

Tecnólogo en Informática



Asistente Virtual con IA



Integrantes:

Carlos Santana
Andres Romano
Juan Pilón
Nicolás Silva

CI: 5.017.001-9
CI: 4.760.801-9
CI: 4.692.663-6
CI: 4.885.055-6

Tutor:

Pablo Granero

Año:

2024

Contenido

Contenido.....	1
Organización del documento.....	4
Resumen del trabajo.....	5
Capítulo I: Introducción.....	6
Descripción del cliente.....	6
Introducción al problema.....	6
Objetivo general.....	6
Objetivos específicos.....	7
Descripción del producto.....	7
Capítulo II: Estado del arte.....	8
Introducción al capítulo.....	8
Investigación.....	8
Estudio de mercado.....	8
Tipos de chatbots existentes en la actualidad.....	10
Formas para incluir bases de datos al chatbot.....	12
Cuadro de comparación de tecnologías.....	13
Uso del modelo y desarrollo del chat.....	14
Sistemas similares.....	14
Conclusiones.....	15
Camino a seguir.....	15
Herramientas para desarrollo.....	15
Capítulo III: Desarrollo del sistema.....	16
Introducción al capítulo.....	16
Roles en el Equipo.....	16
Planificación.....	17
Diagrama de Gantt.....	17
Arquitectura.....	19
Introducción.....	19
Alternativas.....	19
Requerimientos.....	20
Requerimientos funcionales.....	20
Requerimientos no funcionales.....	20
Plan de SQA.....	21
¿Qué es SQA?.....	21
Estándares generales para gestión del proyecto.....	22
Estándares de documentación técnica.....	22
Estándares para implementación.....	23

Estándares para Manual de Usuario.....	24
Estándares para testing.....	24
Análisis de riesgos.....	25
Riesgos identificados.....	25
Plan de contingencia.....	26
Descripción de Casos de Uso.....	27
Casos de Uso.....	27
Diagrama de Casos de Uso.....	28
Desarrollo del chat y entrenamiento del modelo.....	29
Entrenamiento del modelo.....	29
Funcionalidades.....	32
Capítulo IV: Toma de decisiones.....	36
Introducción al capítulo.....	36
Arquitectura.....	36
Diagrama de Arquitectura.....	37
Herramientas.....	38
Capítulo V: Pruebas, resultados y comparaciones.....	39
Introducción.....	39
Cronograma de trabajo.....	39
Tiempo planificado vs tiempo ejecutado.....	40
Ejecución del Proyecto.....	42
Iteración I: Consolidado de ideas y validado de Mockups.....	42
Propuesta.....	42
Resultados.....	42
Conclusiones.....	42
Iteración II: Relevó de requerimientos.....	44
Propuesta.....	44
Resultados.....	44
Conclusiones.....	44
Iteración III - Funcionalidades.....	45
Propuesta.....	45
Resultados.....	45
Conclusiones.....	45
Iteración IV - Primera validación del sistema.....	46
Propuesta.....	46
Resultados.....	46
Conclusiones.....	46
Iteración V - Segunda validación del sistema.....	47
Propuesta.....	47
Resultados.....	47
Conclusiones.....	47

Capítulo VI: Conclusiones y Trabajo futuro.....	48
Introducción al capítulo.....	48
Devolución del cliente.....	48
Conclusiones del equipo.....	50
Trabajo futuro.....	50
Agradecimientos.....	51
Bibliografía.....	52
Glosario.....	54
Anexo.....	56
Diagrama de Gantt en 2 partes.....	56
Código para crear DataSet.....	58
Código para crear DataSet con respuestas vacías.....	61
Código para separar preguntas de las respuestas.....	65

Organización del documento

Este documento estará dividido en 6 capítulos de los cuales hablaremos brevemente a continuación: primero tenemos un capítulo introductorio donde presentaremos nuestro cliente, el problema que se nos planteó y los objetivos que definimos para solucionar dicha problemática. Luego un segundo capítulo llamado estado del arte en el cual desarrollaremos la investigación de nuevas tecnologías para abordar este proyecto. En el tercer capítulo comenzamos con el desarrollo del sistema en los cuales definimos los roles del equipo, hablamos sobre la arquitectura de nuestra solución y los requerimientos que la misma tendrá. En el siguiente, definiremos las herramientas que utilizaremos para todo el desarrollo. Luego, en el quinto capítulo profundizamos en cada etapa del proyecto y en el sexto les entregamos nuestras conclusiones sobre el trabajo realizado.

Resumen del trabajo

En el presente documento se describe paso a paso el desarrollo de un chatbot basado en inteligencia artificial, diseñado para integrarse al WMS 8 Warehouse management system de la empresa WIS.

Dentro de este proyecto se buscó optimizar la experiencia del usuario, reducir tiempos de atención y ofrecer una herramienta que esté disponible las 24 horas y los 7 días de la semana.

El documento incluye apartados donde se podrá observar un análisis exhaustivo del estado del arte para identificar las mejores herramientas y prácticas, la implementación de un proceso iterativo de entrenamiento y pruebas del modelo de IA. A sí mismo, la generación de data sets personalizados para mejorar la precisión y relevancia de las respuestas.

El resultado final es un sistema que no solo cumple con las expectativas establecidas, sino que también sienta las bases para futuras mejoras y adaptaciones.

Este documento detalla las etapas del proyecto, desde la identificación del problema y los objetivos específicos, hasta las decisiones técnicas, la planificación, resultados obtenidos y conclusiones finales. Además, se incluye un análisis de riesgos, diagramas de arquitectura, casos de uso, y anexos técnicos para complementar la documentación del proceso. Con este trabajo se busca no solo resolver las necesidades actuales del cliente, sino también establecer un marco de referencia para el desarrollo de soluciones similares en otros entornos empresariales.

Capítulo I: Introducción

Descripción del cliente

WIS (Warehouse Information System) es una empresa de software dedicada al mundo de la logística (WIS, 1999).

Dicha empresa cuenta con un abanico amplio de soluciones informáticas para sus diversos clientes que van desde la administración de depósitos y almacenes. Desde la recepción de productos hasta su expedición, ofreciendo también multitud de servicios especializados como lo son: software de gestión para locales, módulo de facturación, tracking de vehículos, administración de trámites con la Dirección Nacional de Aduanas, acceso web, entre otras.

Introducción al problema

Nuestro cliente nos plantea la necesidad de incorporar un chatbot a sus servicios que pueda solucionar consultas recurrentes a sus clientes consumidores. Como consultas simples del manejo de su software hasta consultas más complejas, que ayuden al usuario con los procesos para acceder a alguna parte de su sitio. Esto con la finalidad de ahorrar en tiempo y facilitar una herramienta de consulta disponible en todo momento, para así brindar un servicio más integral y amigable con el usuario.

Objetivo general

Generar un chatbot especializado utilizando inteligencia artificial que responda a las distintas necesidades de información de los usuarios de la empresa de nuestro cliente WIS.

Objetivos específicos

Se plantean dos grandes objetivos específicos:

En primer lugar, lograr que el chatbot responda a las dudas y consultas de los usuarios de los servicios WIS, utilizando RAGS para el desarrollo de la IA.

Y por último, lograr que el chatbot se pueda vincular a la plataforma por medio del uso de API o de una conexión con la base de datos y así el modelo pueda aprender un mayor volumen de información y pueda resolver consultas más complejas.

También el proyecto tendrá como objetivo:

- ❖ Investigar soluciones similares y diferentes que ya existan.
- ❖ Documentar el proceso de trabajo.
- ❖ Aplicar los conocimientos y herramientas adquiridos a lo largo de la carrera.
- ❖ Utilizar tecnologías innovadoras para el desarrollo del proyecto.
- ❖ Generar una arquitectura de trabajo aplicable a diferentes empresas del medio.

Descripción del producto

Se requiere generar un chatbot que responda a las distintas necesidades de nuestro cliente. Para ello se implementará un chatbot que trabaje con inteligencia artificial, el cual tendrá que ser capaz de responder a las consultas tanto como para ayudar a los usuarios con la respuesta a consultas genéricas, o también ayudarlos respondiendo preguntas especializadas. Por ejemplo, preguntas basadas en tickets / issues del sistema de nuestro cliente WIS.

Para realizar esta tarea nuestro modelo se alimenta de RAGS (Generación Aumentada de Recuperación) los cuales tendrán documentación que nos provee nuestro cliente con el fin de que el modelo pueda contestar a las preguntas de los usuarios del sistema basados en la información de dichos manuales.

Por otra parte, el modelo IA obtendrá información directamente de la base de datos de nuestro cliente para así poder responder preguntas y/o resolver tickets. Siendo una forma más especializada de obtener información para resolver problemas activos de los usuarios con relación a sus propios datos.

El resultado del proyecto será capaz de integrarse al sistema de WIS como una funcionalidad habilitada para sus usuarios.

Capítulo II: Estado del arte

Introducción al capítulo

Para encontrar la solución ideal para la problemática planteada y con el fin de utilizar las mejores y más adecuadas herramientas desarrollamos un análisis profundo del mercado actual y de las distintas tecnologías que ya existen.

Investigación

Como primer acercamiento con esta tecnología nos pareció competente indagar en el mercado sobre la relevancia e impacto que tiene y tendrá este tipo de proyecto. Así como también indagar sobre las IA más utilizadas de tipo conversacional.

Estudio de mercado

Previsión del número de asistentes virtuales en uso a nivel mundial de 2019 a 2024 (en miles de millones)

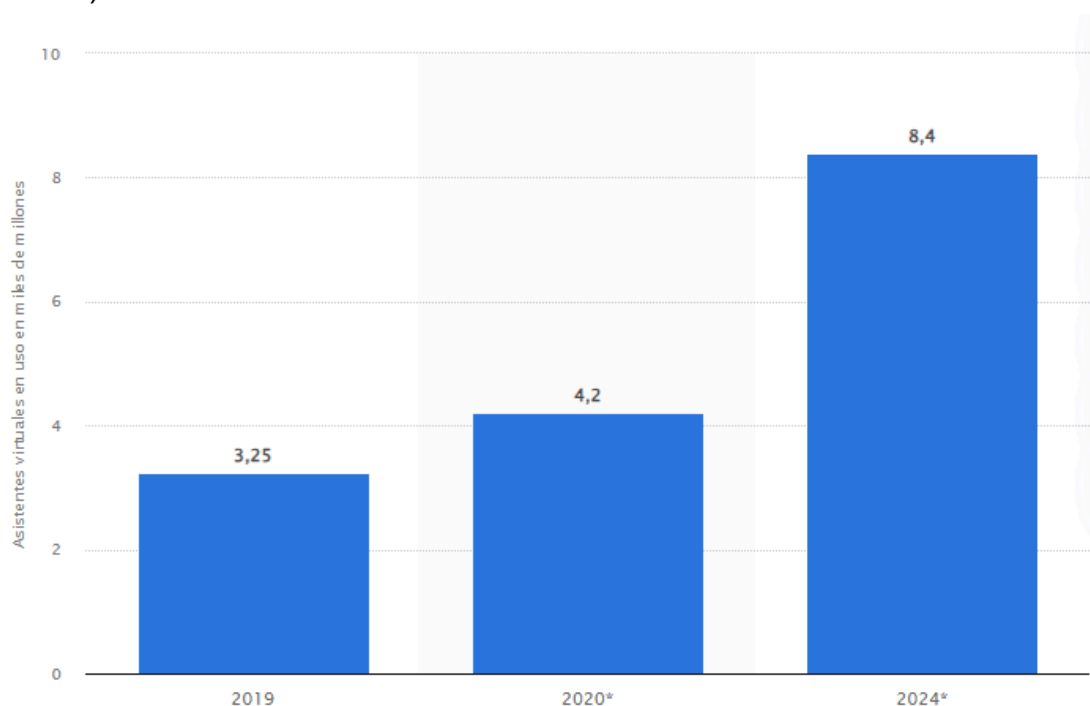


Figura. 1¹

¹ Nota: Tomado de "Asistentes virtuales en uso en el mundo" por Statista, 2023 (<https://es.statista.com/estadisticas/972995/asistentes-virtuales-en-uso-en-el-mundo/>).

Teniendo esto en cuenta, podríamos decir que los usuarios cada vez más harán uso de este tipo de herramientas. También encontramos que esto puede dar multitud de beneficios a las empresas según la investigación de Capgemini de 2019:

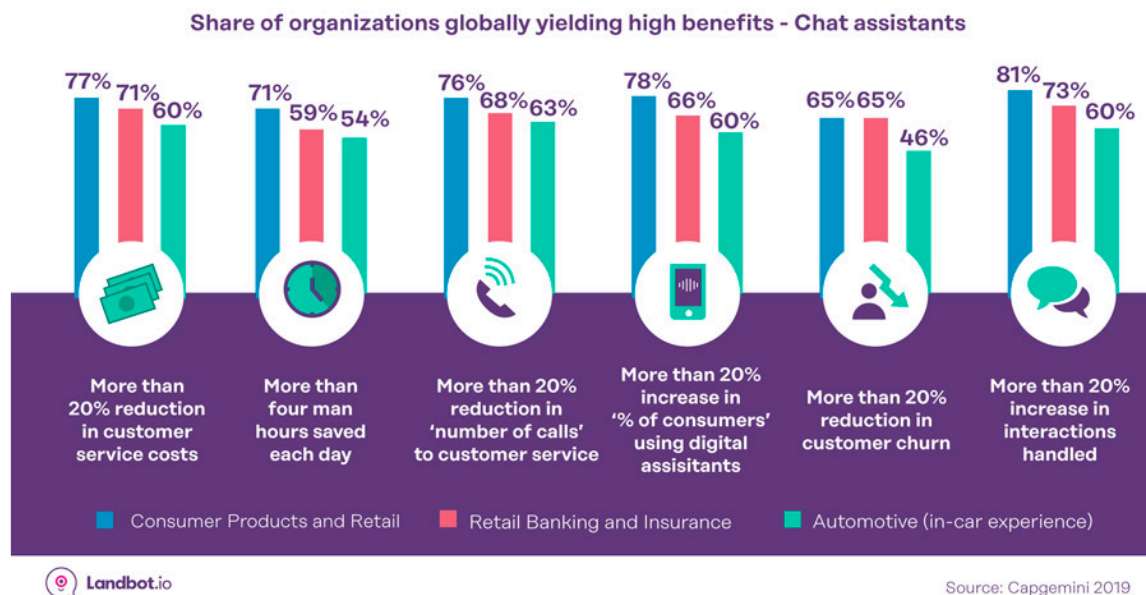


Figura. 2²

Cabe destacar que nuestro cliente es una empresa de logística que trabaja con el consumidor final así como con otras empresas de la industria, viendo así los distintos beneficios que un chatbot le podría ofrecer. Entre ellos podemos mencionar la reducción de costes en servicio al cliente, ahorro de tiempo y menor tasa de abandono de la clientela.

