Chatbot RAG APP

Fernandez Ian
Challenge Promtior
Resistencia (Chaco), Argentina

{ianstgo@gmail.com}

Resumen. Los Modelos de Lenguaje a Gran Escala (LLM) son entrenados principalmente mediante aprendizaje autosupervisado, en el que se utilizan enormes cantidades de datos textuales sin etiquetar para que el modelo aprenda patrones, estructuras y relaciones en el lenguaje. Posteriormente, en algunos casos se aplica un refinamiento con aprendizaje supervisado o mediante técnicas de Reinforcement Learning from Human Feedback (RLHF) para afinar su desempeño en tareas específicas. Es importante destacar que la inteligencia en estos modelos reside en la capacidad del LLM para comprender y generar lenguaje natural a partir de ese entrenamiento. La implementación realizada en este informe presenta una solución o mejora al funcionamiento y respuesta de los LLM. La implementación denominada RAG (Retrieval Augmented Generation Application) esta solución se orienta a resolver uno de los desafíos más críticos en el uso de LLMs: la generación de respuestas que sean no sólo coherentes y fluidas, sino también factualmente fundamentadas y contextualmente relevantes. Con RAG, cuando se realiza una consulta, el sistema primero recupera información pertinente de una base de datos o repositorio de documentos, y luego utiliza esa información para enriquecer la respuesta generada por el LLM. De esta manera, se consigue una sinergia que permite obtener respuestas más completas y precisas, especialmente en contextos donde la veracidad y la actualización de la información son esenciales. En este documento mostraremos los componentes de la solución propuesta al desafío de desarrollar un chatbot con arquitectura RAG, el cual debe responder preguntas relacionadas a Promtior. También en este documento se encuentran las herramientas utilizadas y capturas que demuestran la aplicación funcional junto con la respuesta.

1 Introducción

Una arquitectura RAG (Retrieval Augmented Generation Application) es una solución diseñada para potenciar el rendimiento de los modelos de lenguaje (LLMs) al integrar capacidades de recuperación de información con generación de texto. Su objetivo principal es mejorar la precisión y relevancia de las respuestas generadas, combinando la habilidad de los LLMs para comprender y sintetizar lenguaje natural con la capacidad de acceder y extraer información actualizada o especializada de fuentes externas.

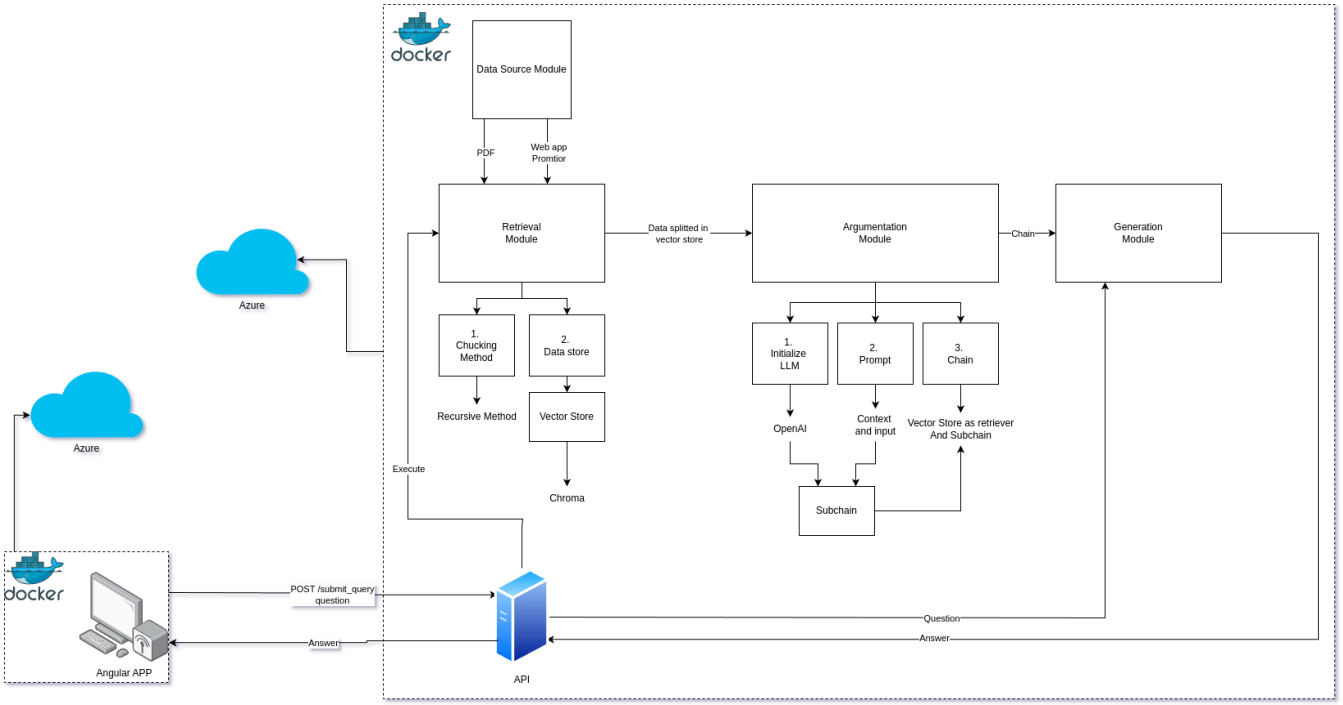
La arquitectura RAG(Retrieval Augmented Generation Application) no es, en sí misma, la "IA" sino un marco que integra dos componentes esenciales:

