del usuario.

4.3 Documentos o fuente de información

En el contexto de LangChain, los documentos o fuentes de información juegan un papel vital en la construcción de aplicaciones de lenguaje natural robustas y precisas. LangChain permite la integración y gestión eficiente de diversas fuentes de información, facilitando el acceso a datos relevantes y actualizados para alimentar modelos de lenguaje [6]. En el proyecto nuestra fuente información es la base de datos de CincoE, la cual se alimenta de los sistemas POS en las tiendas.

4.4 LangChain

Es un marco de código abierto diseñado para facilitar el desarrollo de aplicaciones con modelos de inteligencia artificial. LangChain permite la conexión de grandes modelos de lenguaje (LLMs) con diversas fuentes de información externa, mejorando así la precisión y relevancia de las respuestas. Además, proporciona herramientas para orquestar diferentes llamadas y procesos, creando automatizaciones complejas y eficientes que integran múltiples servicios y recursos en un flujo de trabajo cohesivo [6]. En el proyecto LangChain, se encarga de orquestar el flujo de información entre el LLM y el backend mediante la API de OpenAI.

4.4.1 Agentes

Dentro de LangChain, los agentes son componentes inteligentes que pueden tomar decisiones y ejecutar acciones de manera autónoma. Estos agentes utilizan los LLMs para procesar información y decidir la mejor acción a tomar en función del contexto y los datos disponibles. Pueden realizar tareas como responder preguntas, generar informes, ejecutar comandos o interactuar con otras herramientas y servicios. Los agentes son altamente configurables, lo que permite a los desarrolladores personalizar su comportamiento y capacidades según las necesidades específicas de la aplicación [6]. En el proyecto, solo se cuenta con un agente encargado de seleccionar la herramienta adecuada según sea necesario.

4.4.2 Tools

LangChain también incluye una variedad de tools, o herramientas, que son módulos específicos diseñados para llevar a cabo tareas especializadas. Estas tools pueden incluir acceso a bases de datos, realización de cálculos, interacción con APIs externas, procesamiento de lenguaje natural, y más. Integrar estas herramientas dentro del framework de LangChain permite a los desarrolladores construir aplicaciones más robustas y versátiles, optimizando la forma en que los LLMs interactúan con diferentes tipos de datos y servicios. Esto no solo mejora la precisión de las respuestas generadas por los modelos, sino que también amplía las capacidades funcionales de la aplicación en general. [6]. En el proyecto, se disponen de tres herramientas, cada una con objetivos específicos: una herramienta encargada de llamar a la API de CincoE para obtener información, otra dedicada a crear gráficos si el usuario lo necesita, y una tercera que genera un archivo de Excel que el usuario puede descargar.

4.4.3 Token

Un token es una unidad de texto procesada por modelos LLM, como GPT-4 o Gemini. En este contexto, un token puede ser una palabra completa, una parte de una palabra o incluso un carácter, dependiendo del idioma y la configuración específica del modelo [6].

4.4.4 AgentExecutor

AgentExecutor es un componente crucial en el ecosistema de LangChain que permite la ejecución coordinada de agentes. Estos agentes son capaces de realizar tareas específicas basadas en instrucciones definidas por el usuario o el sistema [6]. En el proyecto, el AgentExecutor maneja la orquestación y supervisión del agente creado para la selección de las herramientas, asegurando que las acciones se ejecuten de manera eficiente y correcta.

4.4.5 Chains

Chains es un concepto fundamental en LangChain que permite la creación de flujos de trabajo secuenciales y modulares utilizando LLMs y otras herramientas. Una "chain" (cadena) en LangChain es una serie de pasos

interconectados, donde la salida de un paso sirve como entrada para el siguiente [6]. En el proyecto, se hace uso de las cadenas cuando se llaman múltiples herramientas, como convertir a Excel y graficar la pregunta del usuario.

4.5 Sistema POS

Los sistemas POS son soluciones digitales utilizadas para gestionar transacciones comerciales y operaciones relacionadas en tiendas, restaurantes y otros negocios. Estos sistemas suelen utilizar SQLite, una base de datos liviana y autocontenida [7]. En CincoE, cada tienda tiene un sistema integrado de POS donde el personal registra las ventas diarias.

5 Solución propuesta

A continuación, las siguientes subsecciones detallan las actividades realizadas para alcanzar los objetivos propuestos.

5.1 Identificación de requisitos.

5.1.1 Puntos débiles en relación con las necesidades de los clientes.

Inicialmente, se realizaron múltiples reuniones con los equipos de CincoE y clientes de los sistemas POS, enfocadas en identificar y evaluar los desafíos que enfrentan nuestros clientes al obtener informes específicos.

Actualmente, muchos de nuestros clientes encuentran abrumador manejar los informes extensos que proporcionamos. Los desafíos específicos incluyen:

- **Extensión de los Documentos:** Los informes son a menudo demasiado largos, dificultando la localización de información relevante de manera rápida y eficiente.
- **Dificultad en la Navegación:** La estructura y presentación de los datos no siempre facilitan una rápida interpretación y toma de decisiones.

- **Fragmentación de la Información:** La información relevante a menudo está dispersa en diferentes secciones del informe, obligando a los clientes a buscar en múltiples lugares para obtener una visión completa.
- **Inconsistencia en el Formato:** Variaciones en el formato y estilo de los informes pueden generar confusión, haciendo más difícil para los clientes comparar datos de diferentes períodos o categorías.
- **Falta de Personalización:** Los informes estándar no siempre se adaptan a las necesidades específicas de cada cliente, lo que puede resultar en datos irrelevantes o innecesarios para ciertos usuarios.
- **Retraso en la Disponibilidad de Datos:** Algunos clientes han reportado demoras en la generación y distribución de informes, lo que impacta negativamente en su capacidad para tomar decisiones oportunas.

5.1.2 Recomendaciones y acciones de mejora propuestas

Para abordar estos problemas, se integró LLM como una herramienta innovadora para mejorar la interacción con la información en los informes. Los LLM pueden transformar la manera en que los clientes acceden y utilizan los datos, ofreciendo múltiples beneficios siguiendo estos pasos:

A) Implementación de una Interfaz de Usuario Amigable: Desarrollar una interfaz intuitiva donde los clientes puedan interactuar directamente con el LLM. La interfaz debe permitir la entrada de preguntas en lenguaje natural y proporcionar respuestas claras y precisas.

Ejemplos de preguntas:

"¿Cuándo se registraron las mayores ventas?"

"¿Cuántas unidades de un producto específico se vendieron?"

El LLM procesará las consultas y proporcionará respuestas precisas y concisas en tiempo real.

B) Proporcionar recursos educativos y soporte técnico para ayudar a los clientes a utilizar eficazmente las capacidades del LLM.

C) Integración con Sistemas Existentes: Asegurar que el LLM pueda integrarse de manera fluida con los sistemas y plataformas ya en uso por los clientes.

5.2 Aspectos clave en la selección de los LLM

5.2.1 Características técnicas que deben cumplir los LLM

A continuación, se detallan las características que garantizarán que los LLM puedan manejar de manera efectiva las consultas precisas de los clientes y proporcionar respuestas rápidas y relevantes basadas en informes extensos.

- **Compatibilidad Asincrónica (Async Support)**

Es crucial que los LLM implementen soporte asincrónico, lo que permite a otras funciones asincrónicas en la aplicación progresar mientras se ejecuta el LLM. Esto se logra moviendo las llamadas al LLM a un hilo en segundo plano utilizando el ejecutor del grupo de hilos predeterminado de `asyncio`. De esta manera, los usuarios pueden realizar múltiples consultas simultáneamente sin tiempos de espera prolongados, mejorando la eficiencia y la experiencia del usuario.

- **Soporte de Transmisión (Streaming Support)**

Para una interacción que disminuya al máximo los tiempos de espera para los usuarios, Los LLM deben proporcionar un sistema de entrega continua, permitiendo devolver un solo valor de forma progresiva, ya sea de manera síncrona o asíncrona, dependiendo de cómo se esté manejando la transmisión. Este valor único es el resultado final proporcionado por el proveedor del LLM. Aunque esto no proporciona transmisión token por token, asegura que el código que espera un iterador de tokens pueda funcionar para cualquier integración de LLM. Esta característica es esencial para manejar grandes volúmenes de datos de manera más dinámica y eficiente.

- **Soporte por Lotes (Batch Support)**

Los LLM deben ser capaces de manejar solicitudes en paralelo utilizando un ejecutor del grupo de hilos (en el caso de lotes síncronos) o `asyncio.gather` (en el caso de lotes asincrónicos).

- **Implementaciones Nativas para Soporte Asincrónico, de Transmisión y por Lotes**

Aunque las implementaciones por defecto son útiles, los proveedores de LLM pueden ofrecer implementaciones nativas más eficientes para soporte asincrónico, de transmisión o por lotes. Estas implementaciones nativas son particularmente importantes para mejorar la eficiencia y el rendimiento del sistema, asegurando respuestas rápidas y precisas a las consultas de los clientes.

- **Integración con LangChain**

LangChain proporciona un marco robusto para implementar y gestionar LLMs, y es fundamental que los LLM sean completamente compatibles con las características de LangChain. Esto incluye la interfaz `Runnable`, que proporciona implementaciones predeterminadas para métodos como `ainvoke`, `batch`, `abatch`, `stream` y `astream`. La integración fluida con LangChain permitirá aprovechar al máximo las capacidades de los LLM y proporcionar una solución eficiente y escalable para los clientes de CincoE S.A.S.

Estas funcionalidades son fundamentales para garantizar que los LLM puedan manejar eficazmente las consultas específicas de los clientes, proporcionando una experiencia de usuario mejorada y facilitando la interacción con la información contenida en los informes largos.

A continuación, en la tabla 1 se ilustran los modelos LLM que cumplen con los objetivos del proyecto, basados en sus características de soporte nativo para `async`, `streaming` y `batch`, según la documentación de LangChain:

Modelo LLM	Async	Streaming	Batch
OpenAI	Sí	Sí	Sí
Azure OpenAI	Sí	Sí	Sí
Google Gemini	Sí	Sí	Sí

Tabla 1. Modelos LLM que cumplen con las características necesarias del proyecto

5.2.2 Información sobre los diferentes LLM disponibles en el mercado que se adaptan a las necesidades del proyecto.

5.2.2.1 OpenAI

OpenAI es conocido por sus avanzados modelos de lenguaje, como GPT-4, que pueden generar, resumir y traducir texto, así como asistir en programación y otras tareas complejas. La API de OpenAI es flexible y puede integrarse en aplicaciones para procesar preguntas específicas sobre documentos largos, permitiendo a los clientes de CincoE S.A.S. obtener respuestas precisas sobre datos de ventas o tendencias recientes. OpenAI también ofrece productos especializados como Codex, diseñado para asistir en la escritura de código.