² Nota: Porcentaje de organizaciones a nivel global que obtienen altos beneficios utilizando asistentes de chat. Reimpreso de "Estadísticas IA Conversacional," por Landbot, 2024, <https://landbot.io/es/blog/estadisticas-ia-conversacional>. Copyright 2019 por Capgemini.

Tipos de chatbots existentes en la actualidad

Hay dos grandes formas de clasificar los tipos de chatbots que existen en la actualidad.

Podemos categorizarlos según la forma con la que interactúa con los usuarios y según la tecnología que se utiliza para su desarrollo.

Según su manera de interactuar con los usuarios:

TIPOS DE CHATBOTS SEGÚN SU MANERA DE INTERACTUAR CON EL USUARIO	DESCRIPCIÓN	TIPOS DE NEGOCIO RECOMENDADOS	VENTAJAS
Chatbots de texto	Interactúan con el usuario exclusivamente a través de texto escrito. Se utilizan comúnmente en canales con limitaciones de formato, como los SMS.	Pequeñas empresas, servicio al cliente, soporte técnico.	Fácil implementación, bajo costo, adecuado para comunicaciones simples.
Chatbots de texto enriquecido	Utilizan texto enriquecido con imágenes, emoticonos, gifs, botones interactivos, etc. Mejoran la interacción haciéndola más amena, informal y personalizada.	E-commerce, marketing digital, grandes empresas.	Interacción atractiva y personalizada, mayor engagement del usuario, versátil para diversas plataformas.
Chatbots de voz	Interactúan directamente con el usuario mediante la voz. Son populares en entornos personales, especialmente con asistentes de voz como Siri, Cortana o Alexa.	Domótica, atención al cliente en servicios de telecomunicaciones , sectores con alto uso de tecnología.	Interacción natural y fluida, manos libres, accesible para personas con discapacidad visual.

Figura. 3³

³ Nota: tomado de [aunoa.ai](https://aunoa.ai/blog/que-tipos-de-chatbot-existen/), 2024. (<https://aunoa.ai/blog/que-tipos-de-chatbot-existen/>)

Según la tecnología que se utiliza para su desarrollo:

Tipos de Chatbots según la tecnología utilizada	Descripción	Tipos de negocio recomendados	Ventajas
Chatbots de ITR	No suelen utilizar Inteligencia Artificial. Funcionan con menús y árboles de decisión predefinidos. Son útiles para la autogestión de servicios sin necesidad de atención humana, como reportar incidencias.	Empresas con consultas y tareas repetitivas, servicios públicos, atención al cliente básica.	Fácil implementación, bajo costo, adecuado para resolver tareas sencillas y repetitivas.
Chatbots de aprendizaje automático	Utilizan Inteligencia Artificial, capaces de mantener conversaciones naturales y aprender de la experiencia. Emplean NLP (Procesamiento de Lenguaje Natural) y ML (Machine Learning). Necesitan entrenamiento constante para mejorar y evitar comportamientos indebidos.	Grandes empresas, plataformas tecnológicas, servicios financieros.	Conversaciones naturales, aprendizaje continuo, alta personalización y capacidad de manejar consultas complejas.
Chatbots de reconocimiento de palabras clave	Funcionan identificando palabras clave en el entorno conversacional y proporcionando respuestas programadas. Son útiles como complemento o reemplazo de un buscador tradicional para responder preguntas frecuentes.	Páginas web con secciones de FAQs, empresas con consultas repetitivas, comercio electrónico.	Implementación simple, útil para preguntas frecuentes, mejora la experiencia del usuario en búsquedas específicas.
Chatbots cognitivos	Basados en la Inteligencia Artificial y el Machine Learning, pueden entender el lenguaje natural y las intenciones del usuario, interpretándolas en un contexto más amplio. Permiten cambiar de tema sin perder el hilo de la conversación.	Empresas tecnológicas avanzadas, atención al cliente de alto nivel, sectores con alta variabilidad de consultas.	Alta capacidad de comprensión contextual, manejo de conversaciones complejas, adaptabilidad a cambios de tema.

Figura. 4⁴

⁴ Nota: tomado de aunoa.ai, 2024. (<https://aunoa.ai/blog/que-tipos-de-chatbot-existen/>)

Formas para incluir bases de datos al chatbot

En un principio, debemos tener en cuenta que implementar esta funcionalidad se puede realizar de múltiples formas pero muchas veces depende de qué enfoque se utilice para correr el modelo. Ya que deseamos generar la interacción mediante el chatbot con IA pero utilizando como servidor local LM Studio.

Se observó que el primer caso posible o método para lograr implementar esta funcionalidad es a través del uso de una query integrada con los RAGS o bases de datos. En este método el usuario realiza una consulta y el chatbot busca en la base de datos de vectores y a su vez en la base de datos SQL para devolver una respuesta generada en un único contexto conteniendo información de ambas fuentes. Ejemplos de librerías que se usan para implementar esto son: Pinecone y FAISS.

Otra alternativa en la cual estuvimos indagando centra su idea en la división por contexto. En esta opción se define una lógica que a través de palabras claves o contextos evalúa si la consulta del usuario debe de ser atendida por la base de datos de vectores o por la base de datos SQL.

Por último, se estudió la posibilidad de usar una API intermediaria. En dicho caso, el chatbot estará conectado a una API que se encargará de evaluar si se debe de usar la base de datos de vectores, la base de datos SQL o ambas para generar así una respuesta óptima al usuario. Una posible opción para implementar esto es Groq. Si se quisiera realizar la API desde cero en Python, se puede realizar usando Flask.

Cabe mencionar que en cualquiera de los casos descritos anteriormente se debería de crear un usuario en la base de datos con permisos de solo lectura para asignar el mismo al chatbot de manera que éste sólo pueda realizar consultas y no efectuar modificaciones. De igual forma, se deberá proporcionar visibilidad a las tablas que sean relevantes para realizar consultas sobre ellas.

Cuadro de comparación de tecnologías

En cuanto a las tecnologías que investigamos para el cuadro comparativo evaluamos solo las opciones open source. Discriminamos opciones como OpenAI GPT-4 con Azure, Google Cloud Vertex AI, IBM Watson Assistant, entre otras.

Modelo	Versión	Cant. Parámetros	Tamaño	Acepta RAGS
LLama 3	3.2	3B	6GB	SI
LLama 3	3.2	1B	1.3GB	SI
LLama 3	3.1	8B	16GB	SI
Llama 2	2	7B	14GB	SI
Llama 2	2	13B	26GB	SI
Gemma 2 9B	2	9B	18GB	SI
Gemma 2 2B	2	2B	5GB	SI
Gemma 7B	7B	7B	34GB	SI
GPT-2	XL	1.5B	6GB	NO
Mistral	7B	7B	16GB	SI
Mistral	Nemo	12B	25GB	SI

Figura. 5

En este cuadro podremos apreciar a simple vista los datos de mayor importancia a la hora de comparar modelos de IA. Por un lado, nos importa tener en cuenta la cantidad de parámetros que el modelo acepta ya que cuantos más parámetros acepte más opciones tendrá nuestra IA para generar una respuesta acertada.

Por otra parte, más parámetros se traduce a un mayor tamaño del modelo y mayor consumo de memoria en tiempo de ejecución, lo cual genera una demanda de recursos de hardware importante a tener en cuenta. Y el último dato hace referencia a una tecnología llamada RAG (Generación Aumentada por Recuperación) que es lo que nos permitirá recuperar datos relevantes de una base de datos de vectores y los utiliza para mejorar la precisión de las respuestas para un modelo pre-entrenado.

Uso del modelo y desarrollo del chat

Lo siguiente en la línea de trabajo para desarrollar nuestro chatbot, es definir qué tecnologías necesitamos para efectivamente crear un chat y que éste a futuro pueda ser colocado en un servidor o generar una API capaz de utilizar nuestro modelo a través de peticiones. Para ello utilizaremos una primera instancia en la que nuestro modelo de IA interactúa con una aplicación en python con la capacidad de comunicarse con el front-end de nuestro cliente.

Pretendemos que nuestro chatbot sea un **Producto Mínimo Viable (MVP) y no un producto terminado**, ya que contará con las funcionalidades esenciales para validar hipótesis de negocio y objetivos planteados, recoger retroalimentación y reducir riesgos, permitiendo iterar y mejorar el producto mediante las reuniones con el cliente. Además, **se dispondrá solo de hardware doméstico** y no un servidor dedicado con los recursos necesarios para desarrollar un chatbot lo suficientemente poderoso como para escalar los requerimientos.

Sistemas similares

Existen en el mercado ya servicios similares a los que se desean desarrollar, en el siguiente apartado mencionaremos los más destacados del área:

- ❖ Genexus GPT
- ❖ Microsoft Copilot
- ❖ Google Asistant
- ❖ IBM Watson Discovery
- ❖ Cohere's RAG-as-a-Service
- ❖ OpenAI ChatGPT con Plugins

Conclusiones

Comprendimos luego de investigar, que este proyecto tendrá un gran impacto sobre la empresa de nuestro cliente, aportando así una mejoría considerable al soporte de los usuarios consumidores. Observamos que ya existen tecnologías similares a la que queremos desarrollar, éstas las tendremos en cuenta para llevar a cabo con éxito el desarrollo de nuestro chat de soporte con inteligencia artificial.

Camino a seguir

En principio decidimos utilizar como modelo pre entrenado de IA, Llama versión 3.1 con 8 billones de parámetros ya que es la opción que mejor cubre nuestras necesidades y limitaciones de equipo. Por otra parte, utilizaremos una aplicación usando python para interactuar con dicho modelo y el front-end de nuestro cliente. A su vez, haremos uso de LanceDB que es una base de datos de vectores vinculado al modelo. Por último, incorporaremos una comunicación con la base de datos de nuestro cliente para que este mismo pueda contestar a dudas más específicas de los usuarios.

Herramientas para desarrollo

- ❖ Modelo de inteligencia artificial LLama 3.1, 8B
- ❖ Nomic embed text v1.5
- ❖ LanceDB/Chromadb
- ❖ Visual Studio Code
- ❖ Anything LLM
- ❖ Google Colab
- ❖ Hugging Face
- ❖ UnSloth

Capítulo III: Desarrollo del sistema

Introducción al capítulo

En primeras instancias el cliente nos otorgó diferentes manuales de su empresa para poder trabajar en iteraciones tempranas con una versión primitiva del chatbot y que el mismo pueda generar respuestas de prueba en base a las mismas.

Comenzamos con gran incertidumbre por la naturaleza del proyecto, ya que este mismo implica el uso de nuevas tecnologías en el mercado y que al mismo tiempo pueden ser de gran complejidad. El mayor desafío presentado es abordar un proyecto a gran escala e integrarlo con las tecnologías anteriormente mencionadas.

Roles en el Equipo

A la hora de realizar el trabajo presentamos la correspondiente división de trabajo, sin embargo, es importante destacar que sin importar las diferentes áreas de desempeño, todos los integrantes somos partícipes de todas las tareas relacionadas con el proyecto.

Integrantes	Desempeño	Responsabilidad
Carlos Santana	Desarrollo de documentos, desarrollo back-end y tester funcional	Desarrollo back-end y testeo funcional.
Andres Romano	Desarrollo back-end, front-end y tester funcional.	Desarrollo back-end y front-end.
Juan Pilón	Desarrollo de documentos, desarrollo front-end, back-end	Documentación y desarrollo front-end.
Nicolás Silva	Desarrollo de documentos, planificación, desarrollo back-end, tester funcional y diseñador.	Planificación, documentación y testeo funcional.

Figura. 6

Planificación

Para este apartado decidimos usar una herramienta online y gratuita llamada Agantty para realizar el diagrama de Gantt. La misma nos permitió sin limitaciones y de manera colaborativa ver y editar el mismo diagrama. El mismo tiene un periodo finito de uso, pero de todas maneras fue suficiente para nosotros. A continuación, expondremos una imagen del resultado:

Diagrama de Gantt

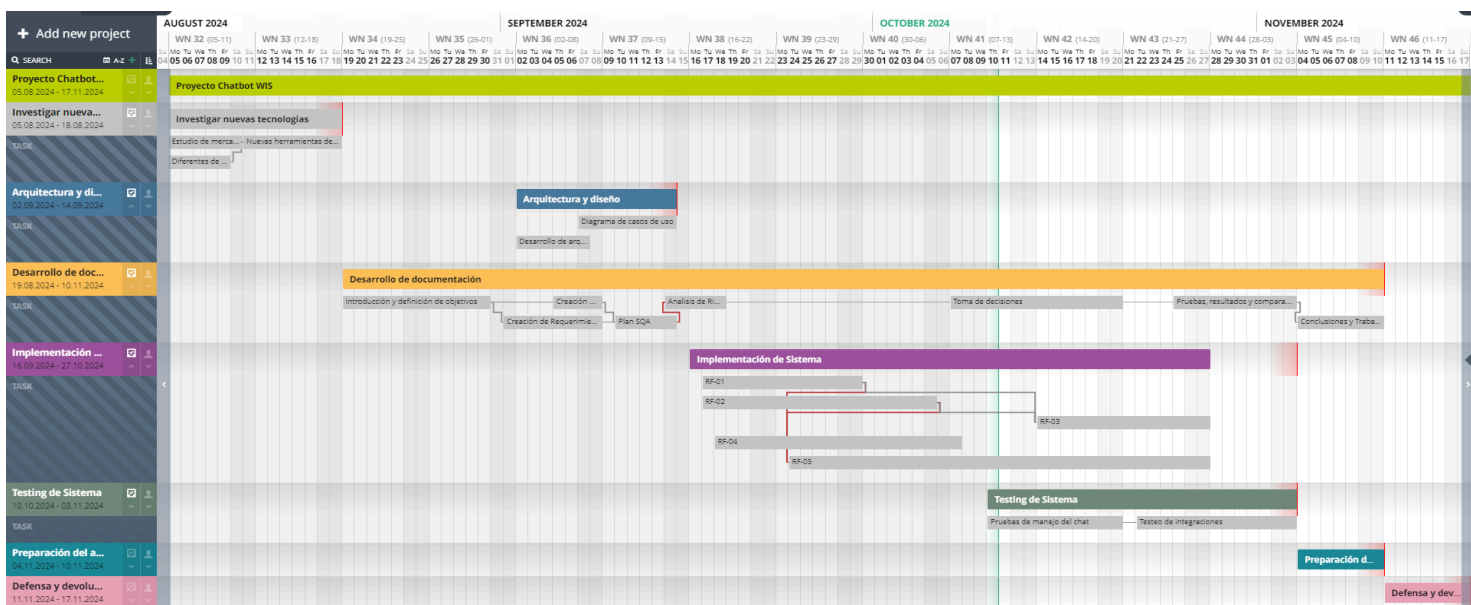


Figura. 7

En el apartado del anexo adjuntamos el mismo diagrama en 2 partes para mejorar su calidad y se puedan apreciar mejor los detalles del mismo.