1. El LLM, que representa la parte de inteligencia del sistema, encargado de generar respuestas coherentes y contextualmente relevantes.
2. El motor de recuperación (retrieval), que busca y extrae información actualizada y especializada de fuentes externas o repositorios de datos.

Mediante la combinación de estas dos funcionalidades, una arquitectura RAG enriquece las respuestas generadas por el LLM al integrarlas con información real y verificable. Esto permite que la solución no solo dependa de la base de conocimiento pre entrenada del modelo, sino que también se adapte a datos actualizados o específicos de un dominio, mejorando significativamente la precisión y la utilidad de las respuestas.

2 Descripción de la solución implementada

La solución implementada para desarrollar un chatbot bajo la arquitectura RAG se puede explicar en el siguiente gráfico :



Como podemos observar la solución se divide en dos partes. La parte del frontend, la cual interactúa con el usuario fue desarrollada en Angular, dockerizada y desplegada en Azure. La parte del backend fue desarrollada en Python, en la cual se divide en dos partes, los módulos de la arquitectura RAG y la API desarrollada con la librería FAST API, toda la app fue también dockerizada y deployada en azure.

Lo más importante es la arquitectura RAG la cual fue desarrollada en 4 módulos.

- Data Source Module
- Retrieval Module
- Argumentation Module
- Generation Module

Data Source Module brinda los datos necesarios para la aplicación, como son en este caso el pdf del challenge y también el texto encontrado en la web de Promtior.
Retrieval Module recibe los datos de entrada, como primer paso aplica un método de separación de datos, en este caso la recursividad. Dicho método nos permite obtener fragmentos de textos adecuados, organizados en cierto patrón y tamaño. El segundo paso consiste en guardar en una base de datos de vectores los datos obtenidos en el paso uno. En nuestro caso usamos la base de datos Chroma. La importancia de una base de datos de vectores es que permite identificar rápidamente cuáles fragmentos de información son semánticamente relevantes a una consulta.
Argumentation Module recibe la base de datos de vectores, en donde aplica 3 pasos, la primera es la inicialización del LLM que en este caso es OpenAI. El segundo paso es la generación del prompt indicando un contexto adecuado. A partir del primer y segundo paso podemos generar una simple subcadena. Esta subcadena se combina en el 3er paso, el cual consiste en convertir nuestra base de datos de vectores en un recuperador de datos y brindarle el LLM inicializado más el prompt. Con ello ya tenemos una cadena, la cual es un objeto “callable”, un disparador ante una consulta, que recupera y responde a la consulta ingresada.
Generation Module recibe la cadena y la pregunta realizada a través de la API, este módulo simplemente ejecuta la cadena ingresando la consulta. Como salida obtenemos la respuesta aplicando la arquitectura RAG.