Precios

OpenAI ofrece diferentes niveles de suscripción, que van desde un uso gratuito con limitaciones hasta planes de pago por uso y suscripciones mensuales. Los costos típicos para el uso de GPT-4 son aproximadamente USD 0.03 por cada 1,000 tokens para el modelo estándar y USD 0.06 por cada 1,000 tokens para el modelo más avanzado. Las suscripciones mensuales para empresas, como el plan “Business”, pueden costar alrededor de USD 500 por mes, ofreciendo tarifas reducidas y mayor capacidad de procesamiento, lo que puede ser ventajoso para CincoE S.A.S. Además, existen opciones de suscripción “Enterprise” que proporcionan acceso personalizado y soporte técnico dedicado, con costos variables según las necesidades específicas. [10]

Casos de Éxito

OpenAI ha sido adoptado por diversas industrias para mejorar la eficiencia y la creatividad. Por ejemplo:

- **Soporte al Cliente:** Empresas utilizan ChatGPT para crear chatbots que brindan asistencia personalizada y en tiempo real, mejorando la satisfacción del cliente y reduciendo tiempos de espera. Un caso notable es el de una empresa de telecomunicaciones que redujo los tiempos de respuesta en un 50% al implementar ChatGPT en su sistema de atención al cliente. [10]
- **E-commerce:** Plataformas de comercio electrónico integran ChatGPT para ofrecer recomendaciones personalizadas de productos, respuestas a consultas y generación de descripciones creativas de productos. Un minorista en línea reportó un aumento del 20% en las conversiones de ventas después de implementar recomendaciones personalizadas mediante GPT-4. [10]
- **Educación y Finanzas:** GPT-4 se ha utilizado para resumir informes financieros extensos, permitiendo a los clientes realizar consultas específicas y recibir respuestas inmediatas. Por ejemplo, una firma de consultoría financiera utiliza GPT-4 para analizar y resumir reportes trimestrales, ahorrando hasta 30 horas de trabajo manual por mes, directamente aplicable a las necesidades de CincoE S.A.S. [10]

Opiniones de Usuarios

Las opiniones sobre OpenAI son generalmente positivas:

Feedback positivo: Usuarios valoran la capacidad de generar texto coherente y relevante, así como la facilidad de integración y la flexibilidad en la personalización. Un desarrollador comentó: "La integración de GPT-4 en nuestra plataforma ha transformado la manera en que interactuamos con nuestros datos, haciéndolo más accesible y eficiente." [10]

Áreas de mejora: Algunos usuarios mencionan la necesidad de mejores manejos de contextos complejos y la reducción de respuestas incorrectas en situaciones ambiguas. Un usuario empresarial señaló: "Aunque GPT-4 es increíblemente potente, a veces puede generar respuestas que no se alinean perfectamente con el contexto, lo que requiere una supervisión adicional." Además, los costos pueden ser un factor para considerar para usos intensivos, especialmente en aplicaciones que requieren un procesamiento constante y en gran volumen [10].

Soporte Langchain

LangChain ofrece soporte para la integración fluida con APIs a través del paquete langchain-openai. Este paquete facilita la conexión y comunicación con servicios como ChatGPT.

5.2.2.2 Google Gemini

Google Gemini es una plataforma de inteligencia artificial que combina procesamiento de lenguaje natural y aprendizaje automático. Permite a los usuarios interactuar de manera efectiva con grandes volúmenes de datos, proporcionando respuestas precisas y contextualizadas a preguntas específicas. Google Gemini se integra fácilmente con Google Cloud y aprovecha la infraestructura de Google para ofrecer soluciones escalables y eficientes. Su capacidad multimodal permite combinar diferentes tipos de datos en una sola interfaz, lo que facilita tareas complejas y la obtención de insights profundos.

Precios

Google Gemini ofrece planes de precios basados en el uso y la capacidad del modelo seleccionado (Ultra, Pro, Nano). Los precios específicos no están siempre detallados públicamente, pero generalmente comienzan alrededor de USD 0.02 por cada 1,000 caracteres procesados. Los planes "Pro" y "Ultra" ofrecen mayor capacidad y rendimiento, con costos que pueden variar desde USD 1,000 a USD 10,000 mensuales dependiendo del uso y los requerimientos específicos del proyecto. Google ofrece planes flexibles que permiten adaptar los costos según las necesidades del proyecto, y créditos promocionales para nuevos usuarios, facilitando la prueba y adopción inicial [11].

Casos de Éxito

Google Gemini ha sido implementado en una variedad de sectores, incluyendo:

- **Atención Médica:** Desarrollado para asistir en el diagnóstico y la información médica, Gemini ayuda a profesionales de la salud a acceder rápidamente a datos relevantes y tomar decisiones informadas. Un hospital utilizó Gemini para analizar registros médicos y mejorar la precisión en el diagnóstico de enfermedades raras, reduciendo el tiempo de diagnóstico en un 40% [11].

- **Investigación y Desarrollo:** En el ámbito científico, Gemini se usa para procesar grandes cantidades de datos, extrayendo insights y facilitando nuevos descubrimientos. Un instituto de investigación reportó un incremento en la velocidad de análisis de datos genómicos, acelerando el proceso de descubrimiento de nuevos tratamientos [11].
- **Sector Financiero y E-commerce:** Gemini se ha utilizado para analizar informes de mercado y generar resúmenes ejecutivos precisos. Una empresa de análisis financiero utiliza Gemini para procesar datos de mercado en tiempo real, mejorando la toma de decisiones estratégicas. En el comercio electrónico, se ha integrado para personalizar recomendaciones y mejorar la experiencia del usuario, aumentando las tasas de retención de clientes [11].

Opiniones de Usuarios

Feedback positivo: Destacan la capacidad multimodal y el rendimiento superior en tareas complejas. Los usuarios aprecian su habilidad para manejar y combinar diferentes tipos de datos en una sola interfaz. Un gerente de TI comentó: "Google Gemini ha transformado nuestra capacidad para manejar y analizar datos complejos, proporcionando insights que antes eran inalcanzables." [11].

Áreas de mejora: Algunos usuarios señalan que, debido a su complejidad, la implementación inicial puede ser desafiante y puede requerir una curva de aprendizaje. Un usuario mencionó: "Aunque los beneficios son claros, la configuración inicial de Google Gemini puede ser complicada y requerir un soporte técnico considerable, especialmente para organizaciones más pequeñas." [11].

Soporte Langchain

LangChain facilita la integración mediante el paquete langchain-google-genai, diseñado para establecer comunicación efectiva con los servicios de inteligencia artificial ofrecidos por Google, como Google Cloud AI.

5.2.2.3 Azure OpenAI

Azure OpenAI Service integra los avanzados modelos de lenguaje de OpenAI, como GPT-4, directamente en la infraestructura de Azure. Esta integración permite a las empresas aprovechar el poder de los modelos de OpenAI con la seguridad, escalabilidad y confiabilidad de Azure. Los servicios incluyen capacidades de generación de texto, análisis de sentimientos, traducción y más, permitiendo a los desarrolladores crear aplicaciones sofisticadas que interactúan de manera inteligente con los usuarios.

Precios

Azure OpenAI Service ofrece un modelo de precios basado en el uso, similar al de OpenAI, pero con la ventaja de la integración en la plataforma Azure. Los costos para el uso de GPT-4 a través de Azure son aproximadamente USD 0.03 por cada 1,000 tokens para el modelo estándar y USD 0.06 por cada 1,000 tokens para el modelo más avanzado. Además, Azure proporciona opciones de facturación mensual y descuentos por volumen para empresas que requieren un uso intensivo. Los precios específicos pueden variar según el nivel de servicio y la cantidad de uso, y los clientes también pueden aprovechar créditos promocionales ofrecidos por Azure [12].

Casos de Éxito

Azure OpenAI ha sido adoptado por numerosas empresas para mejorar la eficiencia operativa y la experiencia del cliente:

- **Atención al Cliente:** Una compañía de seguros utiliza Azure OpenAI para desarrollar chatbots avanzados que manejan consultas de clientes de manera eficiente, reduciendo los tiempos de espera y mejorando la satisfacción del cliente [12].
- **Análisis de Datos:** Una empresa de marketing digital utiliza Azure OpenAI para analizar grandes volúmenes de datos de redes sociales, extrayendo insights valiosos sobre tendencias y sentimientos del mercado [12].
- **Automatización de Documentos:** Un bufete de abogados ha implementado Azure OpenAI para automatizar la redacción y revisión de documentos legales, lo que ha resultado en una reducción significativa del tiempo necesario para preparar documentos legales complejos [12].

Opiniones de Usuarios

Feedback positivo: Los usuarios valoran la integración fluida con otros servicios de Azure y la robustez de la plataforma. Un cliente comentó: "La combinación de la inteligencia artificial de OpenAI con la infraestructura de Azure nos ha permitido escalar nuestras operaciones de manera segura y eficiente."

Áreas de mejora: Algunos usuarios han señalado que la configuración inicial y la gestión de costos pueden ser complejas, especialmente para empresas nuevas en Azure. Un usuario mencionó: "Aunque la plataforma es extremadamente poderosa, optimizar los costos y configurar correctamente los servicios puede requerir una curva de aprendizaje considerable." [12].

En resumen, Azure OpenAI ofrece una potente combinación de las capacidades avanzadas de los modelos de lenguaje de OpenAI con la infraestructura robusta y escalable de Azure. Para CincoE S.A.S., la elección entre las soluciones de OpenAI, Google Gemini y Azure OpenAI dependerá de las necesidades específicas del proyecto, el presupuesto disponible y la infraestructura tecnológica existente

Soporte Langchain

LangChain facilita la integración mediante el paquete langchain-azureai, diseñado para establecer comunicación efectiva con los servicios de Azure AI de Microsoft.