Agantty también posee un dashboard interno que nos organiza en forma de lista las tareas por hacer que incluye el tiempo que nos queda o el tiempo que se excede una tarea de su fecha límite.

✓	TASK	PROJECT / TEAM	USER / DATE
	Desarrollo de documentación 19.08.24 - 10.11.24		
<input type="checkbox"/>	Toma de decisiones	Desarrollo de documentación WIS	Mo, 07.10 - 9:00 Su, 20.10 - 18:00 10 days left
<input type="checkbox"/>	Pruebas, resultados y comparaciones	Desarrollo de documentación WIS	Fr, 25.10 - 9:00 Su, 03.11 - 18:00 24 days left
<input type="checkbox"/>	Conclusiones y Trabajo futuro	Desarrollo de documentación WIS	Mo, 04.11 - 9:00 Su, 10.11 - 18:00 31 days left
	Implementación de Sistema 16.09.24 - 27.10.24		
<input type="checkbox"/>	RF-04	Implementación de Sistema WIS	We, 18.09 - 9:00 Mo, 07.10 - 18:00 3 days overdue
<input type="checkbox"/>	RF-05	Implementación de Sistema WIS	Tu, 24.09 - 9:00 Su, 27.10 - 18:00 17 days left
<input type="checkbox"/>	RF-03	Implementación de Sistema WIS	Mo, 14.10 - 9:00 Su, 27.10 - 18:00 17 days left
	Testing de Sistema 10.10.24 - 03.11.24		
<input type="checkbox"/>	Pruebas de manejo del chat	Testing de Sistema WIS	Th, 10.10 - 9:00 Su, 20.10 - 18:00 10 days left
<input type="checkbox"/>	Testeo de integraciones	Testing de Sistema WIS	Tu, 22.10 - 9:00 Su, 03.11 - 18:00 24 days left

Figura. 8

Arquitectura

Introducción

Se pretende establecer en esta sección las posibles arquitecturas que podríamos llegar a utilizar para nuestro proyecto, teniendo en cuenta las diferentes necesidades de nuestro cliente. También tuvimos presente que nuestro objetivo principal será el crear un chatbot que cumpla con ciertos requerimientos conectado a diferentes servicios, por lo tanto nuestro foco de atención se va a basar en el desarrollo del chat y no de los sistemas y servicios que lo rodearán.

Alternativas

Cuando estuvimos investigando manejamos diferentes opciones para implementar los requerimientos del cliente. Decidimos implementar una arquitectura monolítica teniendo solo un componente principal. Por otro lado, en el mercado existen diferentes arquitecturas para la implementación de chatbots. Muchas de las opciones más populares dejan como ventaja una rápida implementación de un LLM pero son poco personalizables, lo cual nos deja con poco control sobre los cambios y adaptaciones necesarias para nuestro modelo. Herramientas de esta índole podemos nombrar chat GPTBuilder, Zapier, Make, Botpress, VoiceFlow, StackAI entre muchas otras.

Otro indicador importante a la hora de decidir sobre otras opciones fue la capacidad de las herramientas para aceptar diferentes LLM. Muchas de estas tienen predeterminadas los LLMs que puedes usar y eso fue un gran punto de inflexión entre cual elegir ya que necesitábamos un modelo que no solo se pudiera modificar, si no que sea 100% gratuito.

Requerimientos

Requerimientos funcionales

- **RF-01:** el proyecto debe ser capaz de responder preguntas de los usuarios con respecto al uso de los servicios de wis.
- **RF-02:** el chatbot debe guiar al usuario paso a paso en procesos clave de la empresa de nuestro cliente (ej: creación de informes).
- **RF-03:** el proyecto debe estar conectado a wis a través de una base de datos para obtener información actualizada.
- **RF-04:** se deben poder regenerar las respuestas dadas por el sistema.
- **RF-05:** el sistema debe de poder integrarse en la plataforma de nuestro cliente.

Requerimientos no funcionales

- **RNF-01:** El sistema debe de ser intuitivo y de fácil aprendizaje. Un usuario promedio deberá poder usar el chatbot de la empresa en menos de 5 minutos de investigación.
- **RNF-02 :** el chatbot deberá ser funcional y estar disponible cuando el usuario decida hacer cualquier consulta sobre la empresa, con excepción de los momentos en que el chat o la aplicación del cliente esté en mantenimiento.
- **RNF-03 :** obtener respuestas a las preguntas y/o consultas del usuario en menos de 20 segundos en el 90% de los casos .
- **RNF-04 :** El sistema debe ser compatible con múltiples browsers, priorizando su uso para Google Chrome, FireFox y Opera.

Plan de SQA

Es de vital importancia la correcta implementación de un plan SQA robusto para asegurar los estándares de calidad deseados para el proyecto.

Será la totalidad del grupo de trabajo quienes monitorean las distintas partes del proceso de desarrollo para garantizar la calidad tanto en la documentación, la implementación y el testeo de la misma.

¿Qué es SQA?

“SQA (Software Quality Assurance o Aseguramiento de la Calidad del Software) implica revisar y auditar los productos y actividades de software para verificar que se cumplen los procedimientos y los estándares, además de proveer a las gerencias apropiadas (incluyendo a la de proyectos) con los resultados de estas revisiones. Por lo tanto, SQA envuelve al PROCESO de desarrollo de software completo: monitoreando y mejorando el proceso; asegurándose que cualquier estándar y procedimientos adoptados sean seguidos; y, asegurándose que los problemas sean encontrados y tratados.”⁵

⁵ NOTA: Fragmento de texto tomado de Findingtc
(<http://findingtc.com/sqa-aseguramiento-de-la-calidad-del-software>)

Estándares generales para gestión del proyecto

Para estándares de gestión de proyecto optamos por usar las siguientes metodologías y/o herramientas:

- **PMBOK (Project Management Body of Knowledge)** - El PMBOK es un conjunto de estándares y buenas prácticas para la gestión de proyectos publicado por el Project Management Institute (PMI). Define cinco grupos de procesos (inicio, planificación, ejecución, monitoreo y control, y cierre) que cubren todas las fases de un proyecto. Es una referencia fundamental para la gestión profesional de proyectos.
- **SCRUM⁶** - En nuestro proyecto usaremos SCRUM como metodología de trabajo para el control y gestión del mismo, donde usar una metodología de este estilo cobra mucho peso al trabajar de manera dinámica e iterativa. Esto es sumamente esencial, ya que en un proyecto como el nuestro, debemos ser conscientes donde el cambio y la adaptación es la prioridad número uno. Al trabajar con un cliente, los requerimientos pueden cambiar o evolucionar sobre la marcha, lo que nos permite adaptar el camino que estamos tomando, para lograr los objetivos propuestos.
Siguiendo esta metodología dividimos nuestro trabajo en 5 fases:
Inicio, Planificación y Estimación, Implementación, Revisión y Retrospectiva y Lanzamiento.
- Sumado a esto, para nuestra gestión, usamos las herramientas anteriormente mencionadas, como Agantty para generar un diagrama de gantt con tiempos fijados y Excel para el cronograma de trabajo inicial. También definimos utilizar discord como herramienta para la comunicación y como espacio de trabajo.

Estándares de documentación técnica

Se utilizaron normas APA7 para el desarrollo del documento y una extensión de los navegadores llamada "MyBibCitation Generator" que utiliza normas APA para extraer las diferentes fuentes de información utilizadas que se encuentran en internet.

⁶ SCRUM: "es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto." "Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales." Extraído de (<https://proyectosagiles.org/que-es-scrum/>).

Estándares para implementación

- **Refinar-** Para la gestión del proyecto debemos tener en cuenta las dos principales partes que debemos abordar de nuestro asistente de chat. En primera instancia, debemos ser capaces de entregar un producto que sea práctico de usar, intuitivo el cual sea capaz de responder preguntas espontáneas y específicas sobre el sistema WIS. Para esto el asistente debe estar entrenado con cierto grado de lo que llamamos temperatura a través de varias iteraciones sobre la documentación que fue proporcionada. Siendo este el caso debemos asegurarnos de que el asistente sea capaz de tener coherencia en sus respuestas, que deben estar ubicadas bajo el contexto necesario para su entendimiento. Y en segunda instancia, el asistente debe ser capaz de responder y asistir para brindar soluciones a problemas que posea el cliente, como puede ser a través de un sistema de tickets , o también debe ser capaz de responder a preguntas específicas que estén relacionadas directamente con la información dinámica e interna del sistema WIS, ya sea, datos sobre una carga de un producto, asistencia respuesta de un ticket, entre otras cosas.
- **Repositorio de LLM-** Utilizamos Huggingface para descargar el LLM base de nuestro proyecto y a su vez luego de crear nuestro dataset y entrenar el modelo. Lo subimos al mismo repositorio para poder utilizarlo y poder descargarlo si se necesitara.
- **Control de Versiones-** A la hora de entrenar nuestro modelo y actualizar su versión usamos un código numérico al final del nombre del archivo indicando el mismo. Dicho archivo estará compuesto primero por el nombre del modelo original de LLM usado, en este caso LLama 3.1, el nombre de nuestro cliente que será WIS y al final la versión del mismo. El nombre de nuestros archivos entonces tendrán un formato similar a: "LLama3.1WISv2.0"

Estándares para Manual de Usuario

- Para los estándares de uso del usuario, debemos tener en cuenta que el uso de este tipo de herramientas IA no sigue un patrón específico establecido. Sin embargo, el usuario debe saber de antemano que el uso incorrecto del asistente puede arrojar datos incorrectos.
- También se debe saber que el asistente está entrenado específicamente para el uso dentro del sistema WIS y que cualquier pregunta por fuera de eso, puede ser poco acertado.
- Otra cosa que cabe mencionar es que el modelo puede equivocarse y que ninguna información debe tomarse como 100% verídica.

Bajos estos estándares, el usuario debería ser capaz de utilizar el asistente de manera bastante práctica e intuitiva.

Estándares para testing

Para el estándar de testing, usaremos un sistema de pulido en base a machine learning con un script y con herramientas de HuggingFace, para entrenar por iteraciones a nuestro asistente, el cual nos permitirá tener cierto estándar al momento de testear. A su vez, el estándar de testing se basará en sesiones de preguntas y respuestas con el mismo, donde se medirá el nivel de asertividad en cada respuesta junto a el nivel de coherencia y fidelidad en la misma. Este mismo procedimiento se realizará cada vez que se dé el lanzamiento de una nueva versión del modelo para asegurar la mejora en los cambios entre versiones.

Análisis de riesgos

Nuestro proyecto centra parte importante de su complejidad en la investigación e incorporación de nuevas tecnologías emergentes en el mercado lo cual genera cierto grado de incertidumbre sobre la fiabilidad de las herramientas, así como también del tiempo que llevará adquirir los conocimientos necesarios para su uso.

También destacamos los factores de riesgo externos a nuestro desarrollo que serán claves para lograr la implementación deseada por nuestro cliente. A continuación, detallaremos los riesgos con su correspondiente plan de contingencia:

Riesgos identificados

Riesgo 1- El proyecto es bastante ambicioso y puede estar por fuera de los recursos y tiempo disponible.

Riesgo 2- Utilización de tecnologías recientes y/o complejas.

Riesgo 3- Uso de recursos(CPU,GPU,RAM,etc) físicos para poder entrenar al asistente por fuera de lo que podemos disponer.

Riesgo 4- El asistente puede no quedar entrenado correctamente y no otorgar el producto esperado. El mismo puede proporcionar respuestas incorrectas, incompletas o poco claras, lo que puede afectar negativamente la experiencia del usuario.

Riesgo 5- El asistente puede ser comprometido en brechas de seguridad, ya sea externas o propias del mismo, comprometiendo el sistema.

Riesgo 6- El chatbot puede no integrarse correctamente al sistema que ya utiliza nuestro cliente.

Riesgo 7- Problemas de rendimiento a la hora de que los usuarios interactúen con el modelo.

Plan de contingencia

Luego de analizar los diferentes riesgos que conlleva llevar a cabo nuestro proyecto ideamos los siguientes planes de contingencia:

Contingencia 1- Tendremos que ser precavidos con la organización de nuestros tiempos como equipo y estar en constante comunicación entre nosotros y con nuestro cliente para poder lograr los objetivos planteados. También se planteó una división de trabajos en la que se prioriza el correcto funcionamiento del chatbot sobre nuevas funcionalidades. En caso de necesitar mucho tiempo para el desarrollo del chatbot se dejará para un futuro el trabajo para acceso de datos del modelo a una base de datos.

Contingencia 2- Se le dedicará una porción importante de tiempo disponible para indagar e investigar sobre las nuevas tecnologías y herramientas de desarrollo.

Contingencia 3- Utilizaremos en su mayoría recursos gratuitos pero alojados en la nube para mitigar este riesgo como lo es GoogleColab.