3 Herramientas utilizadas

Lenguajes de programación :

- Python 3.12
- Typescript

Librería de API :

- Fast API

Frontend :

- Angular 18

LLM :

- OpenAI

Framework de LLM :

- LangChain

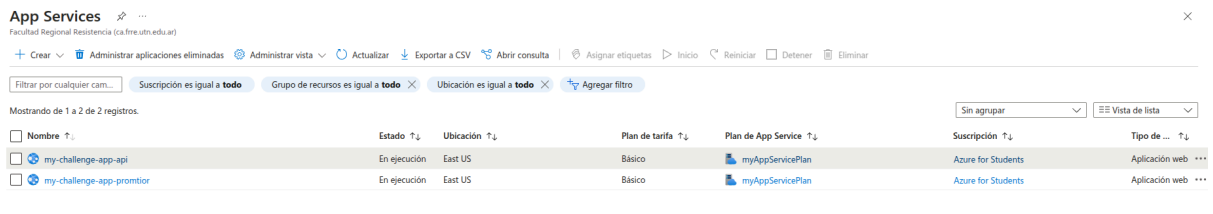
Base de datos de vectores :

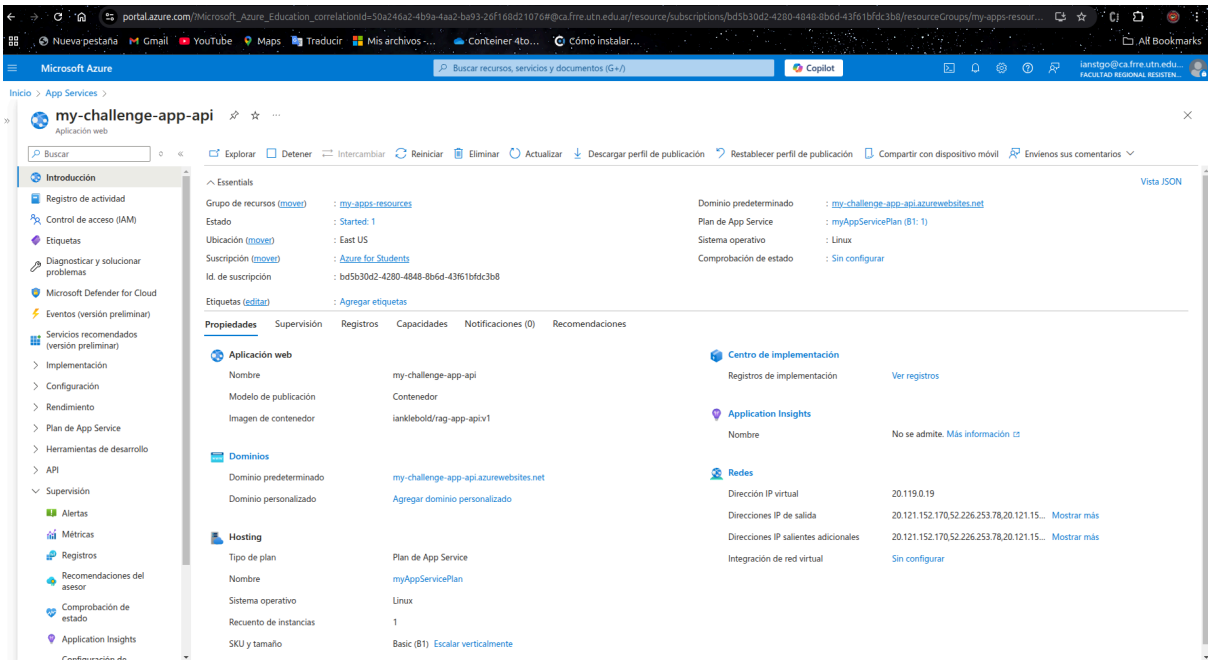
- Chroma

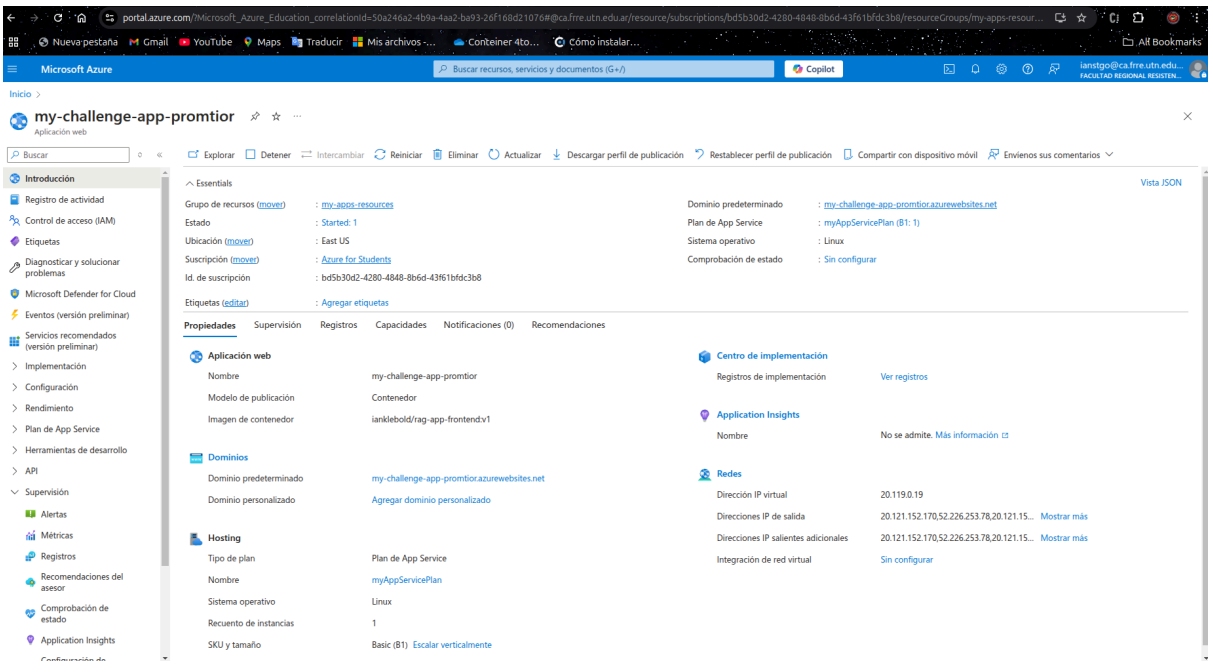
Herramientas de despliegue:

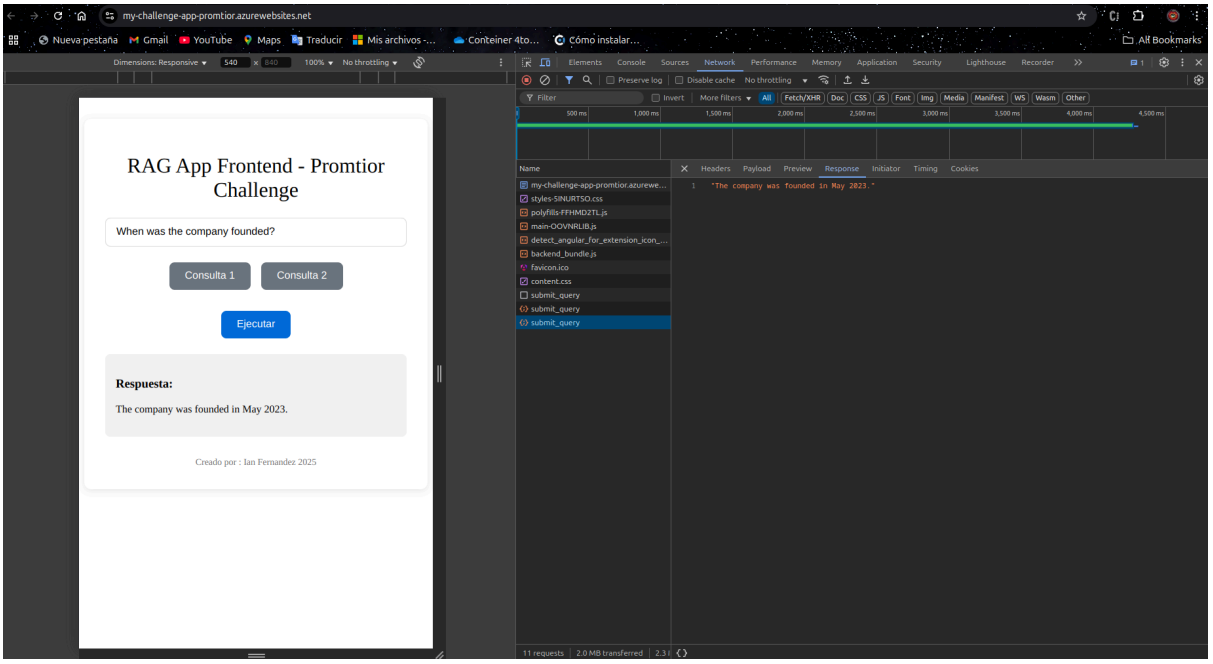
- Docker
- Docker-compose
- Azure CLI

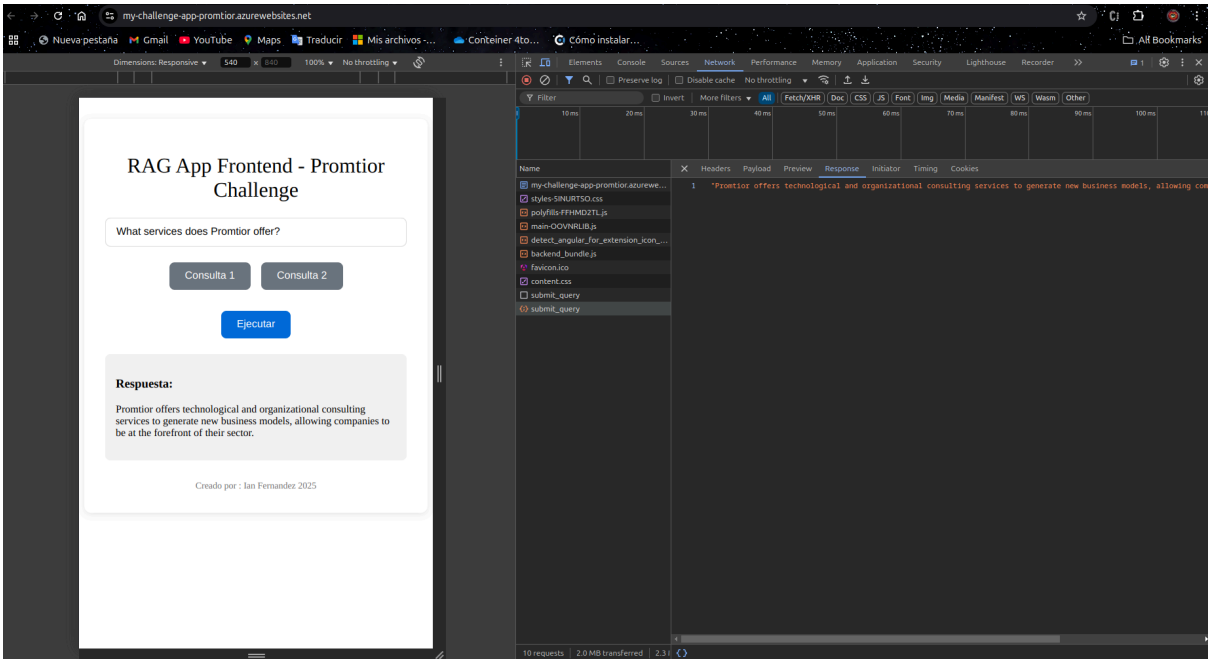
4 Capturas :