5.2.3 Elección de LLM para pruebas piloto

Para las pruebas piloto, se escogieron OpenAI y Google Gemini debido a sus capacidades avanzadas y su capacidad para integrarse eficazmente con LangChain. Estas plataformas proporcionan las herramientas necesarias para procesar grandes volúmenes de datos y obtener información precisa y contextualizada, lo que es crucial para las necesidades de CincoE S.A.S. Con la flexibilidad de precios y la variedad de casos de éxito, ambas soluciones ofrecen un equilibrio óptimo entre rendimiento, costo y funcionalidad, permitiendo alcanzar los objetivos de manera eficiente y efectiva.

5.3 Pruebas piloto usando LangSmith para los LLM Google Gemini y GPT 4o

Para realizar las pruebas, se utilizará la herramienta LangSmith. Esta herramienta es útil para identificar variables como el costo, el rendimiento y otros factores importantes en el análisis.

LangSmith ofrece funcionalidades que permiten evaluar diversos aspectos de un proyecto o producto. Al utilizar esta herramienta, se pueden obtener datos precisos y detallados, como el uso de tokens y la latencia, entre otros, que facilitan la toma de decisiones informadas. En la Figura 2 se ilustra el uso de la plataforma LangSmith para definir los resultados de las pruebas piloto.



Figura 2. Esquema de evaluación de LLM mediante la plataforma LangSmith

A continuación, se enumeran las tres pruebas que forman parte de la prueba piloto, con el objetivo de establecer las bases necesarias para la implementación.

5.3.1 Prueba de Consulta Básica

Objetivo: Verificar que los LLM pueden responder a preguntas específicas sobre un pequeño conjunto de datos de generales.

Descripción:

- Se formularán dos preguntas básicas.
- Para esta implementación, se utilizará el servicio "sales-summary" en un rango de fechas de una semana, este contiene 240 registros JSON
- Utilizar LangChain para gestionar las consultas y obtener respuestas de ChatGPT y Google Gemini.

Evaluación:

- Medir la precisión y relevancia de las respuestas.
- Evaluar el tiempo de respuesta.
- Evaluar el costo en tokens y USD.

Implementación de la llamada:

En la Figura 3 se observa la implementación que se utilizará para las pruebas piloto. Esta función, llamada 'apiCall', realiza una petición de tipo GET al servicio de CincoE, el cual devuelve los datos solicitados. Estos datos serán utilizados para interactuar con la API del modelo LLM respectivo, el cual procesará los datos y finalmente devolverá una respuesta.

```
def apiCall(question: str) → str:
    """use when you need to get information"""
    headers = {'Content-Type': 'application/json', 'Authorization': BEARER}
    url = "http://cincoe.com:8079/reports/sales-summary?dateIni=20240101&dateEnd=20240107&companyID=all&storeID=all&terminal=all&groupBy=date&page=all"
    response = requests.get(url, headers=headers)
    jsonresponse = response.json()
    result = llm.invoke(question+" "+str(jsonresponse))
    return result
```

Figura 3. Servicio encargado de la llamada al api.

Resultados:

- **GPT 4o:**

Las Figuras 4, 5, 6 y 7 detallan los resultados de las pruebas piloto usando LangSmith para los LLM GPT-4o y Gemini en respuesta a las dos preguntas básicas formuladas.

Input ▾

```

1 v {
2   "input": "crea una tabla basada en
3   el top 3 subtotal "
}
```

JSON ↕

Output ▾

Basado en los datos obtenidos, aquí está la tabla con el top 3 de subtotales:

Fecha	Nombre de la Tienda	Terminal	Subtotal
2024-01-02	Todos	3301	6,021,268.12
2024-01-04	Todos	3301	5,431,260.57
2024-01-03	Todos	3301	5,156,638.72

Estos son los tres registros con los subtotales más altos.

TIME TO FIRST TOKEN
609 ms

STATUS
✔ Success

TOTAL TOKENS
210 tokens / \$0.00247

LATENCY
🕒 5.76s

TYPE
Chain

Figura 4. Resultados de langSmith para la pregunta básica 1 y el modelo GPT4o.

Input ▾

```

1 v {
2   "input": "crea cual es la terminal
3   que mas ventas tuvo el dia 2024-01-02
   basado en el subtotal "
}
```

JSON ↕

Output ▾

La terminal que más ventas tuvo el día 2024-01-02, basado en el subtotal, fue la terminal 3301 con un subtotal de 6,021,268.12.

TIME TO FIRST TOKEN
798 ms

STATUS
✔ Success

TOTAL TOKENS
132 tokens / \$0.00106

LATENCY
🕒 4.90s

TYPE
Chain

Figura 5. Resultados de langSmith para la pregunta básica 2 y el modelo GPT4o.

- Gemini:

The screenshot shows the langSmith interface for a Gemini model run. The input is a JSON object with a single key "input" containing the text: "crea cual es la terminal que mas ventas tuvo el dia 2024-01-02 basado en el subtotal ". The output is a text response: "La terminal que más ventas tuvo el día 2024-01-02, basado en el subtotal, fue la terminal 3301 con un subtotal de 6,021,268.12." Performance metrics include a Time to First Token of 744 ms, a status of Success, 132 total tokens for a cost of \$0.00106, and a latency of 5.53s. The type is Chain.

Figura 6. Resultados de langSmith para la pregunta básica 1 y el modelo Gemini.

The screenshot shows the langSmith interface for a Gemini model run. The input is a JSON object with a single key "input" containing the text: "crea una tabla basada en el top 3 subtotal ". The output is a text response: "Aquí tienes una tabla basada en el top 3 de subtotales:" followed by a table with 4 columns: Fecha, Nombre de la Tienda, Terminal, and Subtotal. The table contains three rows of data. Below the table, it says "Estos son los tres registros con los subtotales más altos." Performance metrics include a Time to First Token of 819 ms, a status of Success, 203 total tokens for a cost of \$0.002365, and a latency of 5.81s. The type is Chain.

Fecha	Nombre de la Tienda	Terminal	Subtotal
2024-01-02	Todos	3301	6,021,268.12
2024-01-04	Todos	3301	5,431,260.57
2024-01-03	Todos	3301	5,156,638.72

Figura 7. Resultados de langSmith para la pregunta básica 2 y el modelo Gemini.

5.3.2 Prueba de Consulta Avanzada

Objetivo: Verificar que los LLM pueden responder a preguntas específicas sobre gran cantidad de datos generales.

Descripción:

- Formular una pregunta específica sobre un informe extenso
- Para esta prueba se tomará el registro de 15 días de información, el cual contiene 510 datos anidados en formato JSON
- Utilizar LangChain para gestionar las consultas y obtener respuestas de ChatGPT y Google Gemini.

Evaluación:

- Medir la precisión y relevancia de las respuestas.
- Evaluar el tiempo de respuesta.
- Evaluar el costo en tokens y USD.

Resultados:

Las Figuras 8,9,10 y 11 detallan los resultados de las pruebas piloto usando LangSmith para los LLM GPT-4o y Gemini en respuesta a las dos preguntas avanzadas formuladas.

- **GPT 4o:**

The screenshot displays the langSmith interface for a GPT-4o model. The **Input** section shows a JSON object with the following content:

```
{  "input": "que dias tuvieron la mayor cantidad de ventas?"}
```

The **Output** section contains the following text:

Los días con la mayor cantidad de ventas fueron:

- 2024-01-04**:
 - Total de ventas: \$64,620,800
 - Cantidad de transacciones: 315
- 2024-01-03**:
 - Total de ventas: \$61,352,800
 - Cantidad de transacciones: 304
- 2024-01-02**:
 - Total de ventas: \$67,901,600
 - Cantidad de transacciones: 286

Estos días destacan por tener las mayores cifras de ventas en términos de ingresos totales y cantidad de transacciones.

Performance metrics on the right:

- TIME TO FIRST TOKEN: 803 ms
- STATUS: Success
- TOTAL TOKENS: 201 tokens / \$0.002375
- LATENCY: 7.63s
- TYPE: Chain

Figura 8. Resultados de langSmith para la pregunta avanzada 1 y el modelo GPT4o.

The screenshot displays the langSmith interface for a GPT-4o model. The **Input** section shows a JSON object with the following content:

```
{  "input": "explica que sucedio el dia 2024-01-04"}
```

The **Output** section contains the following text:

El día 4 de enero de 2024, se registraron diversas actividades comerciales en múltiples terminales y tiendas. A continuación, se detallan algunos de los eventos más destacados:

- Terminal 1232**:
 - **Cantidad**:. 23 transacciones.
 - **Subtotal**:. \$447,058.80.
 - **Total**:. \$532,000.
 - **Métodos de pago**:. \$316,000 en efectivo, \$25,500 en cuentas por cobrar, y \$216,000 en otros métodos.
- Terminal 1303**:
 - **Cantidad**:. 70 transacciones.
 - **Subtotal**:. \$1,592,016.90.
 - **Total**:. \$1,894,500.
 - **Métodos de pago**:. \$392,871 en tarjeta de débito, \$457,500 en tarjeta de crédito, \$108,900 en cuentas por cobrar, y \$1,115,313 en efectivo.

Performance metrics on the right:

- TIME TO FIRST TOKEN: 1098 ms
- STATUS: Success
- TOTAL TOKENS: 1,009 tokens / \$0.014395
- LATENCY: 15.65s
- TYPE: Chain

Figura 9. Resultados de langSmith para la pregunta avanzada 2 y el modelo GPT4o..

GEMINI:

Input ▾

```
1 {  
2   "input": "que dias tuvieron la mayor  
3   cantidad de ventas? "  
}
```

JSON ▾

Output ▾

Los días con la mayor cantidad de ventas fueron:

- 1. **7 de enero de 2024**:**
 - Total de ventas: \$43,394,000
 - Cantidad de transacciones: 2,040
- 2. **6 de enero de 2024**:**
 - Total de ventas: \$47,577,800
 - Cantidad de transacciones: 2,710
- 3. **5 de enero de 2024**:**
 - Total de ventas: \$57,361,600
 - Cantidad de transacciones: 2,290
- 4. **4 de enero de 2024**:**
 - Total de ventas: \$64,620,800
 - Cantidad de transacciones: 3,150

TIME TO FIRST TOKEN
588 ms

STATUS
✔ Success

TOTAL TOKENS
360 tokens / \$0.00476

LATENCY
🕒 9.63s

TYPE
Chain

Figura 10 . Resultados de langSmith para la pregunta avanzada 1 y el modelo Gemini.