Contingencia 4- Para asegurar un correcto funcionamiento del modelo se implementarán las fases de testing mencionadas anteriormente.

Contingencia 5- Se usarán set de datos de la empresa que están destinados a ayudar a los diferentes usuarios sin comprometer los datos privados de nuestro cliente. A su vez, cuando el LLM consuma información de la base de datos no lo hará directamente para no exponer los datos sensibles a cualquier usuario.

Contingencia 6- Necesitaremos realizar un análisis previo de los sistemas de nuestro cliente para prevenir este riesgo.

Contingencia 7- Generar un sistema de fácil acceso para los usuarios del servicio y contar un sistema escalable.

Descripción de Casos de Uso

Casos de Uso

Para los casos de uso tenemos un flujo diferente a lo que estaríamos acostumbrados normalmente en el ámbito tecnológico. En este caso, crearemos diagramas de casos de uso en los cuales tendremos interacciones básicas entre un usuario y nuestro asistente / sistema. En las interacciones nos interesa saber que si el input es sobre una pregunta sobre lo que fue entrenado, nos interesa que el output esperado es el correcto, con el contexto correcto y que posea un alto nivel de coherencia. De la misma manera, si el input es incorrecto y no relacionado, nos interesa, que pueda dar una respuesta, pero no exactamente tiene que ser una respuesta correcta, ya que no fue entrenado en una pregunta no relacionada.

Nombre	CU1- Soporte y atención al cliente
Actores	Usuario/Cliente
Sinopsis	El chatbot tendrá que ser capaz de solucionar las incertidumbres de los usuarios que utilicen el sistema de WIS. Cumpliendo así con el rol de asistente personal para satisfacer las necesidades de los mismos.

Nombre	CU2- Gestión y seguimiento de pedidos
Actores	Usuario/Cliente
Sinopsis	El chatbot tendrá que ser capaz de conocer los datos del estado del inventario y la logística del mismo para poder solucionar consultas sobre esos datos.

Nombre	CU3- Soporte y asistencia técnica
Actores	Usuario/Cliente
Sinopsis	El chatbot podrá asistir con información relevante cuando los usuarios no tengan los conocimientos necesarios sobre alguna funcionalidad del sistema de WIS.

Nombre	CU4- Respuesta del sistema a preguntas no relacionadas
Actores	Usuario/Cliente
Sinopsis	El chatbot podrá asistir con información acotada a otras consultas de los usuarios pero estará restringido para contestar lo mínimo sin relación al sistema de WIS.

Diagrama de Casos de Uso

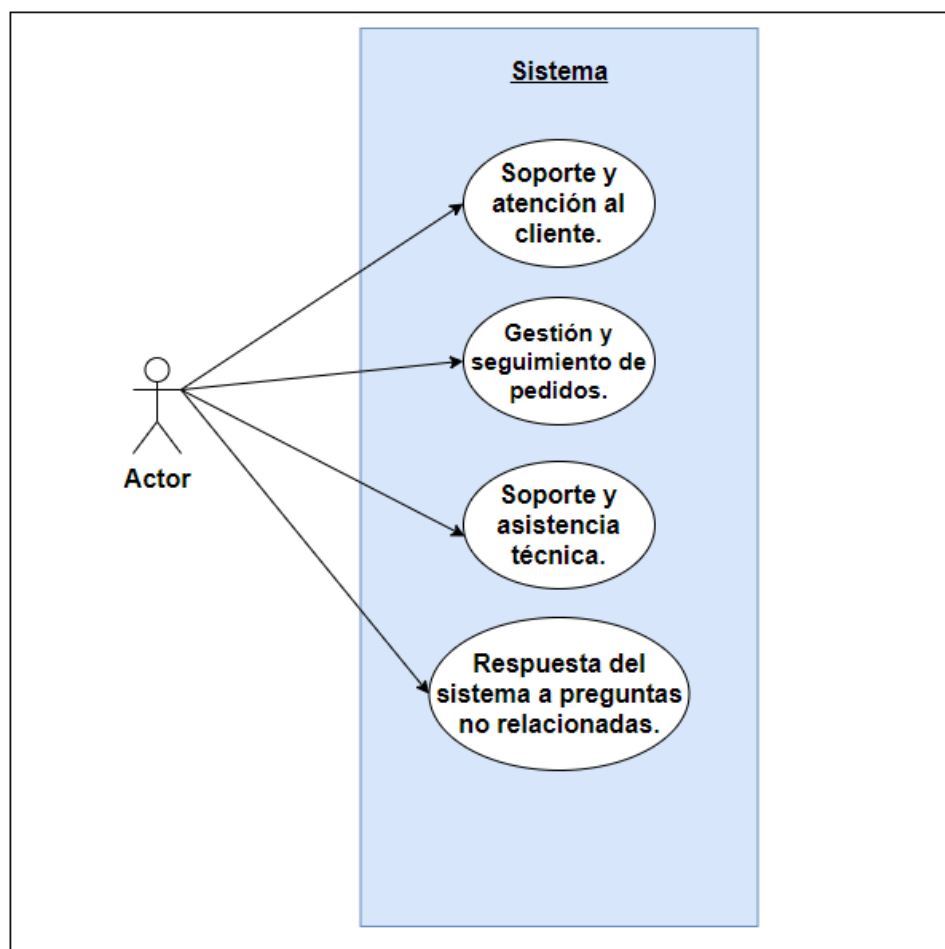


Figura. 9

Desarrollo del chat y entrenamiento del modelo

Entrenamiento del modelo

En un inicio se propuso usar un modelo solo con la implementación de RAGS. Esto supuso grandes limitantes en cuanto a la calidad de las respuestas que recibimos del modelo. Primero intentamos cambiar la base de datos de vectores de LanceDB a ChromaDB, esto mejoró un poco la calidad de las mismas respuestas pero no llegamos a un resultado satisfactorio.

Entonces, se propuso re entrenar a un LLM, en este caso probamos con Llama 3.1 con 8b y cuantificado en q8. Para esto creamos un dataset en base al documento más pequeño que nos otorgó nuestro cliente, para ello creamos un script utilizando python el cual toma el índice del documento para crear las preguntas del usuario y el número de página para buscar la información y agregarla como respuesta de sistema. Se detalla en el anexo, el código correspondiente. Para hacer fine tuning (ajuste fino) se utilizó la herramienta de Google Colab y un código proporcionado por UnSloth (empresa start up de IA) que contiene las librerías necesarias para optimizar el fine tuning. Se realizaron modificaciones al código proporcionado para ajustarlo a nuestro dataset que previamente tuvimos que alojar en Hugging Face. Luego de re entrenar el modelo utilizando las funciones gratuitas de Google Colab (Por ej: gráfica nvidia t4) al obtener una pérdida baja en el entrenamiento y luego de enfrentar múltiples bajas por límite de tiempo. Logramos subir el nuevo modelo a Hugging Face. Para finalizar, descargamos el modelo utilizando LM Studio y usamos la función de crear servidor para desde nuestro código en python del chatbot poder interactuar con el nuevo modelo y hacer los tests correspondientes.

El modelo se comportó de forma esperada aunque notamos que el formato del texto no era el deseado y proporcionaba más información de la que era necesaria por respuesta.

Adjuntamos la imagen del resultado en ese momento:

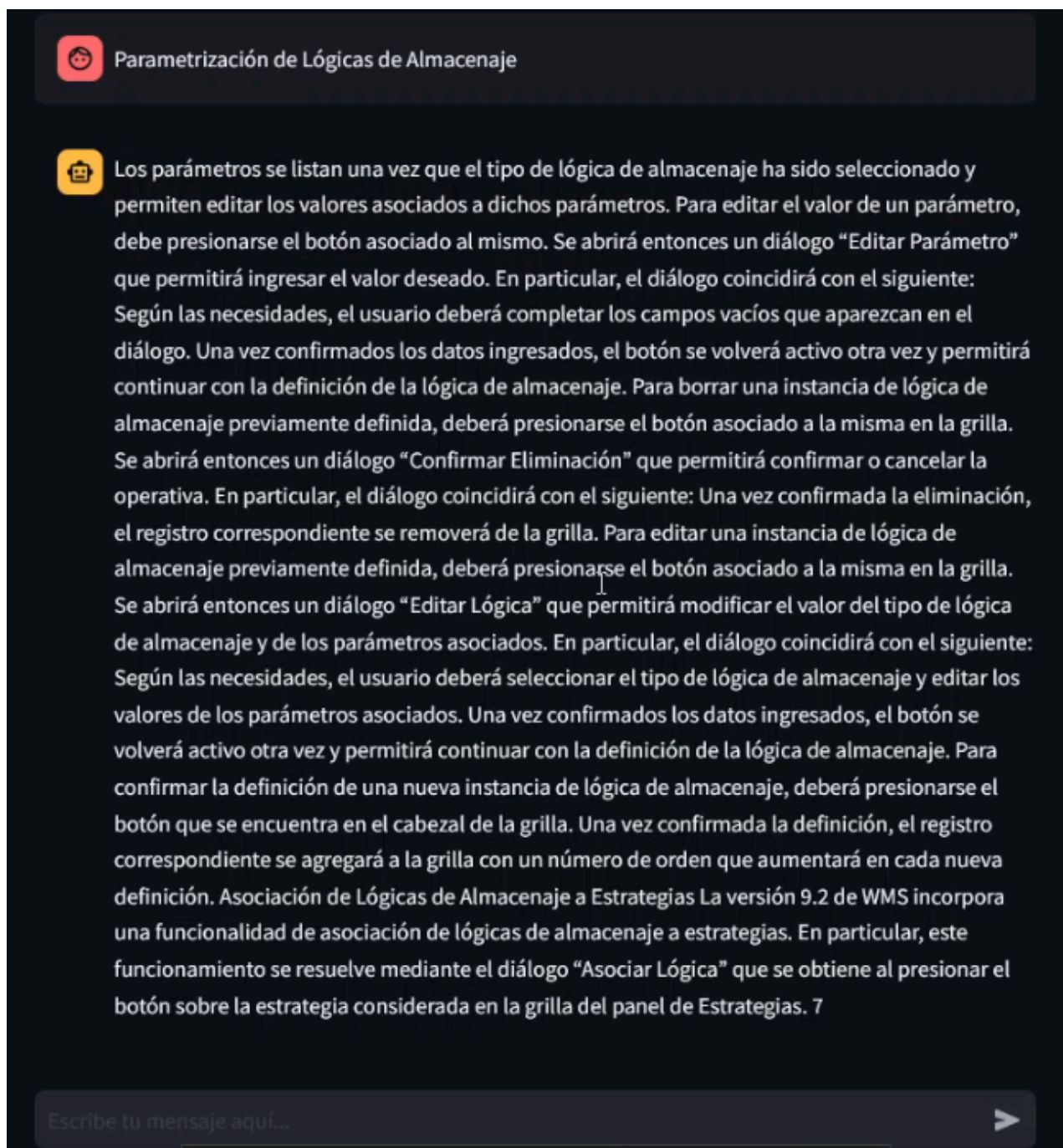


Figura. 10

En la imagen se puede apreciar que la información tiene etiquetas no deseadas, también que su formato está compacto y no respeta los saltos de línea. A su vez la información dada no termina de estar bien detallada.

El siguiente paso fue, replicar la metodología anterior para el resto de documentos, subir el nuevo dataset a hugging face y re entrenar el modelo. El primer dataset que usamos estaba

construido en base a un documento de 40 páginas. En esta fase el nuevo dataset agrega la información de tres documentos extra de más de 200 páginas. Debido al nuevo volumen del dataset el proceso de fine tuning fue más laborioso y no logramos igualar el nivel de perdida anterior al entrenar el modelo. Como resultado las respuestas fueron menos precisas y aún no se logra corregir los problemas de formato.

Para mejorar la asertividad de los dataset modificamos el script para que genere las preguntas del usuario en base al índice pero las respuestas del sistema las deja en blanco para completarlas manualmente. Lo que desencadenó un proceso largo y tedioso que congeló el avance del proyecto hasta finalizada esa etapa. Fue necesario contratar el servicio de Google Colab Pro para acceder una gráfica más potente (nvidia A100) y tuvimos que manejar un límite de 100 unidades de procesamiento de las cuales gasta unas 8.4 por hora para entrenar los modelos con los que fuimos trabajando desde este punto. También revisamos el código utilizado para entrenar el modelo logrando mejorar los resultados de los entrenamientos y reduciendo el tiempo del mismo. Sin embargo, el modelo contestaba bien las preguntas predefinidas pero no manejaba bien las variantes de las mismas debido a que las preguntas estaban sacadas de los títulos de los índices lo cual era muy directo y poco natural. El modelo perdió en este punto naturalidad y capacidad de respuesta.

Para solventar los problemas anteriormente mencionados decidimos agregar al dataset cuatro variaciones en lenguaje natural para cada pregunta que deberían tener la misma respuesta de sistema. Para ello creamos un script de python que separa todas las preguntas del usuario y todas las respuestas del sistema en dos archivos Json separados. Luego utilizamos inteligencia artificial para generar las variantes de las preguntas. Obteniendo un dataset mucho más robusto. Procedimos a entrenar el modelo con el nuevo dataset y los recursos que quedaban disponibles en Google Colab Pro. Notamos una mejora en las respuestas del sistema al interactuar en el chat de LM Studio pero encontrando una disonancia con las respuestas que devuelve nuestro chatbot. Para solventarlo eliminamos todo lo relacionado a RAGS de nuestro código y así obtuvimos paridad en las respuestas de ambos entornos.

Para mejorar aún más la precisión de las respuestas decidimos insistir en la implementación de RAGS debido a que su función es mejorar los resultados y complementarse con el fine tuning. Para mejorar la calidad de la base de datos de vectores creamos un script en python que utiliza el dataset para extraer todas las respuestas del sistema generando un archivo .txt que solo contenga las respuestas de sistema lo que es equivalente a todos los documentos entregados hasta el momento por el cliente. Junto con esto modificamos nuestro código en python para que respete los saltos de línea y formato deseados además de lograr implementar la funcionalidad de re generar respuesta mediante un botón.