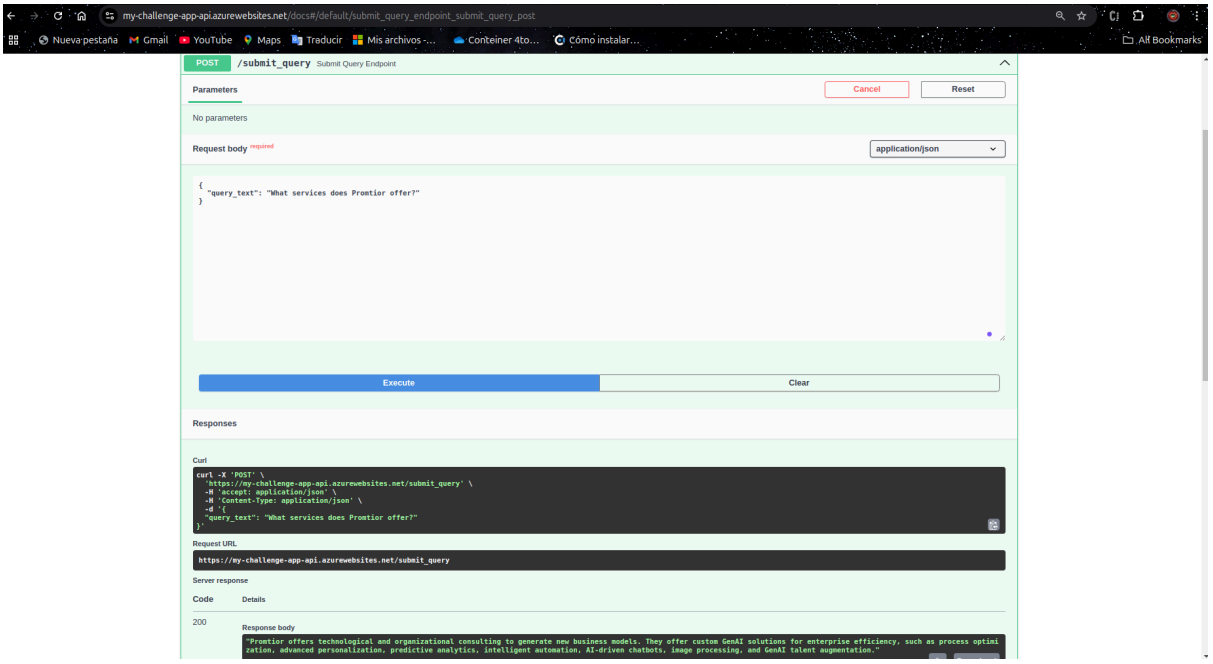





portal.azure.com/?Microsoft_Azure_Education_correlationId=50a246a2-4b9a-4aa2-ba93-26f168d21076#@ca.frre.utn.edu.ar/resource/subscriptions/bd5b30d2-4280-4848-8b6d-43f61bfdc3b8/resourceGroups/my-apps-resour...
Microsoft Azure
Buscar recursos, servicios y documentos (G+/)
Copilot
Inicio
my-challenge-app-promtior
Aplicación web
Explorar
Detener
Intercambiar
Reiniciar
Eliminar
Actualizar
Descargar perfil de publicación
Restablecer perfil de publicación
Compartir con dispositivo móvil
Envíenos sus comentarios
Essentials
Vista JSON
Grupo de recursos (mover) : my-apps-resources
Estado : Started: 1
Ubicación (mover) : East US
Suscripción (mover) : Azure for Students
Id. de suscripción : bd5b30d2-4280-4848-8b6d-43f61bfdc3b8
Dominio predeterminado : my-challenge-app-promtior.azurewebsites.net
Plan de App Service : myAppServicePlan (B1: 1)
Sistema operativo : Linux
Comprobación de estado : Sin configurar
Etiquetas (editar) : Agregar etiquetas
Propiedades
Supervisión
Registros
Capacidades
Notificaciones (0)
Recomendaciones
Aplicación web
Nombre my-challenge-app-promtior
Modelo de publicación Contenedor
Imagen de contenedor ianklebold/rag-app-frontend:v1
Dominios
Dominio predeterminado my-challenge-app-promtior.azurewebsites.net
Dominio personalizado Agregar dominio personalizado
Hosting
Tipo de plan Plan de App Service
Nombre myAppServicePlan
Sistema operativo Linux
Recuento de instancias 1
SKU y tamaño Basic (B1) Escalar verticalmente
Centro de implementación
Registros de implementación Ver registros
Application Insights
Nombre No se admite. Más información
Redes
Dirección IP virtual 20.119.0.19
Direcciones IP de salida 20.121.152.170,52.226.253.78,20.121.15... Mostrar más
Direcciones IP salientes adicionales 20.121.152.170,52.226.253.78,20.121.15... Mostrar más