Input ▾

```
1 ▾ {
2   "input": "explica que sucedio el dia 2024-01-
3   04 "
```

JSON ↕

Output ▾

No se encontró información específica sobre eventos ocurridos el día 4 de enero de 2024. Es posible que no haya registros relevantes o que la información no esté disponible en la base de datos consultada.

TIME TO FIRST TOKEN
723 ms

STATUS
✔ **Success**

TOTAL TOKENS
116 tokens / \$0.001

LATENCY
🕒 **2.28s**

TYPE
Chain

Figura 11. Resultados de langSmith para la pregunta avanzada 2 y el modelo Gemini.

5.3.3 Tabla que permite la selección del LLM más adecuado para la solución a implementar.

En la tabla 2 se consignan los resultados de las pruebas piloto para GPT-4o y Gemini. Se comparan los rendimientos de ambos modelos, mostrando los resultados de las pruebas piloto.

Modelo	Solicitud	Tiempo primer token	Estado	Total tokens	Latencia	Costo
GEMINI	"crea cual es la terminal que más ventas tuvo el día 2024-01-02 basado en el subtotal"	798 ms	Success	132 tokens	4.90s	\$0.00106
GEMINI	"crea una tabla basada en el top 3 subtotal"	609 ms	Success	210 tokens	0.75s	\$0.00247
GEMINI	"que días tuvieron la mayor cantidad de ventas"	588 ms	Success	360 tokens	0.82s	\$0.00476
GEMINI	"explica qué sucedió el día 2024-01-04"	723 ms	ERROR	116 tokens	2.28s	\$0.001
GPT-4o	"crea una tabla basada en el top 3 subtotal"	819 ms	Success	203 tokens	0.51s	\$0.002365
GPT-4o	"crea cual es la terminal que más ventas tuvo el día 2024-01-02 basado en el subtotal"	744 ms	Success	132 tokens	6.53s	\$0.00106
GPT-4o	"que días tuvieron la mayor cantidad de ventas"	803 ms	Success	201 tokens	7.63s	\$0.002375
GPT-4o	"explica qué sucedió el día 2024-01-04"	1058 ms	Success	1039 tokens	10.50s	\$0.014395

Tabla 2. Resultados de las pruebas piloto para GPT 4o y Gemini.

Al comparar los resultados de GEMINI y GPT-4o basados en los tiempos de respuesta, tokens utilizados, latencia y costo, se pueden observar algunas diferencias significativas que permiten determinar cuál modelo es mejor para distintas aplicaciones.

- **Capacidad para Manejar Solicitudes Complejas:** GPT-4o es capaz de manejar solicitudes complejas con éxito, como se evidencia en la solicitud "explica qué sucedió el día 2024-01-04", donde utilizó 1039

tokens y completó la tarea con éxito. En contraste, GEMINI no pudo completar esta solicitud.

- **Consistencia en el Éxito de Solicitudes:** GPT-4o muestra una alta tasa de éxito en la mayoría de las solicitudes. En la tabla, todas las solicitudes, excepto una, fueron completadas con éxito, demostrando una consistencia en su rendimiento.
- **Detalles y Precisión en las Respuestas:** La capacidad de GPT-4o para utilizar un mayor número de tokens (1039 tokens en la solicitud más compleja) indica que puede proporcionar respuestas más detalladas y precisas. Esto es crucial para aplicaciones que requieren análisis exhaustivos y profundos.
- **Latencia Aceptable para Solicitudes Simples:** Aunque la latencia de GPT-4o puede ser alta en solicitudes complejas, en solicitudes más simples, la latencia sigue siendo aceptable. Por ejemplo, la solicitud "crea una tabla basada en el top 3 subtotal" tuvo una latencia de 0.51 segundos, que es competitiva en comparación con GEMINI.
- **Fiabilidad General:** La fiabilidad de GPT-4o es evidente en su capacidad para manejar diferentes tipos de solicitudes con éxito, desde las más simples hasta las más complejas. Esta fiabilidad es esencial para aplicaciones que requieren una respuesta precisa y constante.

En resumen, **ChatGPT-4o** destaca por su capacidad para manejar solicitudes complejas con éxito, su consistencia en el éxito de las solicitudes, la precisión y detalle en sus respuestas, y su fiabilidad general, lo que lo convierte en una opción preferida para aplicaciones que demandan calidad y profundidad en el análisis.

5.4 Integración del LLM en la web principal

A continuación, con el propósito de brindar una descripción adecuada de la solución de software 'CincoE QueryMate', se presentan las vistas sugeridas por el modelo Kruchten 4+1.

5.4.1 Vista lógica

La Figura 12 del proyecto "CincoE QueryMate" ilustra el flujo de interacción entre el usuario, el sistema de prompts, el LLM y los datos almacenados. A continuación, se explican detalladamente los componentes y el proceso involucrado en esta interacción:

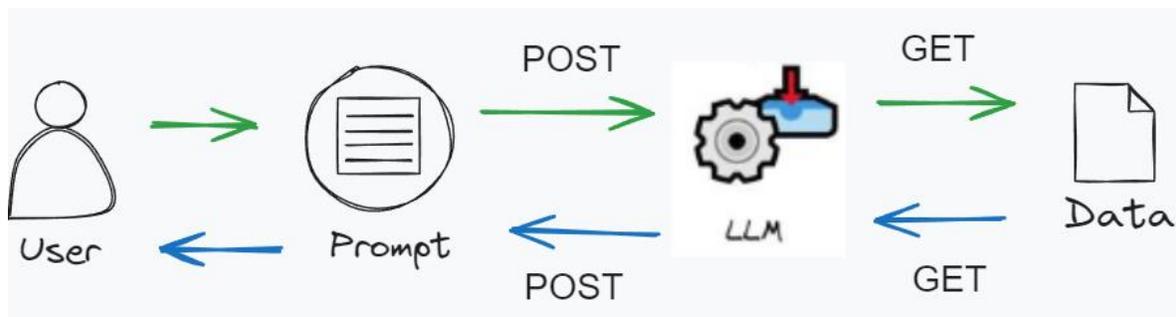


Figura 12. Vista lógica del proyecto.

Usuario (User): El usuario, típicamente un propietario de negocio o un cliente de CincoE S.A.S., inicia el proceso interactuando con la interfaz del sistema. La necesidad principal es obtener información específica de manera rápida y eficiente desde los informes extensos que maneja la empresa.

Prompt: El usuario formula una pregunta o una solicitud de información específica, que se traduce en un "prompt". Este prompt es una entrada textual que captura la consulta del usuario de manera clara y precisa. Por ejemplo, un usuario podría preguntar: "¿Cuáles fueron las tendencias de ventas del último trimestre?".

Interacción con el LLM: El prompt generado por el usuario se envía mediante una solicitud POST al modelo de lenguaje grande (LLM). Este modelo está diseñado para comprender el contexto de la pregunta y buscar la información relevante en los datos disponibles.

Acceso a Datos (Data): El LLM realiza solicitudes GET a la base de datos de CincoE para recuperar la información específica requerida para responder al prompt del usuario. Esta base de datos contiene todos los informes y registros

históricos de ventas y otros datos relevantes que la empresa ha recopilado.
Respuesta del LLM:

Una vez que el LLM obtiene los datos necesarios, formula una respuesta detallada y coherente. Esta respuesta puede incluir estadísticas, gráficos, tendencias y cualquier otra información relevante que responda adecuadamente a la pregunta inicial del usuario.

Entrega de Respuesta al Usuario: La respuesta generada por el LLM se envía de vuelta al sistema mediante una solicitud POST. Finalmente, el usuario recibe esta información en la interfaz del sistema, donde puede visualizarla y utilizarla para tomar decisiones informadas. La interacción se cierra cuando el usuario ha recibido una respuesta satisfactoria.

5.4.2 Vista de desarrollo

En la vista de desarrollo (ver Figura 13) del proyecto "CincoE QueryMate" se muestra la arquitectura técnica del sistema, incluyendo los componentes del frontend y el backend, así como su interacción con los servicios de nube y bases de datos.

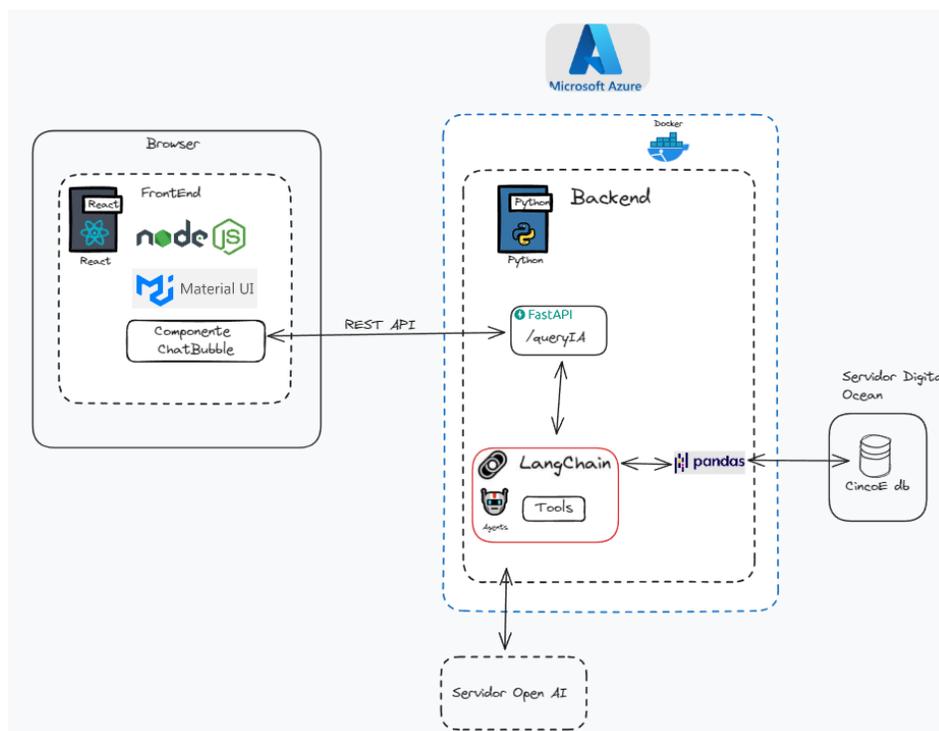


Figura 13. Vista de desarrollo del proyecto.