Funcionalidades

Nuestro chatbot tiene un entorno visual minimalista para resaltar las funciones principales del mismo. Detallamos en la siguiente imagen como se ve en primera instancia nuestro chat sin declarar aún ninguna pregunta o diálogo:



Figura. 11

En la sección de “escribe tu mensaje aquí..” Se pretende que el usuario pueda solventar mediante texto consultas relevantes con el software del WMS o que estén relacionadas con el funcionamiento de los procesos internos del mismo.

Por las limitaciones del modelo decidimos añadir un botón que permita rehacer alguna pregunta que al usuario no le convenza o no termine de entender. Es importante que se tenga en cuenta que al ser un modelo cuantificado el que se entrenó y se está usando existe un margen de error en las respuestas esperadas así como una demora de promedio de entre 10 a 20 segundos por respuesta existiendo excepciones en las que puede alcanzar hasta 60 segundos en generar la respuesta dependiendo del volumen de su búsqueda y de los datos que deba manejar para cotejar un respuesta.


En las siguientes imágenes se muestra este caso:




Figura. 12


A continuación se muestra la respuesta luego de presionar el botón de rehacer:

Asistente virtual WIS

 quiero saber sobre el registro administrativo de usuarios



Registro Administrativo de Usuarios:
Los administradores pueden registrar nuevos usuarios desde la aplicación Authorization Server. Para esto deben dirigirse a la pantalla “Usuarios”, presionando el enlace correspondiente desde el cabezal de la página.
Una vez dentro de esta página se muestra una tabla con información de todos los usuarios:
Para crear un nuevo usuario, los administradores deben acceder al enlace 'Registrar usuario', que aparece en la parte superior izquierda de la página.
Al ingresar a este enlace, se redirige a otra pantalla en la que se muestra el siguiente formulario:
En este formulario se deben completar los campos Username, Nombre, Email e Idioma con la información correspondiente al nuevo usuario.
El campo Tipo indica qué tipo de usuario se está creando: Internal o External. Internal son usuarios que pertenecen a la organización (por ejemplo administrativos u operarios) que pueden acceder a las aplicaciones configuradas en el Authorization Server. External son usuarios externos a la organización (por ejemplo clientes o proveedores) que pueden acceder únicamente a las aplicaciones configuradas en el Authorization Server que admitan este tipo de usuarios.
Finalmente se debe ingresar la contraseña del nuevo usuario en los campos Nueva contraseña y Confirmar contraseña. Si se marca la casilla Realizar validaciones sobre la contraseña, se comprobará que la contraseña elegida cumpla con ciertos requisitos de seguridad. En caso contrario, no se validarán esos requisitos.
Para confirmar la creación del nuevo usuario, se debe presionar el botón 'Registrar'.




Escribe tu mensaje aquí... 

Figura. 13

También adjuntamos la información en la cual el modelo se está apoyando para contestar, en otras palabras parte del contenido del dataset:

"Registro Administrativo de Usuarios:

Los administradores pueden registrar nuevos usuarios desde la aplicación Authorization Server. Para esto deben de dirigirse a la pantalla "Usuarios" presionando el enlace correspondiente desde el cabezal de la página:

Una vez dentro de esta página se muestra una tabla con información de todos los usuarios:

Para crear un nuevo usuario se debe acceder al enlace Registrar usuario que aparece en la parte superior izquierda de la página:

Al ingresar al mismo se redirige a una nueva pantalla en la que se muestra el siguiente formulario:

En este formulario se deben completar los campos Username, Nombre, Email e Idioma con los datos correspondientes al usuario que se está creando.

El campo Tipo corresponde al tipo de usuario, este puede tomar los valores que se detallan a continuación:

Internal: son usuarios que pertenecen a la organización (por ejemplo administrativos u operarios) que pueden acceder a las aplicaciones configuradas en el Authorization Server.

External: son usuarios externos a la organización (por ejemplo clientes o proveedores) que pueden acceder únicamente a las aplicaciones configuradas en el Authorization Server que admitan este tipo de usuarios.

Administrador: son usuarios que pueden acceder a todo el contenido administrativo del Authorization Server como son la configuración de aplicaciones y usuarios.

Finalmente se debe ingresar la contraseña del nuevo usuario en los campos Nueva contraseña y Confirmar contraseña. Si se marca la casilla Realizar validaciones sobre la contraseña se comprobará que la contraseña elegida cumpla con ciertos requisitos de seguridad, en caso contrario no se validarán esos requisitos.

Para confirmar la creación del nuevo usuario se debe presionar el botón 'Registrar'."

Capítulo IV: Toma de decisiones

Introducción al capítulo

Dada la novedad de las herramientas y la enorme cantidad de posibilidades que existen para solucionar el problema (lo que generó mucha incertidumbre), debemos balancear la relación entre investigación y experimentación para que al final podamos elegir la solución más óptima. Lo anterior descrito influye directamente en la arquitectura de la solución y la asertividad de los resultados; Por lo tanto en este capítulo se describen y justifican las decisiones tomadas en cuanto a Arquitectura y Herramientas utilizadas.

Arquitectura

Dada la naturaleza de nuestro proyecto decidimos que la arquitectura deberá tener en un inicio dos grandes componentes:

- Uno que gestione el modelo elegido y otro que interactúe con el usuario y la base de datos de vectores.
- Para la segunda iteración se utilizaron herramientas de procesamiento en la nube para reentrenar el modelo, fue utilizada la documentación del cliente como dataset para asegurar la asertividad de las respuestas, esto fue así dado que la implementación de RAGS no daba los resultados esperados.

De igual forma a la primera iteración, el modelo re-entrenado es gestionado con las mismas herramientas (LM Studio, código de python para el chatbot, etc); En el siguiente punto se describen a detalle los elementos de la arquitectura implementada.

Diagrama de Arquitectura

A continuación, se presenta un diagrama de la arquitectura del sistema que ilustra las relaciones y flujos de comunicación entre los distintos componentes.

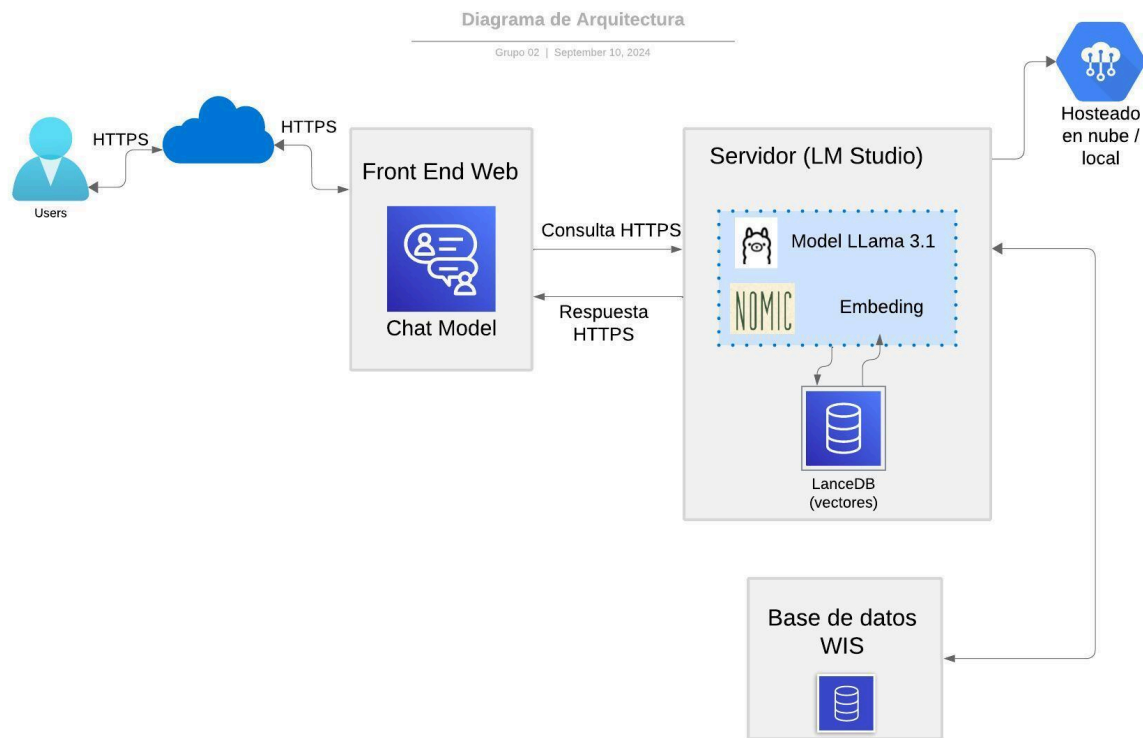


Figura. 11

Front end WEB: Código de Python “**app.py**” con la librería Streamlit para facilitar la implementación de un chatbot, “**model.py**” se comunica con el servidor iniciado en LM Studio que contiene los modelos Llama 3.1 y Nomic Embed Text 1.5, además crea una base de datos de vectores en **LanceDB** con la documentación del cliente. Se define el **prompt**⁷ con el que el chatbot contesta las preguntas y la **temperatura**⁸.

En resumen, la **app.py** se comunica con **model.py** y este último gestiona todo lo anteriormente mencionado.

Servidor: LM Studio para alojar los modelos: Llama 3.1 8B y Nomic Embed Text 1.5, además facilita la creación de un servidor para enviar y recibir consultas.

⁷ **Prompt:** Instrucciones para que un modelo genere una respuesta.

⁸ **Temperatura:** Parámetro que gestiona la “libertad” que tiene un modelo para contestarle a un usuario.

Herramientas

- Python 3.12.6 con las librerías: streamlit, lancedb, PyMuPDF, requests, os, logging, fitz.
- LM Studio: Para gestionar los modelos y levantar el servidor.
- Llama 3.1: Modelo elegido.
- Nomic Embed Text 1.5: Modelo codificador de texto; Propósito: insertar textos como documentos de un dataset.
- LanceDB/ ChromaDB: Base de datos de vectores.
- Huggingface: Plataforma para guardar/descargar modelos y datasets, los modelos s nodos en esa plataforma pueden ser descargados en LM Studio.
- Google Colab: herramienta para re entrenar modelos mediante cloud computing, utiliza huggingface para descargar un modelo y subirlo luego de re entrenarlo.

Cabe destacar que las herramientas que se describen anteriormente son gratuitas y de uso libre.

Capítulo V: Pruebas, resultados y comparaciones

Introducción

En este apartado detallaremos el cronograma inicial de trabajo, así como las diferentes etapas de desarrollo de nuestro proyecto. Utilizamos SCRUM como metodología ágil de trabajo como ya se mencionó anteriormente dividiendo en cinco iteraciones nuestra propuesta.

Cronograma de trabajo



Figura. 12

Tiempo planificado vs tiempo ejecutado

Adjuntamos el diagrama que compara los tiempos estimados en la planificación versus los tiempos que nos tomó llevar a cabo este proyecto. En él se reflejan los puntos en los cuales se invirtieron más horas de lo esperado. A su vez, deja ver de forma gráfica las rectificaciones que llevó a cabo el grupo para intentar cumplir con las metas establecidas.

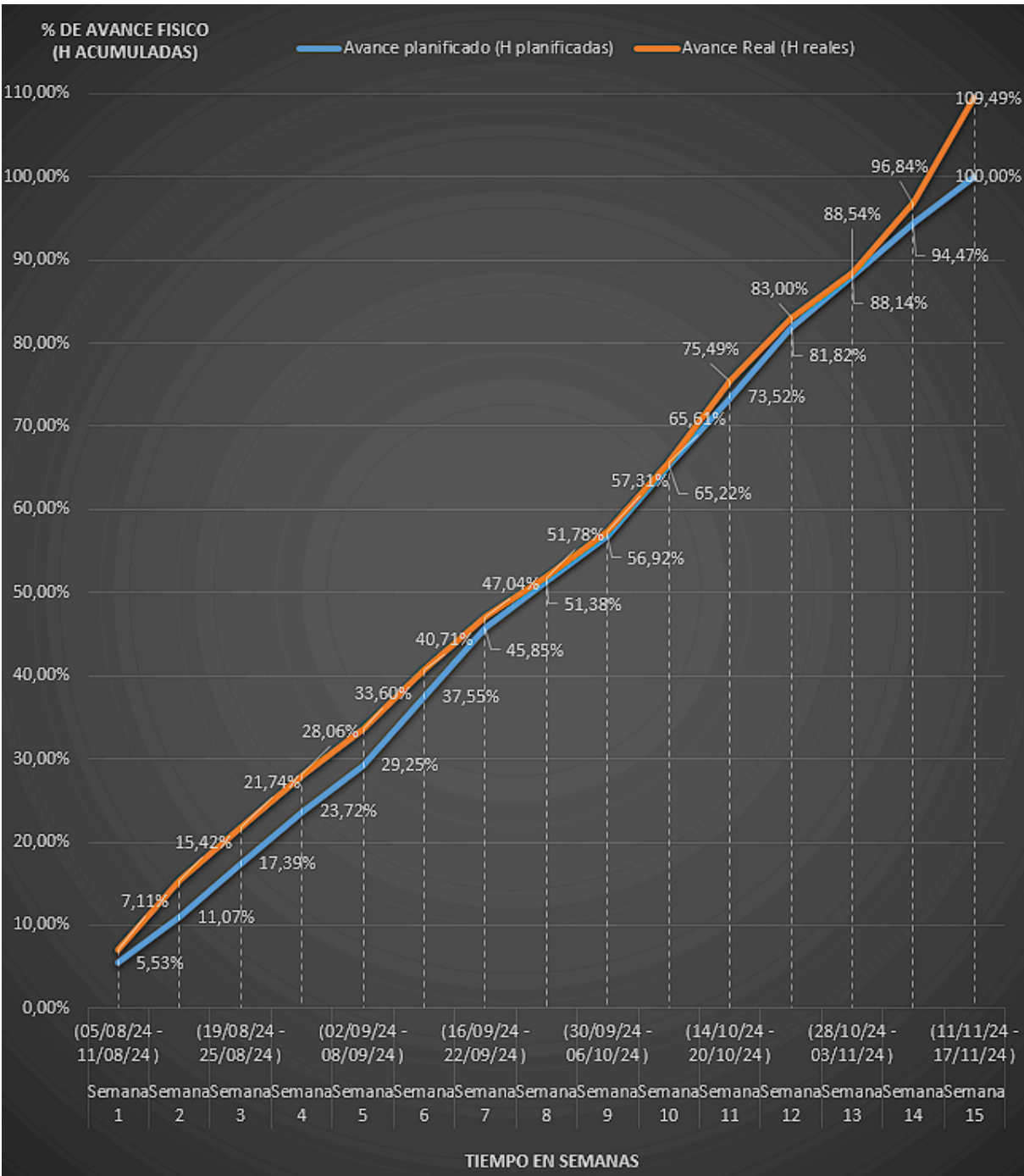


Figura. 13

Podemos observar en el gráfico anterior dos líneas temporales, la azul que hace referencia al tiempo planificado y la naranja al tiempo real de ejecución. En el caso del tiempo planificado se refleja como medida estándar para tomar como referencia el tiempo total, siendo el ancho de la misma igual al %100 esperado de avance. Sin embargo, la línea naranja en este caso muestra que el tiempo ejecutado excede el total planificado. Por eso de forma acumulada sobrepasa el 100% de avance inclinándose por un 109,49%. Un dato de relevancia es que el 100% de avance planificado es equivalente a 253 horas de trabajo acumulado.