Integración de red virtual Sin configurar


my-challenge-app-promtior.azurewebsites.net
RAG App Frontend - Promtior Challenge
When was the company founded?
Consulta 1
Consulta 2
Ejecutar
Respuesta:
The company was founded in May 2023.
Creado por : Ian Fernandez 2025
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Name
Headers
Payload
Preview
Response
Initiator
Timing
Cookies
"The company was founded in May 2023."
11 requests
2.0 MB transferred





5 Conclusiones

Este challenge ha sido una experiencia increíblemente enriquecedora y desafiante. La mayor complicación se presentó al seleccionar el LLM adecuado para la solución. Inicialmente se consideró utilizar Llama, pero rápidamente se evidenciaron ciertas limitaciones y problemas en su desempeño y manejo del contexto, lo que dificultó su aplicación en un entorno de producción robusto.

Frente a estos obstáculos, se optó por cambiar a un modelo de mayor confiabilidad y escalabilidad, integrándose en una arquitectura de Retrieval Augmented Generation (RAG APP). La solución final combina la capacidad generativa del LLM con una base de datos de vectores (Croma) para recuperar información relevante, lo que permite generar respuestas precisas y fundamentadas.

Este enfoque híbrido no solo superó los inconvenientes iniciales, sino que también demostró la potencia de combinar técnicas de recuperación y generación para aplicaciones avanzadas en inteligencia artificial. Sin duda, el challenge me permitió explorar y aprender en profundidad sobre la integración de modelos de lenguaje y sistemas de recuperación, abriendo nuevas oportunidades para el desarrollo de soluciones innovadoras.

6 Referencias

1. https://python.langchain.com/docs/tutorials/rag/

2. Distintos contenido multimedia de internet.