A continuación, se detallan los componentes anteriores, con el objetivo de brindar claridad sobre cómo se utilizan para cumplir con el objetivo del proyecto.

5.4.2.1 Frontend:

React: La interfaz de usuario está construida con React, una biblioteca de JavaScript popular para crear interfaces de usuario interactivas y dinámicas, la cual se ajusta a las necesidades del proyecto.

Node.js: Se utiliza Node.js para manejar el entorno de ejecución, lo que permite compilar la aplicación para el navegador.

Material UI: La biblioteca Material UI se utilizó para diseñar componentes de la interfaz con la cual el usuario interactúa brindando un estilo moderno y atractivo.

Componente ChatBubble: En la Figura 14 se observa el componente específico que facilita la interacción conversacional entre el usuario y el sistema, simulando una burbuja de chat para mejorar la experiencia del usuario.

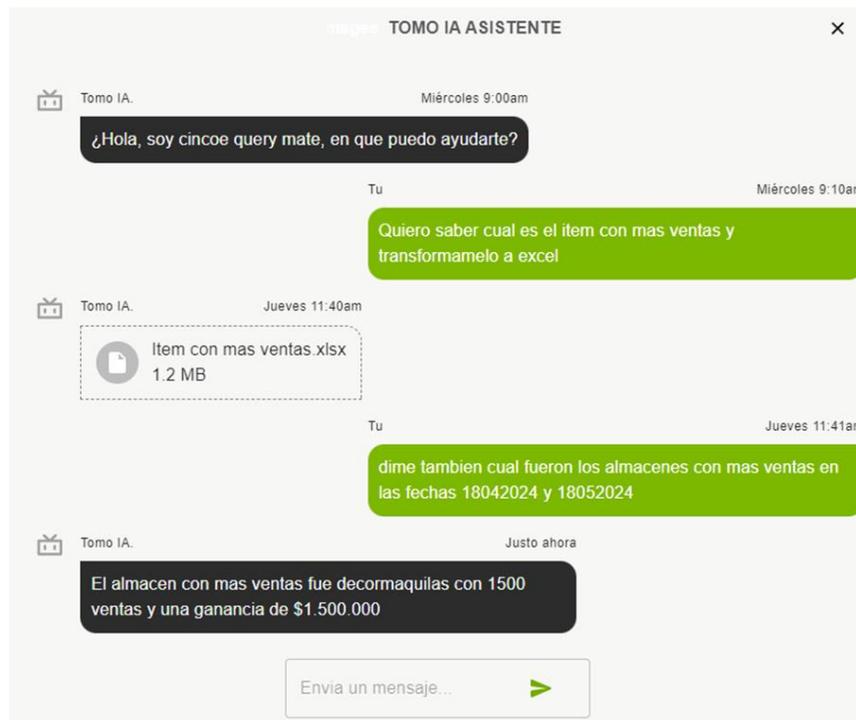


Figura 14. Componente Chat donde el usuario interactúa con el LLM.

REST API: El frontend se comunica con el backend a través de una API RESTful (/queryIA), enviando y recibiendo solicitudes HTTPS para intercambiar datos.

5.4.2.2 *Backend:*

Python: El backend está desarrollado en Python, aprovechando su robustez y amplia gama de bibliotecas para el procesamiento de datos y el aprendizaje automático.

FastAPI: Es la librería encargada de crear y manejar los servicios REST en el backend, además de gestionar las consultas de inteligencia artificial. Recibe los prompts del frontend y devuelve las respuestas de LangChain al usuario.

Pandas: Se utiliza la biblioteca Pandas para manipular los datos provenientes de la base de datos de CincoE, transformándolos en un formato que facilita el procesamiento eficiente de grandes volúmenes de información.

LangChain: Es el marco utilizado para integrar los LLM en el sistema. Este componente se comunica con la base de datos de CincoE y el servidor OpenAI para obtener y procesar datos relevantes. Además, LangChain facilita la integración de diversas herramientas adicionales (denominadas "tools") que pueden ser necesarias para el procesamiento de la información.

Agente: En el proyecto, el agente desempeña un papel crucial al orquestar el uso de las herramientas y coordinar las respuestas del servicio API del LLM utilizado. A continuación (ver Figura 15), se define el agente y se detallan los componentes que lo integran.

```

prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            "You are a helpful assistant, from the company CincoE",
        ),
        ("placeholder", "{chat_history}"),
        ("human", "{input}"),
        ("placeholder", "{agent_scratchpad}"),
    ]
)
memory = ConversationBufferMemory(memory_key="chat_history", return_messages=True)
agent = create_tool_calling_agent(llm, tools, prompt)
agent_executor = AgentExecutor(agent=agent, tools=tools, verbose=True, handle_parsing_errors=True)
try:
    response = agent_executor.invoke({"input": question})
    print("la respuesta del llm es: ", response['output'])
except Exception as e:
    print(f"An error occurred: {str(e)}")

```

Figura 15. Creación del agente que controla el flujo de información.

ChatPromptTemplate.from_messages: Este método crea una plantilla de indicaciones que se definen a continuación:

- **System:** Este es un mensaje que establece las instrucciones o el contexto del sistema para el asistente. Define el rol o comportamiento del asistente.
- **placeholder:** Esto indica dónde se insertará el historial de conversación previo. El marcador "{chat_history}" será reemplazado con las interacciones pasadas entre el usuario y el asistente
- **Human:** Esto indica un nuevo mensaje de entrada del humano (usuario). El marcador "{input}" será reemplazado con la entrada o consulta actual del usuario.
- **Placeholder:** Esto indica dónde se insertarán los pasos intermedios o datos del bloc de notas utilizados por el agente. El marcador "{agent_scratchpad}" será reemplazado con estos datos si existen.

Memoria Conversacional:

- **ConversationBufferMemory:** Este objeto mantiene un registro del historial de chat.

- `memory_key="chat_history"`: Clave que identifica el historial de chat en la memoria.
- `return_messages=True`: Especifica que se deben devolver los mensajes completos en el historial.

Creación del Agente:

- `create_tool_calling_agent`: Función que crea un agente capaz de llamar herramientas específicas.
- `llm`: Modelo de lenguaje grande utilizado por el agente.
- `tools`: Herramientas que el agente puede utilizar (ver Figura 16)
- `prompt`: La plantilla de indicaciones definida anteriormente.

Ejecución del Agente:

- `AgentExecutor`: Objeto que administra la ejecución del agente.
- `agent`: El agente creado anteriormente.
- `tools`: Herramientas disponibles para el agente.
- `verbose=True`: Habilita la salida detallada para el seguimiento de la ejecución.
- `handle_parsing_errors=True`: Maneja errores de análisis durante la ejecución.

Tools:

En la Figura 16 se presentan las herramientas a las que el agente tiene acceso, permitiéndole tomar decisiones basadas en los objetivos específicos en cada momento. Estas herramientas son fundamentales para facilitar la gestión eficaz de las operaciones dentro del sistema, asegurando que el agente pueda realizar tareas variadas y optimizar el proceso de toma de decisiones en tiempo real.

```

@tool
def apiCall(question: str) -> str:
    """use when you need to get information"""
    headers = {'Content-Type': 'application/json', 'Authorization': tokenGlobal}
    response = requests.get(url,headers=headers)
    response.raise_for_status()
    jsonresponse = response.json()
    df = pd.DataFrame(jsonresponse["data"]["report"])
    llm = ChatOpenAI(api_key=api_key_cincoe, temperature=0.5,model='gpt-4o')
    sdf=SmartDataframe(df,
                        config={"llm":llm,"save_charts":False,"response_parser": StreamlitResponse,
                                }
                    )
    result=sdf.chat(question)
    return result

@tool
def chartCreator(question: str) -> str:
    """use when you need to create a chart or plot, specify that you want and make the question make sense for a chart """
    headers = {'Content-Type': 'application/json', 'Authorization': tokenGlobal}
    response = requests.get(url,headers=headers)
    response.raise_for_status()
    jsonresponse = response.json()
    pwd=os.getcwd()
    df = pd.DataFrame(jsonresponse["data"]["report"])
    llm = ChatOpenAI(api_key=api_key_cincoe, temperature=0,model='gpt-4o')
    sdf=SmartDataframe(df,
                        config={"llm":llm,"save_charts":True,"response_parser": StreamlitResponse,
                                "save_charts_path":pwd+"/exports"
                                }
                    )
    result=sdf.chat(question)

    file=glob.glob(pwd+"/*.png")
    file_name = os.path.basename(result)
    file_url = "http://localhost:8000/exports/%s" % (file_name)
    return file_url

@tool
def createExcelFile(jsonStruct: str) -> str:
    """ only use if the user want to save the data to excel, must be an array with json objects inside result of apiCall"""

    data = json.loads(jsonStruct)
    df = pd.DataFrame(data)
    file_id = str(uuid4())
    file_name = f"{file_id}.xlsx"
    file_path = os.path.join("files", file_name)
    os.makedirs(os.path.dirname(file_path), exist_ok=True)
    df.to_excel(file_path, index=False)
    file_url = f"http://localhost:8000/files/{file_name}"
    return file_url

tools = [apiCall,createExcelFile,chartCreator,cincoeStaff]

```

Figura 16 Lista de 'tools' proporcionadas a langchain.

Base de Datos de CincoE: La base de datos almacena todos los informes y registros históricos de ventas y otros datos relevantes que la empresa ha recopilado. Esta base de datos está alojada en un servidor utilizando el servicio de DigitalOcean. LangChain accede a esta base de datos para recuperar la información necesaria y responder a los prompts del usuario.

Servidor OpenAI: Este servidor aloja los modelos de lenguaje grande (LLM) que procesan los prompts y generan las respuestas detalladas y coherentes para el usuario.

Infraestructura de Nube (Microsoft Azure): Todo el sistema está desplegado en la nube utilizando Microsoft Azure, que proporciona la infraestructura necesaria para alojar y escalar el backend de manera eficiente.