También, adjuntamos un gráfico de barras para que se pueda comparar de forma visual el proceso semana a semana:

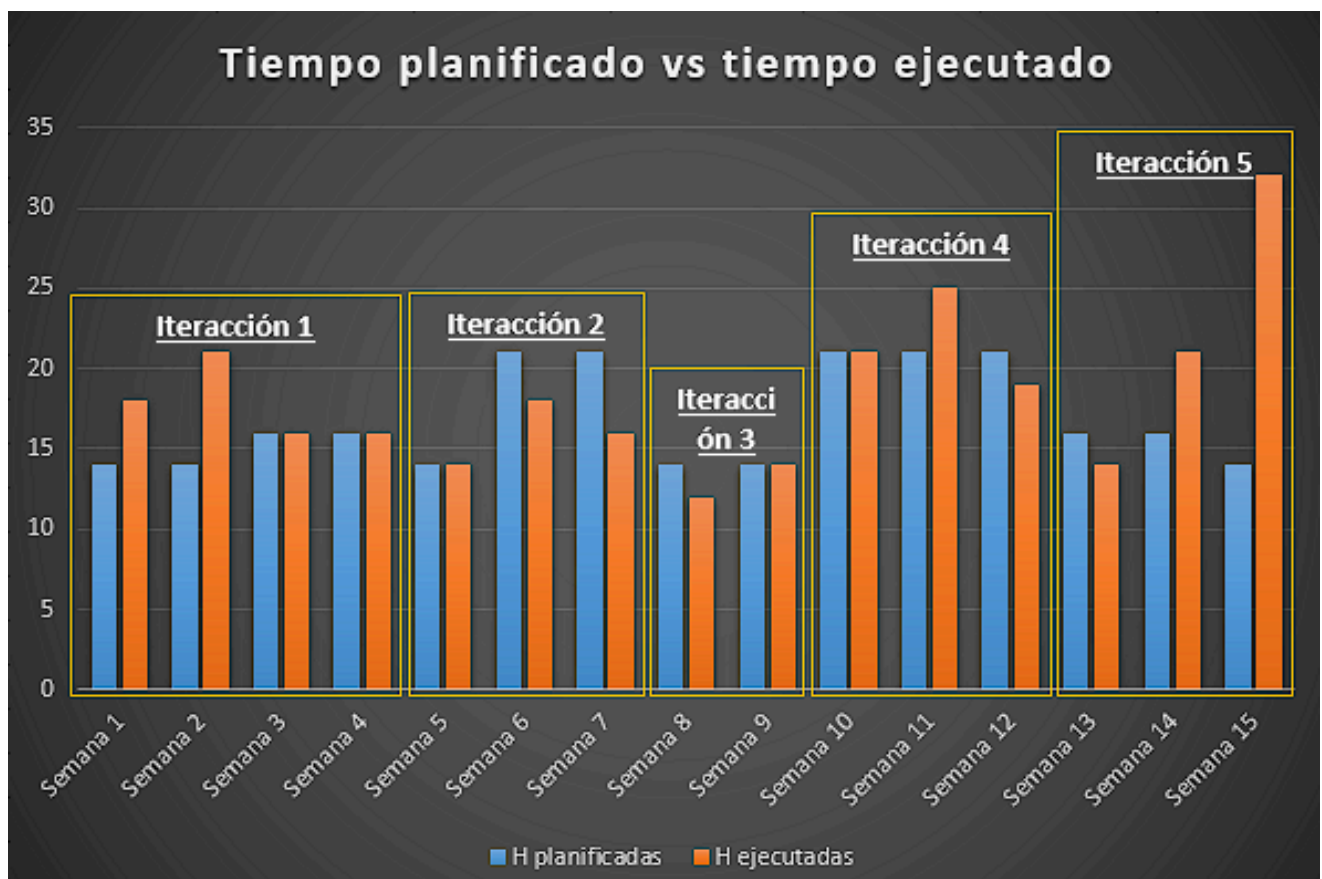


Figura. 14

Ejecución del Proyecto

Iteración I: Consolidado de ideas y validado de Mockups

Propuesta

Para mitigar el riesgo 2 planteado en el apartado de riesgos se propuso una primera instancia de investigación profunda haciendo uso de diferentes prototipos de software para comparar los resultados de los mismos. En dicha investigación se espera elegir la LLM que más se adapte a nuestros requerimientos y necesidades. Así como también las herramientas que daremos uso para la implementación del chatbot y el entrenamiento del modelo.

Se estima que esta iteración debe durar un mes para ser coherentes con los tiempos disponibles pero sin descuidar ser prudentes a la hora de investigar a profundidad como se requiere.

Resultados

Se logró cumplir con los plazos establecidos para la iteración, se esperaba definir las herramientas para un primer prototipo estable con un LLM ya seleccionado. Para esto utilizamos el LLM LLama 3.1 con 8 billones de parámetros que acepta el uso de RAGs para su aprendizaje y entrenamiento. En cuanto a herramientas definimos como entorno de trabajo inicial a LM Studio para descargar el modelo y la base de datos de vectores.

Encontramos dificultades para el entrenamiento del modelo y las respuestas del mismo en esta instancia no son coherentes con la información que se pretende que responda.

Conclusiones

Fueron muchas las herramientas testeadas para el inicio de este proyecto, se tuvo mucha incertidumbre al inicio y eso frenó el avance en varias ocasiones. Nos topamos con la primera gran dificultad que es plantear una solución que proporcione al usuario respuestas más acertadas tomando en cuenta el material que debe de aprender el modelo. En la próxima iteración será prioridad. Se adjunta una imagen del software que utilizamos con Anything LLM para hacer las primeras pruebas:

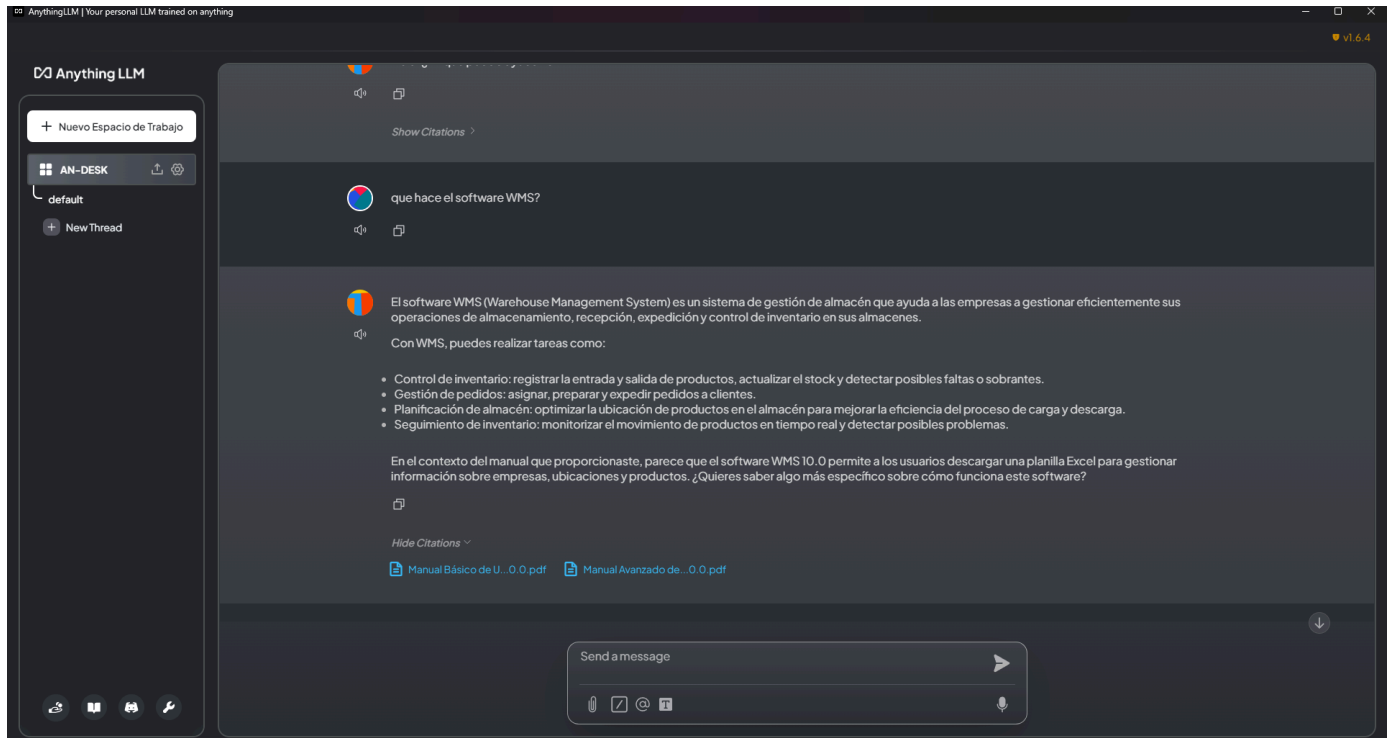


Figura. 15

Iteración II: Relevo de requerimientos

Propuesta

Se pretende seguir con el entrenamiento del modelo y en esta instancia generar datasets con los datos que nos proporcionaron los documentos de nuestro cliente. Para así por medio del fine tuning lograr solucionar el problema descrito en la iteración anterior. Se plantea también el cierre de la implementación de varios de los requerimientos planteados en este caso RF-01, RF-02 y continuar con el desarrollo del RF-04. También se espera profundizar en el análisis de los riesgos y el plan SQA para estandarizar ciertos apartados del proyecto necesarios para una organización más certera.

Se destinarán 3 semanas a la implementación y entrenamiento de modelo.

Resultados

Luego de reentrenar el modelo con uno de los documentos del cliente (el más pequeño de 40 páginas) y crear el dataset logramos a través del fine tuning que respondiera mejor a las preguntas realizadas en el chat con el contenido que fue entrenado. Igualmente hay detalles en el formato que pueden y deben pulirse a futuro para ofrecerle más detalle y claridad en las respuestas al usuario consumidor.

También se logró describir correctamente lo esperado para la documentación en la cual se seguirá trabajando.

Conclusiones

En esta iteración tuvimos que extender un poco más los tiempos para generar los datasets y re entrenar nuestro modelo ya que las limitantes en cuanto a software y hardware hacen que nuestros procesos se atrasen. Sin embargo, avanzamos hacia una nueva iteración sin descuidar los detalles que aún faltan por cumplir.

Iteración III - Funcionalidades

Propuesta

En este punto nos enfrentamos al reto de integrar de alguna forma la base de datos de nuestro cliente a nuestro chatbot para que el modelo pueda responder a las consultas más especializadas de los usuarios de WIS.

Se llevará a cabo este proceso en 2 semanas.

Resultados

Se pauso la investigación para llevar a cabo la funcionalidad de tomar datos de la base de datos y nos centramos plenamente a la mejora del modelo, re entrenarlo y enseñarle con nuevos datasets con un formato mucho más digerido (preguntas y respuestas) para el chatbot.

Conclusiones

Al cambiar los objetivos de la iteración se logró pulir el modelo para obtener las respuestas deseadas. El proceso llevó mucho más de lo pensado. La implementación de la funcionalidad con base de datos queda para el futuro.

Iteración IV - Primera validación del sistema

Propuesta

Seguiremos documentando el proceso realizado y a su vez se intentará re entrenar el chatbot y lograr testear los resultados a través de consultas específicas sobre los datos del dataset. Se espera poder adaptar nuestra solución a los servicios que ya brinda WIS. Uniendo una ventana de chat con nuestro chatbot a su plataforma.

Se espera cerrar en una semana este sprint.

Resultados

Se logró implementar el servicio deseado y testeamos con éxito el chatbot luego de entrenarlo con un nuevo dataset con más información. El modelo en esta instancia pierde naturalidad y coherencia en las respuestas por el volumen de información que debe de manejar pero contesta de forma más precisa. Falta adaptar el chat a la plataforma del cliente. El chat ya está en condiciones de ser agregado al software del cliente solo falta el permiso del mismo y los accesos para implementar.

Conclusiones

Seguimos avanzando y nos topamos con el problema de tener que esperar los permisos para implementar el chat directo en el browser de WIS. A su vez, se pretende seguir trabajando en el modelo y la forma que el mismo proporciona las respuestas.

Iteración V - Segunda validación del sistema

Propuesta

Se deberá testear a profundidad el producto para evitar riesgos y vulnerabilidades del sistema, a su vez mejorar las integraciones que se hayan podido lograr y crear un manual básico para los usuarios que quieran consumir nuestro servicio.

Se espera utilizar la última semana del plazo total para llevar a cabo esta tarea.

Resultados

Se tuvieron múltiples problemas a la hora de re entrenar el modelo, al generar un dataset específico se había perdido naturalidad en las respuestas y logramos solucionarlo complejizando aún más el dataset para que pueda responder al usuario correctamente. Se implementaron también modificaciones en el chat mismo que permiten al usuario re generar una respuesta si esta no les convence.