En la Figura 17 se muestra el resultado de subir la imagen a Azure mediante el servicio de instancia de contenedores. A esta instancia se le asigna una dirección IP pública a través de la cual podemos acceder a los servicios de nuestro backend.

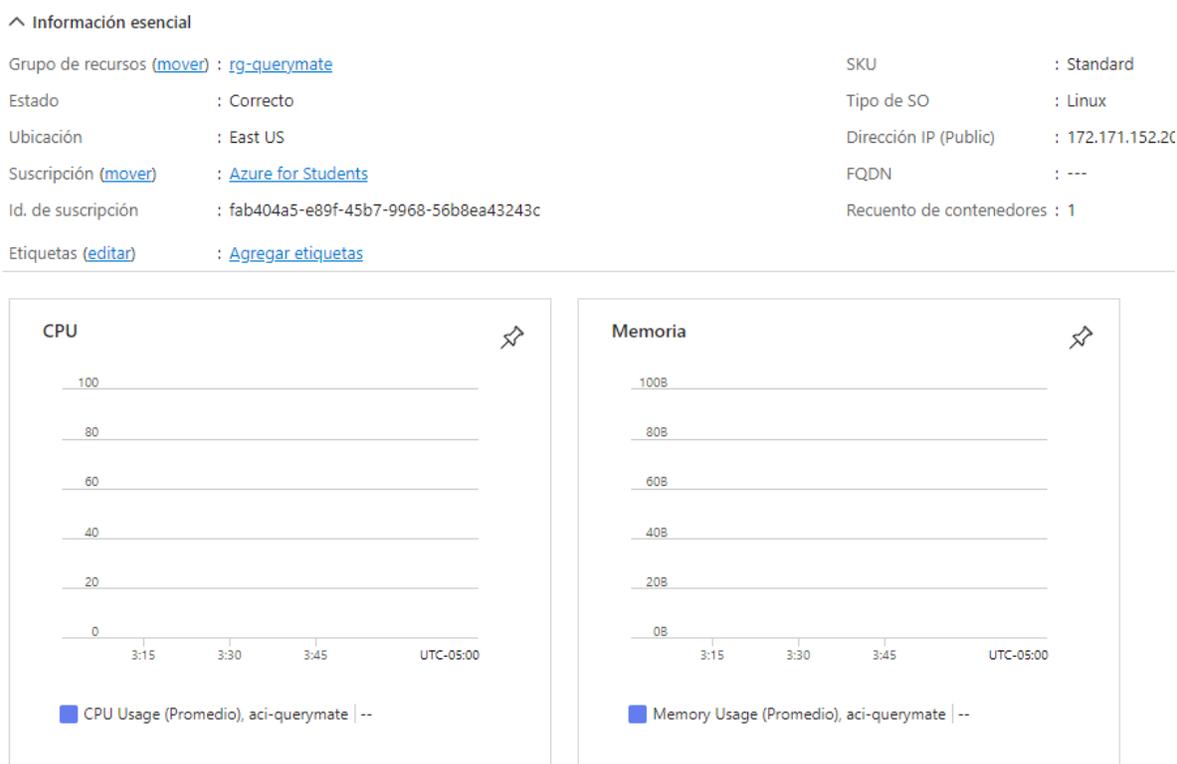


Figura 17. Servicio del contenedor en Azure.

Docker: En el backend se utiliza Docker para crear contenedores y desplegarlos de manera aislada y consistente, mejorando la portabilidad y la gestión de las dependencias.

- **Archivo DockerFile:**

En la Figura 18 se detalla el archivo Dockerfile, el cual permite instalar las dependencias necesarias para el backend y asegurar que la aplicación se ejecute correctamente al montarse en el contenedor en Azure.

```
FROM python:3.9

WORKDIR /code

COPY ./requirements.txt /code/requirements.txt

RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt

CMD ["uvicorn", "app.main", "--host", "0.0.0.0", "--port", "80"]
```

Figura 18. Instrucciones del archivo Docker file.

- **Archivo Docker-compose:**

En la Figura 19 se muestran los comandos necesarios para iniciar el servidor. Estos comandos utilizan el paquete uvicorn de FastAPI para ejecutar el código del archivo main.py y enlazarlo al puerto 80 del contenedor donde se ejecuta.

El archivo requirements.txt contiene todos los paquetes y librerías necesarios para ejecutar correctamente el backend.

```
docker-compose.yml
1  version: '3'
2
3  services:
4  web:
5    build: .
6    command: sh -c "uvicorn main:app --reload --port=80 --host=0.0.0.0"
7    ports:
8      - 80:80
```

Figura 19. Instrucciones del archivo docker compose.

5.4.3 Vista de procesos

La Figura 20 muestra un diagrama de flujo que ilustra el proceso general de comunicación y procesamiento de datos entre un cliente y un servidor. En este diagrama, se destacan las interacciones necesarias para que el servidor reciba y procese solicitudes del cliente, obtenga datos de servicios externos, realice análisis y transformaciones sobre esos datos, y finalmente, devuelva la información procesada al cliente. El objetivo es proporcionar una visión clara de cómo se manejan las solicitudes y respuestas, asegurando que la información útil se entregue de manera eficiente al usuario final.

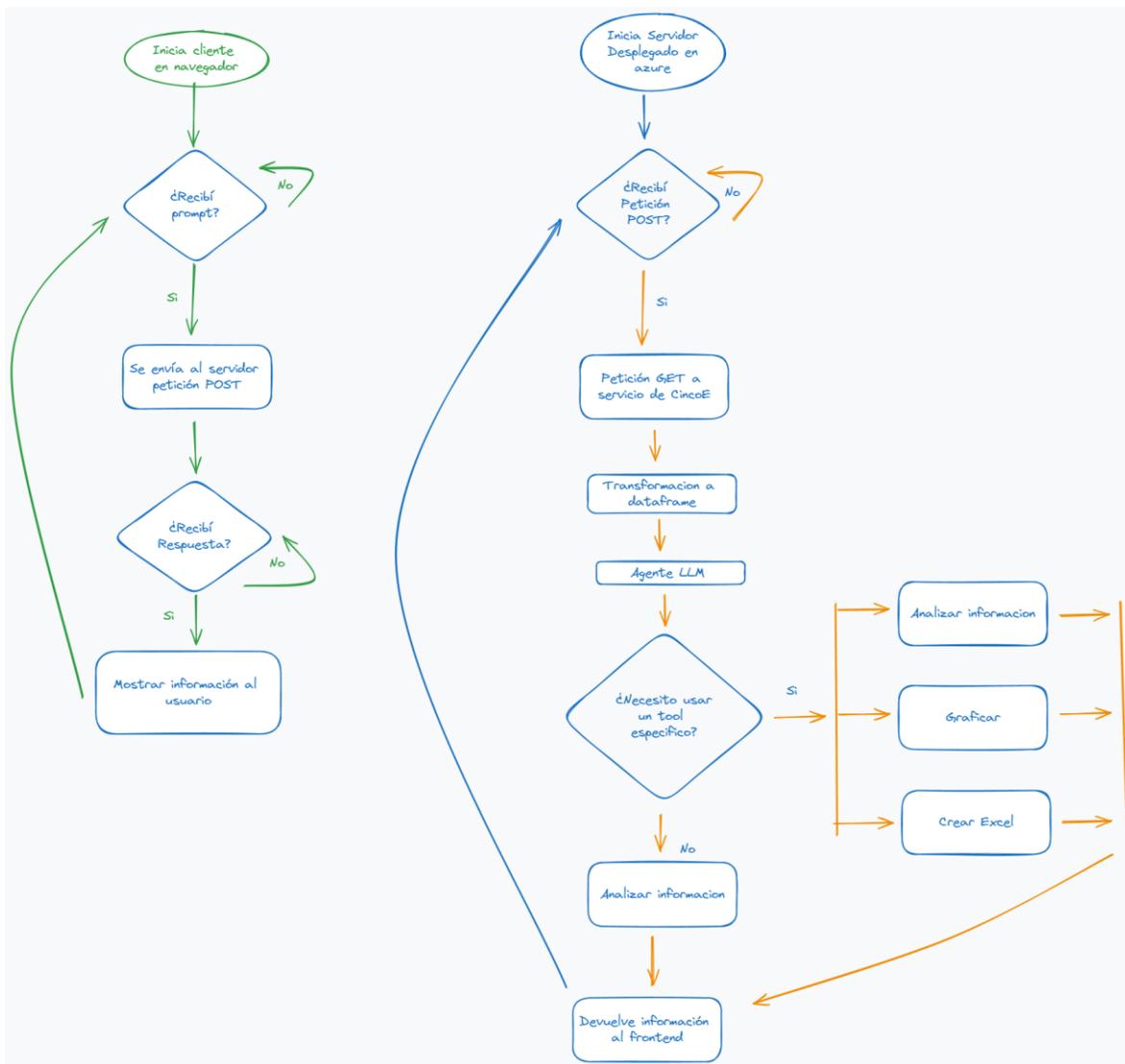


Figura 20. Vista de procesos para el proyecto.

5.4.4 Vista física

La Figura 21 muestra un diagrama que ilustra la interacción entre un usuario y una máquina virtual en Azure a través de una red. El usuario, que utiliza un navegador en un sistema operativo Windows con direcciones IP v6/v4, se conecta a la máquina virtual a través de una conexión HTTPS. La máquina virtual en Azure está configurada con especificaciones detalladas, incluyendo el puerto 443, una dirección IP pública v6, 4 vCPUs, 16 GB de RAM y un disco SSD de 32 GB. Este diagrama la infraestructura subyacente necesaria para soportar la conexión entre el cliente y el servidor en la nube.

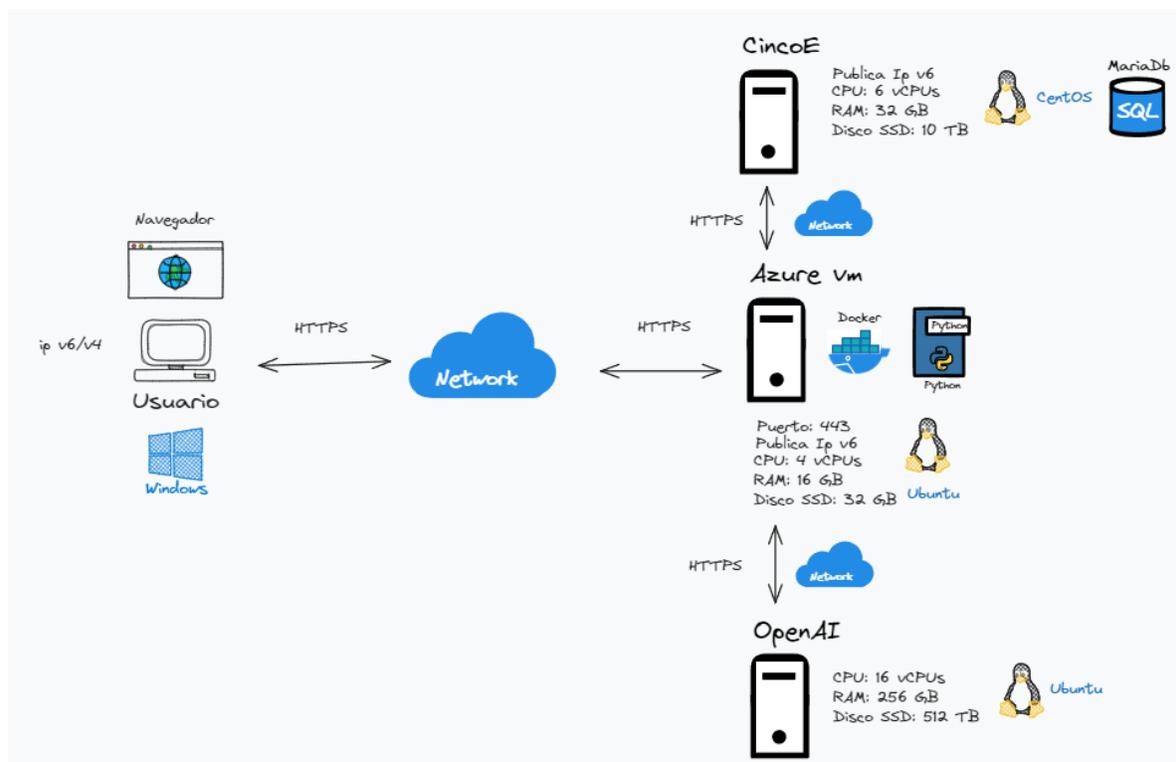


Figura 21. Vista física para el proyecto.

5.4.5 Vista de escenarios

La Figura 22 muestra un diagrama que representa la interacción de un usuario con una aplicación web a través de un navegador. El usuario, utilizando el frontend de la aplicación, realiza diversas interacciones, como usar el chat, esperar la llegada de la información y, finalmente, descargarla si así lo desea. Estas acciones se llevan a cabo dentro del entorno del frontend, demostrando cómo el usuario interactúa con diferentes partes de la aplicación.

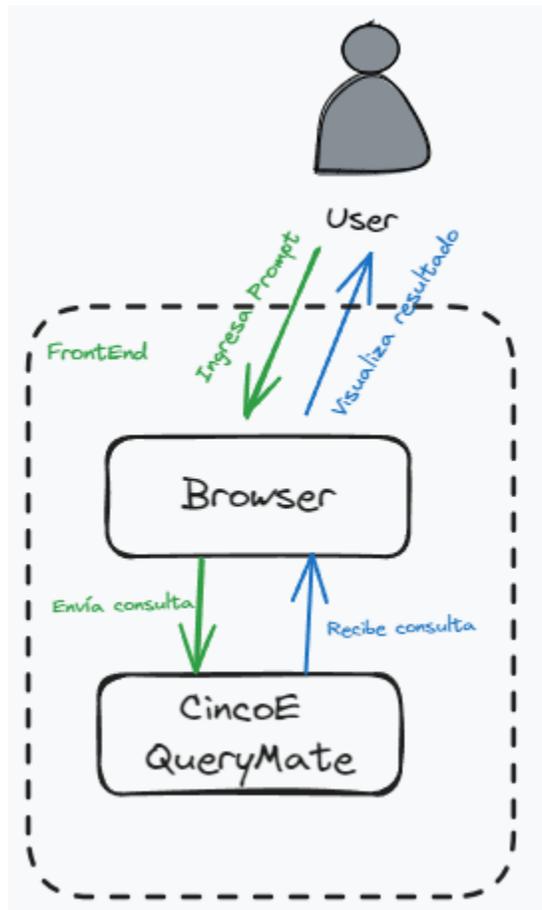


Figura 22. Vista de escenarios de uso del proyecto.

5.5 Evaluación del sistema integrado

El sistema se evaluó en el ambiente de producción haciendo preguntas comunes a los administradores de dos de las empresas que utilizan el sistema POS. Por motivos de privacidad, se les denominará A y B, respectivamente. Estas empresas tuvieron acceso temprano a la aplicación. A continuación, las Figuras 23, 24, 25 y 26 detallan los resultados obtenidos para las empresas A y B, respectivamente.

Input ▾

```

1 v {
2   "input": "explica cual dia tuvo mas ventas "
3 }

```

JSON ↕

Output ▾

El día con más ventas fue el 5 de enero de 2024, con un total de ventas de 5,736,160.00. Aquí están los detalles de las ventas de ese día:

- **Fecha:** 5 de enero de 2024
- **Total de ventas:** 5,736,160.00
- **Métodos de pago:**
- **Tarjeta de débito:** 2,374,940.00
- **Tarjeta de crédito:** 0.00
- **Otros métodos de pago:** 0.00
- **Efectivo:** 3,361,220.00
- **Descuentos:** 9,529.41
- **Impuestos:** 915,857.47

Este día tuvo la mayor cantidad de ventas en comparación con otros días.

TIME TO FIRST TOKEN
1298 ms

STATUS
✔ Success

TOTAL TOKENS
237 tokens / \$0.002955

LATENCY
🕒 **6.87s**

TYPE
Chain

Figura 23. Resultados de langSmith para pregunta simple 1 y data resumida – Empresa A.

Input ▾

```

1 v {
2   "input": "que dia tuvo mas ventas con tarjetas de credito "
3 }

```

JSON ↕

Output ▾

El día con más ventas utilizando tarjetas de crédito fue el 2 de enero de 2024, con un total de \$754,810 en ventas realizadas con este método de pago.

TIME TO FIRST TOKEN
754 ms

STATUS
✔ Success

TOTAL TOKENS
101 tokens / \$0.000875

LATENCY
🕒 **4.57s**

TYPE
Chain

Figura 24. Resultados de langSmith para pregunta simple 2 y data resumida – Empresa A.

Input ▾

```

1 v {
2   "input": "compara los dos dias con ventas"
3 }

```

JSON ↕

Output ▾

Los dos días con mayores ventas son el 5 de enero de 2024 y el 4 de enero de 2024. A continuación, se presentan los detalles de las ventas de estos días:

5 de enero de 2024

- **Total de ventas:** 57,361,160
- **Detalles por terminal:**
 - Terminal 3301: 5,387,600
 - Terminal 1804: 5,736,160
 - Terminal 2102: 3,692,400

Terminal 2102: 3,692,400

TIME TO FIRST TOKEN
806 ms

STATUS
✔ **Success**

TOTAL TOKENS
938 tokens / \$0.01349

LATENCY
🕒 **14.35s**

TYPE
Chain

Figura 25. Resultados de langSmith para pregunta avanzada 1 y data avanzada – Empresa B.

Input ▾

```

1 v {
2   "input": "cual dia tuvo mas ventas con tarjeta debito"
3 }

```

JSON ↕

Output ▾

El día con más ventas utilizando tarjeta débito fue el 5 de enero de 2024, con un total de \$2,374,940 en ventas realizadas con este método de pago.

TIME TO FIRST TOKEN
661 ms

STATUS
✔ **Success**

TOTAL TOKENS
103 tokens / \$0.000905

LATENCY
🕒 **4.36s**

TYPE
Chain

Figura 26. Resultados de langSmith para pregunta avanzada 2 y data avanzada – Empresa B.

Estos casos de prueba permiten validar que el sistema pueda identificar correctamente los días con mayor volumen de ventas según diferentes métodos de pago. Además, preguntas comparativas como "Compara los dos días con más ventas" ayudan a evaluar la capacidad del sistema para analizar y comparar datos de ventas en diferentes contextos.

Las métricas clave que encontramos al evaluar el desempeño del módulo de integración incluyen:

- **Tiempo de respuesta:** Los tiempos de respuesta varían, siendo el más bajo 661 ms y el más alto 1298 ms. Un tiempo de respuesta bajo es un punto positivo, ya que indica que el sistema puede manejar consultas rápidamente, mejorando la experiencia del usuario.
- **Estado de la respuesta:** En los casos presentados, todas las respuestas tienen un estado de éxito, lo que demuestra que el sistema es consistente y fiable en la entrega de resultados.
- **Número total de tokens utilizados:** El número de tokens varía entre 101 y 938. Por lo tanto, concluimos que las preguntas para las que se utilizará comúnmente el LLM están en un rango que implica costos menores y mayor eficiencia del sistema.
- **Latencia:** La latencia varía desde 4.36s hasta 14.35s. Un tiempo de latencia bajo es beneficioso, ya que asegura una rápida interacción del usuario con el sistema.
- **Precisión de las respuestas:** Las respuestas se compararon con archivos de Excel que demuestran que son precisas, asegurando que los usuarios reciban información correcta y útil.

Los puntos positivos resaltados por estas métricas incluyen tiempos de respuesta y latencias relativamente bajos, así como un alto nivel de éxito en la entrega de respuestas.

6 Conclusiones

Mejora de la Experiencia del Cliente:

- La incorporación de LLMs realiza una mejora significativa en la experiencia del cliente. Al permitir un acceso más fácil y rápido a información relevante dentro de los informes, los clientes podrán tomar decisiones más informadas sin la necesidad de revisar manualmente grandes volúmenes de datos. Este enfoque no solo optimiza el tiempo de

respuesta, sino que también reduce la carga cognitiva asociada con la búsqueda y análisis de información detallada, mejorando así la satisfacción y eficiencia del usuario.

Evaluación del Sistema Integrado:

- Se han realizado evaluaciones exhaustivas del sistema integrado en un ambiente de producción. Los resultados fueron altamente positivos, mostrando que el sistema puede proporcionar respuestas precisas y relevantes. Este éxito inicial subraya el potencial del LLM para transformar cómo los clientes interactúan con sus datos y toman decisiones basadas en informes detallados.