Conclusiones

Tardamos mucho más de lo esperado en esta etapa, cuando tuvimos que realizar fine tuning al modelo con los documentos se tornó en un proceso largo y arduo. No fue lo mismo tratar con un bajo volumen de prueba de información como lo era el primer manual de la empresa a todo el material teórico de WIS. Cuando intentamos entrenar con grandes volúmenes de información tuvimos múltiples dificultades que logramos superar.

Capítulo VI: Conclusiones y Trabajo futuro

Introducción al capítulo

Llegados a este punto, nuestro cliente detalla una devolución del trabajo realizado para su empresa. También añadiremos nuestras propias reflexiones y nuestra proyección del trabajo a futuro.

Devolución del cliente

En el siguiente segmento adjuntamos la carta con la devolución final de nuestro cliente:



Figura. 16

Estimado equipo de Proyecto Final - CURE

En nombre de WIS, me dirijo a ustedes en mi calidad de ejecutivo de cuentas de la empresa WIS, para expresar nuestro más sincero reconocimiento y satisfacción del proyecto desarrollo de chatbot. Durante la presentación de cierre, pudimos apreciar cómo esta herramienta tiene el potencial de optimizar significativamente la atención a consultas frecuentes y técnicas dentro de nuestra plataforma.

Valoramos la calidad y la viabilidad de la solución presentada, así como su compromiso con la excelencia en el campo del desarrollo de software. Creemos firmemente que la colaboración entre ambas instituciones.

Desde la concepción hasta la validación final, cada fase del proyecto ha sido abordada con un enfoque minucioso, absoluta responsabilidad y un compromiso excepcional por parte de su

equipo de desarrollo. Las diversas iteraciones han marcado hitos significativos en el proceso, desde la consolidación de ideas hasta la validación final con nosotros, como cliente.

El chatbot demostró ser intuitivo y funcional, cumpliendo con las expectativas planteadas, entendemos que algunas demoras en la respuesta no fue un problema del equipo sino de la capacidad de procesamiento utilizada. Lo que más se destaca es la facilidad de uso para cualquier persona.

Como sugerencia, sería interesante explorar cómo el chatbot podría manejar casos más complejos o incluir idiomas adicionales en el futuro.

Felicitamos al equipo por este logro y estamos confiados en que este desarrollo pueda ser utilizado en el futuro.

Atentamente,

Rolando Oliva Silva

Conclusiones del equipo

Fue un proyecto desafiante que demandó una gran etapa de investigación que a su vez se extendió a lo largo de todo el trayecto de desarrollo. Nos enfrentamos a múltiples dificultades las cuales la mayoría logramos sobrepasar para entregar un producto estable y funcional. Sin dudas, todo el equipo concuerda en que la inteligencia artificial es un nicho en pleno desarrollo. Despertó nuestro interés cuando recibimos inicialmente la propuesta y pese a nuestras limitantes con respecto a recursos logramos desarrollar un chatbot que cumple con los requerimientos iniciales de nuestro cliente. Sin duda esta experiencia nos abrió las puertas a un nuevo mundo en pleno desarrollo y estamos contentos que nos dieran la oportunidad de trabajar en una propuesta innovadora.

Trabajo futuro

Notamos que nuestro proyecto si bien termina en esta instancia hay múltiples mejoras que se pueden implementar para mejorar el producto y la experiencia del usuario, algunas de las cuales detallaremos a continuación:

- ❖ Soporte multilingüe.
- ❖ Optimización del rendimiento del modelo.
- ❖ Aumentar la capacidad base del LLM para procesar más parámetros.
- ❖ Integración con el sistema de WIS.
- ❖ Consultas a una base de datos mediante lenguaje natural.
- ❖ Capacidad para interactuar con el modelo mediante el uso de la voz.
- ❖ Capacidad para narrar las respuestas generadas mediante la voz.
- ❖ Feedback del cliente a través de reacciones o botones para indicar si las respuestas que da el modelo son las que el usuario necesita.

Agradecimientos

En primer lugar, nos gustaría agradecer a nuestro profesor y tutor Pablo Granero por guiarnos durante todo este proceso de desarrollo y alentarnos a seguir adelante pese a las diferentes adversidades que se nos presentaron a lo largo del mismo.

También queremos dedicar este espacio para agradecer a la persona que recibió nuestro producto y nos dio su devolución como cliente, el sr. Rolando Silva el cual desempeña el rol de Encargado de cuenta y PM en la empresa de WIS que muy amablemente dispuso de su tiempo para escuchar nuestra presentación y probar nuestro chatbot.

Y por último nos parece importante darle un cálido agradecimiento a nuestras familias y compañeros que nos apoyaron a lo largo de este trayecto. Sabemos que detrás de cada estudiante comprometido y apasionado por lograr sus metas, hay un hogar que brinda apoyo, paciencia y ánimo constante. Gracias por confiar en nosotros, por entender las largas horas de trabajo, por las palabras de aliento en los momentos de desafío y por estar presentes, aunque sea desde la distancia, en cada paso del camino. Su respaldo ha sido fundamental para que este proyecto se hiciera realidad.

Bibliografía

- ❖ WIS. (2024). WIS. <https://wis.com.uy/>
- ❖ *Guía Normas APA 7a edición*. (n.d.).
<https://normas-apa.org/wp-content/uploads/Guia-Normas-APA-7ma-edicion.pdf>
- ❖ *Ollama*. (2024). Ollama. <https://ollama.com/library>
- ❖ *Models - Hugging Face*. (2024, August 16). Huggingface.co.
<https://huggingface.co/models>
- ❖ *Get Started | Haystack*. (2024). Haystack. <https://haystack.deepset.ai/overview/quick-start>
- ❖ *Asistentes virtuales en uso en el mundo 2019-2024 | Statista*. (2019). Statista; Statista.
<https://es.statista.com/estadisticas/972995/asistentes-virtuales-en-uso-en-el-mundo/>
- ❖ *Nomic AI*. (2024). Nomic.ai. <https://www.nomic.ai/>
- ❖ *AnythingLLM | The all-in-one AI application for everyone*. (2024). Anythingllm.com.
<https://anythingllm.com/>
- ❖ king-theme.com, & Finding-admin. (2016, December 12). *SQA (Aseguramiento de la Calidad del Software)*. Finding | Pruebas de Software SQA.
[http://findingtc.com/sqa-aseguramiento-de-la-calidad-del-software/#:~:text=SQA%20\(Sofware%20Quality%20Assurance%20o,con%20los%20resultados%20de%20estas](http://findingtc.com/sqa-aseguramiento-de-la-calidad-del-software/#:~:text=SQA%20(Sofware%20Quality%20Assurance%20o,con%20los%20resultados%20de%20estas)
- ❖ (2024). Agantty.com. <https://app.agantty.com/?locale=en#/dashboard>
- ❖ Armenia, A. (2024). *PMBOK-7th-Edition-kisstudy.com_-1-150* (3). Scribd.
<https://es.scribd.com/document/598306081/PMBOK-7th-Edition-kisstudy-com-1-150-3>
- ❖ Google Colab. (2019). Google.com. <https://colab.research.google.com/?hl=es>

- ❖ Aunoa. (2024, April 4). *¿Qué tipos de chatbot existen?* Aunoa.
<https://aunoa.ai/blog/que-tipos-de-chatbot-existen/>
- ❖ Google Colab. (2019). Google.com. <https://colab.research.google.com/>
- ❖ Hugging Face – *The AI community building the future*. (2024, November 29).
Huggingface.co. <https://huggingface.co/>
- ❖ Unsloth - *Open source Fine-tuning for LLMs*. (2024). Unsloth - Open Source Fine-Tuning
for LLMs. <https://unsloth.ai/>

Glosario

WIS: Warehouse Information System. Empresa de nuestro cliente.

WMS: Warehouse Management System. Es el tipo de software que ofrece nuestro cliente.

Modelo de IA: Conjunto de algoritmos y estructuras matemáticas que permite a las computadoras realizar tareas que normalmente requieren inteligencia humana, como el reconocimiento de patrones, la toma de decisiones y el procesamiento del lenguaje natural.

Base de datos de vectores: Un tipo de base de datos diseñada para almacenar y buscar datos en forma de vectores (representaciones numéricas). Se utiliza comúnmente en aplicaciones de machine learning para realizar búsquedas similares basadas en la proximidad de los vectores.

RAG (Retrieval-Augmented Generation): Un enfoque de generación de texto que combina la recuperación de información y la generación de lenguaje natural. Utiliza bases de datos o documentos existentes para mejorar la calidad y relevancia del texto generado por una IA.

Fine Tuning: Un proceso de ajuste fino en el que un modelo de IA pre entrenado se adapta a una tarea específica mediante un entrenamiento adicional en un conjunto de datos más pequeño y especializado. Esto mejora su rendimiento en esa tarea particular.

Dataset: Una colección estructurada de datos que se utiliza para entrenar, evaluar y validar modelos de IA. Los datasets pueden incluir texto, imágenes, números u otro tipo de información.

Temperatura: Un parámetro utilizado en modelos de generación de texto que controla la aleatoriedad de las respuestas generadas. Una temperatura baja produce resultados más determinísticos y coherentes, mientras que una temperatura alta genera respuestas más diversas y creativas.

Prompt: Un conjunto de instrucciones o preguntas que se le proporcionan a un modelo de IA para guiar su generación de texto o respuesta. El prompt puede influir significativamente en el resultado obtenido.

Embedding: Una representación numérica de datos (como palabras, frases o imágenes) en un espacio vectorial de menor dimensión; Estos modelos convierten texto en vectores que capturan las relaciones semánticas y contextuales entre palabras o frases.

Cloud Computing: La entrega de servicios informáticos (como almacenamiento, procesamiento y aplicaciones) a través de Internet. Permite el acceso a recursos de computación escalables y flexibles, sin la necesidad de gestionar infraestructuras locales.

SCRUM: Es una metodología usada en el desarrollo de Software basada en un proceso rápido, flexible, iterativo e incremental. Scrum tiene como objetivo ofrecer valor al cliente durante todo el desarrollo del proyecto a través del progreso continuo y una comunicación sincera entre el equipo de desarrollo y el cliente.

Anexo

Diagrama de Gantt en 2 partes

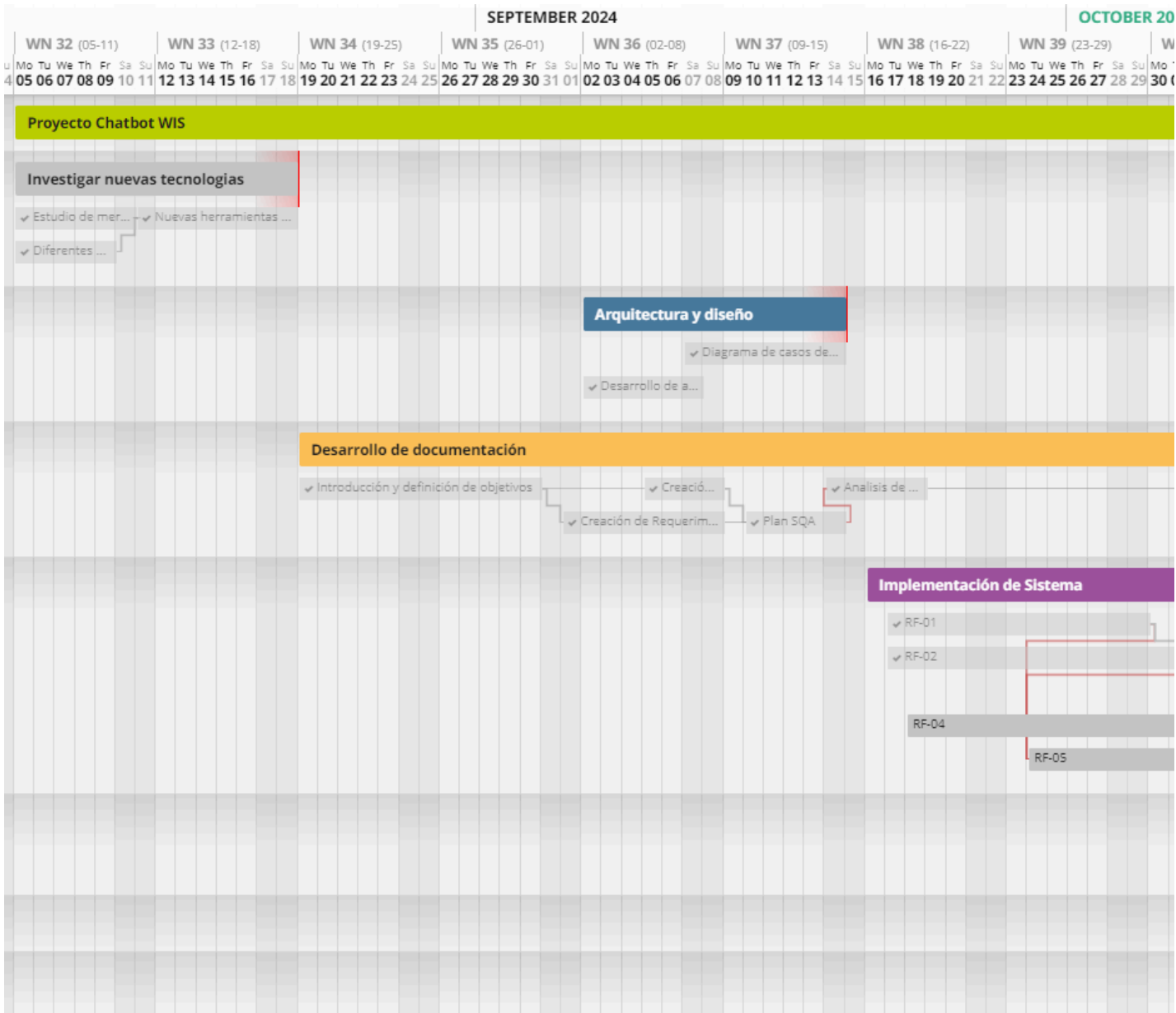


Figura. 17

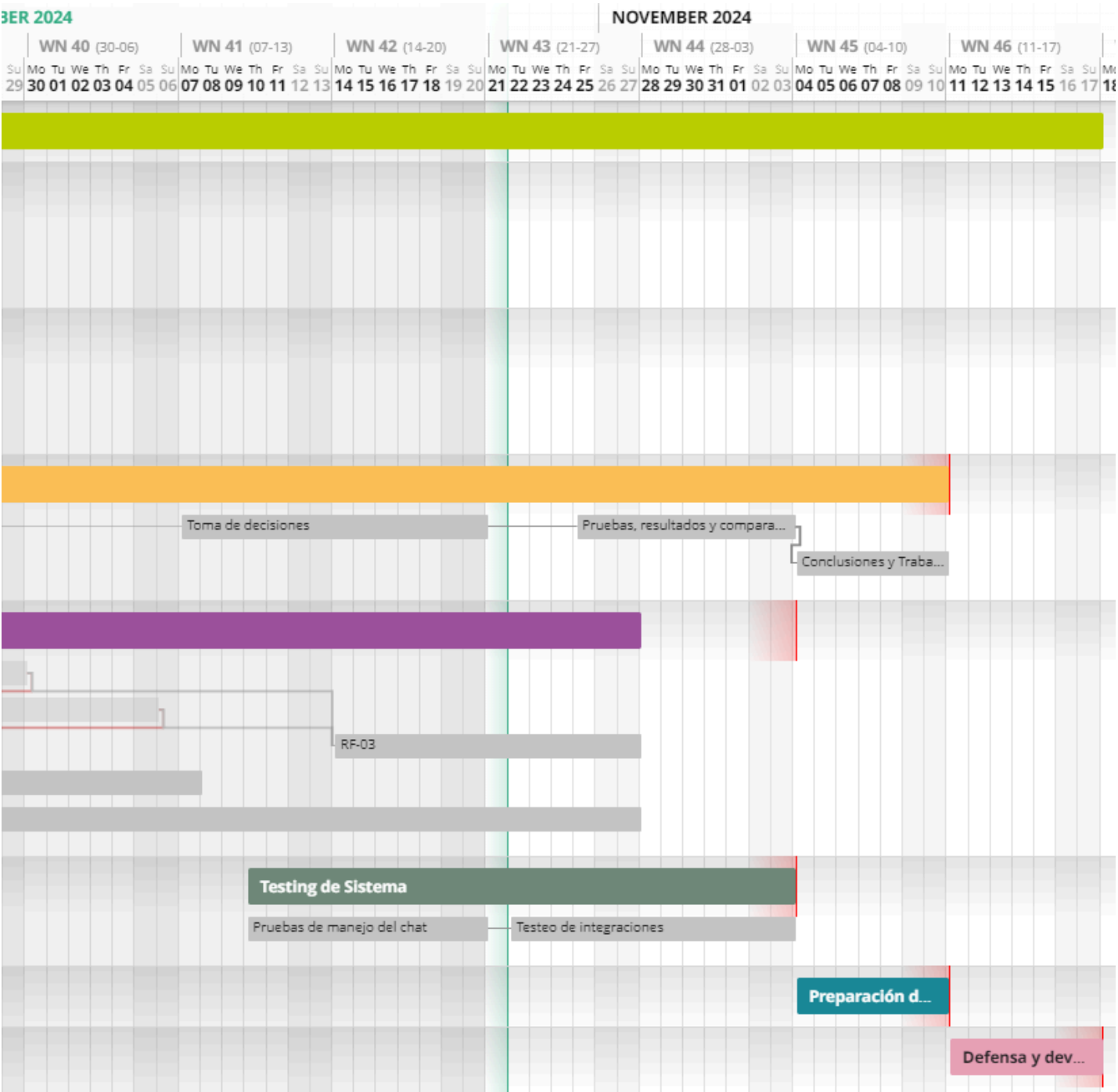


Figura. 18

Código para crear DataSet

A continuación se adjunta el código en python para crear un dataset pasandole un archivo en formato pdf:

```
import os
import json
import fitz  # PyMuPDF para leer PDFs

PDF_FOLDER = "pdf"
OUTPUT_FILE = "dataset.json"

# Índice de ejemplo
index_example = {
    "Reglas de Almacenaje 10.0.pdf": {
        "Conceptos Básicos": 4,
        "Grupos de Productos": 4,
        "Definición de Grupos de Productos": 4,
        "Definición de Reglas de Agrupación": 6,
        "Ver Grupo de un Producto": 10,
        "Asignación de Grupos a Etiquetas de Recepción": 11,
        "Lógica de Almacenaje": 11,
        "Tipos de Lógicas de Almacenaje": 11,
        "Parametrización de Lógicas de Almacenaje": 11,
        "Estrategia de Almacenaje": 15,
        "Asociación de Estrategia de Almacenaje": 15,
        "Descripción General del Algoritmo": 17,
        "Administración de Estrategias de Almacenaje": 23,
        "Panel de Estrategias": 23,
        "Asignación de Lógicas de Almacenaje a Estrategias": 23,
        "Asociación de Estrategias": 27,
        "Deshabilitación de Estrategias de Almacenaje": 30,
        "Almacenamiento Agrupado basado en Estrategias de Almacenaje": 31,
        "Habilitación de Sugerencias de Almacenaje": 30,
        "Administración de Motivos de Rechazo": 32,
        "Colector de Almacenamiento Agrupado": 34,
        "Rechazo de Sugerencias de Almacenaje": 39
    }
}
```

```

}

# Función para extraer texto de PDFs de la carpeta /pdf
def extract_text_from_pdfs(directory):
    texts = {}
    for filename in os.listdir(directory):
        if filename.endswith(".pdf"):
            filepath = os.path.join(directory, filename)
            with fitz.open(filepath) as doc:
                texts[filename] = [page.get_text() for page in doc] # Obtener
texto directamente
    return texts

# Crea el dataset en formato JSON copiando el texto tal cual desde el PDF
def create_dataset(index, texts, output_file):
    data = []

    # Añadir el mensaje del sistema solo una vez
    system_message = {
        "role": "system",
        "content": "Eres un asistente virtual llamado Asistente WMS, enfocado en
responder preguntas relacionadas al sistema WMS (Warehouse Management System), de
forma concreta y precisa."
    }
    data.append(system_message)

    # Generar preguntas y respuestas basadas en el índice
    for filename, chapters in index.items():
        for title, start_page in chapters.items():
            user_content = title # El título será la pregunta
            end_page = get_end_page(chapters, title, start_page)
            assistant_content = extract_relevant_info(texts[filename], start_page,
end_page)

            # Añadir los mensajes de usuario y asistente
            data.append({
                "role": "user",
                "content": user_content
            })

```

```

        data.append({
            "role": "assistant",
            "content": assistant_content # Copiar texto del PDF tal como está
        })

# Guardar el archivo JSON con la estructura generada
with open(output_file, 'w', encoding='utf-8') as f:
    json.dump(data, f, ensure_ascii=False, indent=4)

# Determina la página final de un capítulo
def get_end_page(chapters, current_title, start_page):
    # Obtener las páginas de inicio de los capítulos
    pages = sorted(set(chapters.values()))
    current_index = pages.index(start_page)
    if current_index + 1 < len(pages):
        return pages[current_index + 1] - 1 # Restar 1 para que no incluya la
        # página del siguiente título
    return start_page # Si es el último capítulo, usa la misma página

# Extrae el texto relevante entre páginas de inicio y fin (sin procesar)
def extract_relevant_info(text_pages, start_page, end_page):
    # Extrae el texto desde start_page hasta end_page tal cual
    relevant_text = ""
    for page_num in range(start_page - 1, end_page):
        if page_num < len(text_pages):
            relevant_text += text_pages[page_num].strip() + "\n"

    return relevant_text.strip() # Copiar el texto tal como está

if __name__ == "__main__":
    index = index_example # Aquí puedes cargar tu índice real
    texts = extract_text_from_pdfs(PDF_FOLDER)
    create_dataset(index, texts, OUTPUT_FILE)

```

Código para crear DataSet con respuestas vacías

```
import json

# Nombre del documento y archivo de salida
DOCUMENT_NAME = "License Plate Number 10.0"
OUTPUT_FILE = "License-Plate-Number-dataset.json"

titles = [
    "Generalidades",
    "Introducción",
    "Conceptos básicos",
    "Atributos",
    "Validaciones",
    "Tipos de LPN",
    "Template de etiquetas LPN",
    "Estados",
    "Plantillas de etiquetas de LPN",
    "Sustitución de etiquetas WIS",
    "Operaciones sobre LPNs",
    "Creación y consulta de LPNs",
    "Consulta de LPN",
    "Consulta de códigos de barra de un LPN",
    "Impresión de LPN",
    "API de LPN",
    "Creación de un LPN",
    "Obtención de un LPN",
    "Obtención de un LPN por Identificador Externo y Tipo",
    "Creación de LPNs mediante Excel",
    "Otras consultas sobre LPNs",
    "Consulta de Contenido de LPN",
    "Consulta de LPN por Atributos",
    "Consulta por atributos de cabezal",
    "Consulta por atributos del detalle",
    "Consulta de Logs LPN",
    "Logs de Cabezal",
    "Logs de Detalles",
    "Consulta de Productos por Atributos",
```

"Edición de atributos de LPNs",
"Recepción de LPNs",
"Agendamiento de LPNs",
"Planificación de LPNs",
"Colector de Recepción",
"Confirmación de Recepción",
"Cross Docking sobre LPNs",
"Controles de calidad sobre LPNs",
"Consulta de Etiquetas",
"Mesa de Clasificación",
"Colector de Almacenamiento Agrupado",
"Colector de Almacenamiento Fraccionado",
"Manejo de Stock sobre LPNs",
"Consulta de Stock",
"Colector de Auditoría de LPN",
"Auditoría en una fase",
"Auditoría en dos fases",
"Colector de Transferencia",
"Colector de Inventario",
"Colector de Ajustes de stock",
"Colector de Cambio de lote",
"Colector de Reabastecimiento por etapas",
"Marcado de Averías sobre LPNs",
"Pedidos sobre LPNs",
"Creación de pedidos sobre LPNs",
"Pedidos sobre LPNs por Panel Web",
"Pedidos de LPNs completos",
"Pedidos de mercadería contenida en LPNs específicos",
"Pedidos de mercadería basados en atributos",
"Pedidos sobre LPNs por API",
"Pedidos de LPNs completos",
"Pedidos de mercadería contenida en LPNs específicos",
"Pedidos de mercadería basados en atributos",
"Consultas de LPN en pedidos",
"LPNs de detalle de pedido",
"Atributos de detalle de pedido",
"Detalle de atributos de detalle de pedido",
"Anulación de pedidos pendientes sobre LPNs",
"Eliminar pedidos pendientes",

"Eliminar pedidos pendientes (LPN específico)",
"Eliminar pedidos pendientes (LPN atributos)",
"API de Pedidos Anulados",
"Preparación de LPNs",
"Ajustes en Liberación de Onda para manejo de LPN",
"Análisis de rechazo para preparaciones sobre LPNs",
"Anulación de preparaciones sobre LPNs",
"Colector de Reabastecimiento de ubicaciones bajas",
"Colector de Picking Normal",
"Configuración colector de picking",
"Modalidades de picking",
"Modalidad de picking Tradicional",
"Modalidad de picking LPN",
"Sugerencia de LPNs candidatos",
"Colector de Extracción",
"Modalidad de picking Mixta",
"Modalidades de picking LPN",
"Pickear en LPN y dejar el remanente en un LPN nuevo",
"Pickear en LPN y dejar el remanente suelto",
"Pickear en un nuevo LPN y dejar el remanente en LPN original",
"Pickear suelto y dejar el remanente en el LPN original",
"Colector de Picking Manual",
"Colector de Consolidación de Contenedores",
"Colector de Devolución de picking",
"Colector de Partición de Contenedores",
"Colector de Separación de Picking",
"Mesa de Empaque",
"Colector de Control de contenedor",
"Colector de Ensamblado de fórmulas",
"Expedición de LPNs",
"Panel de Egresos",
"Pedidos de mostrador Panel Web",
"Colector de Expedición",
"Colector de Asignación de carga",
"Colector de Carga de camión",
"Colector de Descarga de contenedores",
"Cierre de Camión"


```

# Función para crear el dataset con respuestas vacías
def create_dataset_with_empty_responses(titles, document_name, output_file):
    data = [
        {
            "role": "system",
            "content": "Eres un asistente virtual llamado Asistente WMS, enfocado
en responder preguntas relacionadas al sistema WMS (Warehouse Management System),
de forma concreta y precisa."
        }
    ]

    # Crear preguntas y respuestas vacías basadas en los títulos
    for title in titles:
        user_content = title
        assistant_content = f"Según el manual {document_name}: "

        # Añadir entrada de usuario y respuesta vacía
        data.append({"role": "user", "content": user_content})
        data.append({"role": "assistant", "content": assistant_content})

    # Guardar el archivo JSON
    with open(output_file, 'w', encoding='utf-8') as f:
        json.dump(data, f, ensure_ascii=False, indent=4)

# Generar el dataset
create_dataset_with_empty_responses(titles, DOCUMENT_NAME, OUTPUT_FILE)

```

Código para separar preguntas de las respuestas

```
import json

# Ruta del archivo de entrada
input_file_path = "WMS-dataset2.json"

# Rutas de los archivos de salida
user_questions_path = "WMS-user-questions.json"
assistant_responses_path = "WMS-assistant-responses.json"

# Cargar el archivo original
with open(input_file_path, "r", encoding="utf-8") as f:
    dataset = json.load(f)

# Separar las preguntas del usuario y las respuestas del asistente
user_questions = [entry for i, entry in enumerate(dataset) if entry["role"] ==
"user" and i % 2 == 1]
assistant_responses = [entry for i, entry in enumerate(dataset) if entry["role"]
== "assistant" and i % 2 == 0]

# Guardar las preguntas del usuario en formato comprimido
with open(user_questions_path, "w", encoding="utf-8") as f:
    f.write("[\n")
    f.write(",\n".join(json.dumps(entry, ensure_ascii=False) for entry in
user_questions))
    f.write("\n]")

# Guardar las respuestas del asistente en formato comprimido
with open(assistant_responses_path, "w", encoding="utf-8") as f:
    f.write("[\n")
    f.write(",\n".join(json.dumps(entry, ensure_ascii=False) for entry in
assistant_responses))
    f.write("\n]")

print(f"Archivos generados en formato comprimido:\n- {user_questions_path}\n-
{assistant_responses_path}")
```