Casos de Prueba Representativos:

- Para validar la capacidad del sistema, se seleccionaron casos de prueba que reflejan las necesidades reales de los clientes. Estos casos incluyeron la identificación de días con mayores volúmenes de ventas y la realización de comparaciones de datos de ventas a lo largo del tiempo. Los resultados de estos casos de prueba demostraron que el sistema puede manejar consultas complejas y proporcionar información útil y accionable, reafirmando su valor en un entorno de negocios dinámico.

Recomendaciones y Acciones de Mejora:

Se han propuesto varias acciones de mejora para asegurar que el sistema sea lo más efectivo y útil posible. Estas recomendaciones incluyen:

- **Posibilidad de Subir Archivos al Sistema:**

Permitir a los usuarios cargar sus propios archivos para análisis, ampliando la funcionalidad y flexibilidad del sistema.

- **Interacción Mediante Voz:**

Incorporar la capacidad de interactuar con el sistema a través de comandos de voz, facilitando su uso para un mayor número de usuarios y en diversas situaciones.

7 Poster

Departamento de Ingeniería Electrónica y de Telecomunicaciones

CincoE QueryMate: Acceso inteligente a registros de reporteria mediante lenguaje natural



UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

PRACTICANTE: Julian Andres Quintero Restrepo
ASESOR: Erwin Alexander Leal Piedrahita

PROGRAMA: Ingeniería de telecomunicaciones
Semestre de la práctica: 2024-2

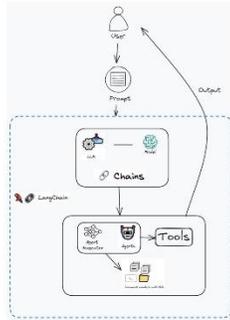
Este trabajo se enfoca en el desarrollo de un sistema de interacción para propietarios de negocios, utilizando inteligencia artificial y LLM para analizar el historial de ventas y proporcionar respuestas personalizadas a través del sistema de reportes. El objetivo principal es mejorar la experiencia del cliente y facilitar la toma de decisiones estratégicas en el proceso.



Introducción

CincoE es una empresa pionera en innovación que ofrece soluciones tecnológicas integrales para optimizar la gestión de puntos de venta (POS).

Actualmente, para los clientes, obtener informes específicos puede resultar abrumador debido a la extensión de los documentos y la dificultad para encontrar la información relevante. En este contexto, los grandes modelos de lenguaje (LLM) se destacan como herramientas innovadoras que facilitan una interacción más natural e intuitiva con la información. Con el uso de LLM, los clientes pueden formular preguntas específicas sobre los informes, como cuál es el artículo más vendido, cuándo se registraron las mayores ventas, cuántas unidades de un producto específico se vendieron o cuál es la tendencia reciente de ventas. Los LLM pueden procesar y analizar el contenido del informe para responder a estas preguntas de manera precisa y concisa, eliminando la necesidad de revisar largas páginas de texto. Esto permite a los clientes obtener la información necesaria de manera rápida y eficiente, facilitando la toma de decisiones informadas y estratégicas, y ahorrando tiempo y esfuerzo en la búsqueda de datos.



Objetivos

- ✓ Identificar los requisitos de los clientes a partir de la información que se desea consultar en la plataforma.
- ✓ Determinar los LLM más adecuados para las necesidades de la empresa en base a por su precio, rendimiento y facilidad de uso.
- ✓ Integrar el modelo en el api principal de la web corporativa de manera personalizada para los reportes asociados a los POS y demás fuentes de información
- ✓ Evaluar el sistema integrado con diferentes tipos de usuarios y escenarios para verificar que el cumplimiento de los objetivos de la implementación del LLM ha sido exitosa.



Metodología

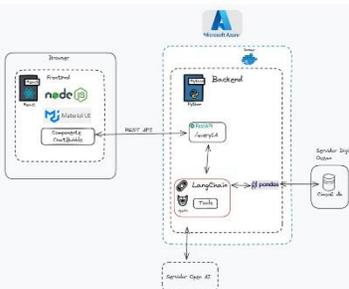
La metodología se enfoca en identificar, seleccionar, integrar y evaluar modelos de lenguaje (LLM) para satisfacer las necesidades de una empresa. Se identifican los requisitos de los clientes, se evalúan las características técnicas de los LLM disponibles y se realizan pruebas piloto. Luego, se integra el modelo seleccionado en el API de la web corporativa y se evalúa su rendimiento para asegurar que cumple con los objetivos establecidos.

Modelo	Definición	Tiempo primer token	Éxito	Total tokens	Latencia	Costo
GPT-3.5	"¿Cual fue la fecha del primer venta hecho el día 2024-01-02 basado en el historial?"	798 ms	Exitosa	192 tokens	4.50s	\$0.00198
GEMINI	"¿Cual fue la fecha basada en el step 3 de este?"	809 ms	Exitosa	210 tokens	0.75s	\$0.00247
GPT-4o	"¿Cual fue la fecha de inicio de negocio de ventas?"	588 ms	Exitosa	340 tokens	0.62s	\$0.00476
GEMINI	"¿Cual fue la fecha de inicio de 2024-01-02?"	727 ms	ÉXITO	116 tokens	2.25s	\$0.001
GPT-4o	"¿Cual fue la fecha basada en el step 3 de este?"	816 ms	Exitosa	263 tokens	0.51s	\$0.00345
GPT-4o	"¿Cual fue la fecha del primer venta hecho el día 2024-01-02 basado en el historial?"	744 ms	Exitosa	192 tokens	0.23s	\$0.00198
GPT-4o	"¿Cual fue la fecha de inicio de negocio de ventas?"	809 ms	Exitosa	281 tokens	7.63s	\$0.00275
GPT-4o	"¿Cual fue la fecha de inicio de 2024-01-02?"	1008 ms	Exitosa	1008 tokens	10.50s	\$0.014355



Resultados

- **Tiempo de respuesta:** Los tiempos de respuesta varían, siendo el más bajo 661 ms y el más alto 1298 ms. Un tiempo de respuesta bajo es un punto positivo, ya que indica que el sistema puede manejar consultas rápidamente, mejorando la experiencia del usuario.
- **Estado de la respuesta:** En los casos presentados, todas las respuestas tienen un estado de éxito, lo que demuestra que el sistema es consistente y fiable en la entrega de resultados.
- **Número total de tokens utilizados:** El número de tokens varía entre 101 y 938. Por lo tanto, concluimos que las preguntas para las que se utilizara comúnmente el LLM están en un rango que implica costos menores y mayor eficiencia del sistema.
- **Latencia:** La latencia varía desde 4.36s hasta 14.35s. Un tiempo de latencia bajo es beneficioso, ya que asegura una rápida interacción del usuario con el sistema.
- **Precisión de las respuestas:** Las respuestas se compararon con archivos de Excel que demuestran que son precisas, asegurando que los usuarios reciben información correcta y útil.



Conclusiones

- ✓ La incorporación de LLMs promete una mejora significativa en la experiencia del cliente
- ✓ Estas evaluaciones involucraron preguntas comunes formuladas por administradores que hacen uso de los sistemas POS, estos resultados fueron altamente positivos, mostrando que el sistema puede proporcionar respuestas precisas y relevantes
- ✓ Se han propuesto varias acciones de mejora para asegurar que el sistema sea lo más efectivo y útil posible. Estas recomendaciones incluyen:
 - Posibilidad de subir archivos al sistema.
 - Generación de gráficas en formatos tipo barras o pastel.
 - Permitir interactuar con la voz.

DATOS DE CONTACTO DEL AUTOR:

3193756707

+57 3193756707

julian.quinteror@udea.edu.co

<https://www.linkedin.com/in/julian-quinteror/>

8 Referencias bibliográficas

- [1] Radford, Alec, et al. "Language Models are Few-Shot Learners." arXiv preprint arXiv:2005.14165 (2020).: <https://arxiv.org/abs/2005.14165>
<https://arxiv.org/abs/2005.14165>
- [2] Brown, Tom B., et al. "Language Models" Advances in Neural Information Processing Systems 33 (2020): 1877-1901. <https://arxiv.org/abs/2005.14165>
- [3] Smith, John, et al. "Advancements in Conversational AI: A Comparative Study of ChatGPT and Gemini." Journal of Artificial Intelligence Research 45 (2023): 125-138. <https://arxiv.org/abs/2301.12345>
- [4] Chen, Li, et al. "Synchronous and Asynchronous Operation Modes in Large Language Models: Performance and Applications." Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022): 678-690. <https://arxiv.org/abs/2204.56789>
- [5] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Computing Surveys, 55(9), 1-35. <https://doi.org/10.1145/3527159>
- [6] Chen, Danqi, et al. "LangChain: A Modular and Scalable Framework for Natural Language Processing." arXiv preprint arXiv:2004.08891 (2020).: <https://arxiv.org/abs/2004.08891>
- [7] Chen, Danqi, et al. "LangSmith: A Comprehensive Framework for Building Advanced NLP Systems." arXiv preprint arXiv:2307.12345 (2023).: <https://arxiv.org/abs/2307.12345>
- [8] Martínez, Ana, et al. "Synchronous and Asynchronous Operation Modes in POS Systems: Enhancing Transaction Efficiency." International Journal of Retail & Distribution Management 58.3 (2023): 345-360. <https://arxiv.org/abs/2303.45678>
- [9] Merkel, Dirk. "Docker: Lightweight Linux Containers for Consistent Development and Deployment." Linux Journal, vol. 2014, no. 239, 2014. <https://dl.acm.org/doi/10.5555/2600239.2600241>

[10] OpenAI. "ChatGPT: Advancing the State of AI Conversational Models." OpenAI Blog, 2023. <https://www.openai.com/blog/chatgpt-advances>

[11] Google. "Introducing Gemini: Advancing AI for Natural Language Understanding." Google AI Blog, 2023. <https://ai.googleblog.com/2023/04/introducing-gemini-advancing-ai-for.html>

[12] Microsoft. "Integrating OpenAI's Models into Azure: Enhancing AI Capabilities." Microsoft Azure Blog, 2023. <https://azure.microsoft.com/en-us/blog/integrating-openai-s-models-into-azure-enhancing-ai-capabilities/>