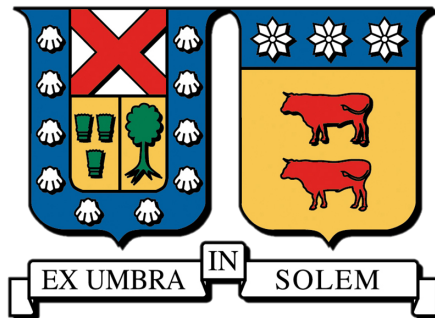


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



“Uso de modelos extensos de lenguaje como motor de una plataforma intermediaria entre usuario y base de datos.”

JOSÉ IGNACIO ALEGRÍA VERGARA

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO
CIVIL TELEMÁTICO**

PROFESOR GUÍA:

WERNER CREIXELL

PROFESOR CORREFERENTE:

NICOLÁS JARA

ABRIL 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Uso de modelos extensos de lenguaje como motor de una plataforma intermediaria entre usuario y base de datos.

Nombre del candidato(a): José Ignacio Alegría Vergara

Carrera / Grado: Ingeniería Civil Telemática / Pregrado

Campus: Casa Central Valparaíso ; **Departamento:** Departamento de Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Werner Creixell, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis: 
Fecha: _____ ; Firma: _____

Estudiante o Candidato(a): 
Fecha: 03-07-2026 ; Firma: _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Agradecimientos

Gracias a la vida que me ha dado tanto.

Resumen

Es innegable que la importancia de la información es cada vez mayor en nuestras vidas, tanto en la cotidianidad como en la vida laboral a medida que contamos con más información somos capaces de tomar decisiones con un mayor grado de seguridad y, en ocasiones, con mayor velocidad. Sin embargo, para un gran número de personas las bases de datos en las que se almacena la información representan un obstáculo para su acceso. Se identifica, entonces, una necesidad: El número de personas capaces de utilizar bases de datos es menor que el grupo beneficiado por su uso.

Con esto en mente, se plantea el uso de modelos extensos de lenguaje, LLM por sus siglas en inglés, como interfaz entre usuario y base de datos. Debe señalarse que estos modelos han demostrado la capacidad de eliminar la brecha digital existente entre diferentes tipos de usuarios.

Dentro de este marco, el siguiente informe detalla el diseño y desarrollo de una plataforma diseñada para el usuario que no cuenta con la expertiz necesaria para interactuar con bases de datos directamente. Dicho de otro modo, esta plataforma aprovecha los modelos extensos de lenguaje para permitir una comunicación eficaz y expedita entre usuario y base de datos, la cual concluye con la obtención de la información pedida, efectivamente democratizando el acceso a la información.

Keywords: Modelo extenso de lenguaje, Large Language Model, LLM, Human-Computer Interaction, HCI, NL-to-SQL, text-to-SQL, Generative AI, Model chaining, LLM chaining, GPT Model

Glosario

HCI: Disciplina ubicada en la intersección entre las ciencias de la computación, el estudio conductual y el diseño, se concentra en el estudio del uso y diseño de interfaces entre usuarios y equipos.

AI: Campo dentro de las matemáticas, ingeniería y ciencias de la computación, preocupado por construir programas que imiten las capacidades de un ser humano en términos de lógica, procesamiento de imágenes y lenguaje, entre otras.

NN: Arquitectura de inteligencia artificial popularizada en las primeras décadas del siglo XXI. Estas están compuestas por una capa inicial (tensor), el cual alimenta n capas “apiladas” las cuales operan secuencialmente sobre la información según pesos obtenidos durante el entrenamiento. Una característica es que cada neurona está conectada a todas las neuronas de la siguiente capa.

CNN: Tipo de red neuronal que se caracteriza por no tener conexiones entre todas sus capas, teniendo regiones pequeñas conectadas a cada neurona de la capa anterior, este enfoque busca concentrarse en patrones de áreas específicas.

Transformer: La arquitectura de Transformers o Transformadores es una arquitectura de IA introducida el año 2017 que reemplaza la recurrencia de las NN por el mecanismo de autoatención. Gracias a esto, es capaz de procesar secuencias completas en paralelo, reduciendo el tiempo de procesamiento de gran manera. Esta arquitectura está compuesta por codificadores (encoders) y decodificadores (decoders).

Autoatención: Mecanismo mediante el cual los transformadores son capaces procesar una secuencia en paralelo, extrayendo la relevancia de cada palabra en el texto a analizar respecto a todas las otras palabras presentes.

LLM: Implementación de la arquitectura de transformadores, utiliza miles de millones de parámetros y volúmenes colosales de datos de entrenamiento para obtener modelos capaces de comprensión y generación de texto a un nivel sin precedentes.

BERT: Modelos de lenguaje en los que se descartan los decodificadores, esto permite a modelos basados en BERT leer la secuencia de entrada en ambas direcciones, logrando

construir contextos ricos y representaciones profundas del lenguaje.

GPT: Modelos de lenguaje en los que se descartan los codificadores, esto junto a un mecanismo modificado de autoatención permiten a los modelos basados en GPT predecir estadísticamente la siguiente palabra de la secuencia. Gracias a esto, los modelos basados en GPT tienen buenos resultados en tareas como la generación de texto natural, razonamiento y la continuidad conversacional.

RAG: Técnica utilizada en la generación de instrucciones para LLM, esta permite obtener documentos específicos que se asemejan al texto introducido por el usuario para poder editar el contexto dinámicamente.

Token: Elemento básico que compone una palabra, estos permiten que el LLM maneje el lenguaje internamente, una palabra puede estar compuesta de uno o más tokens.

Tokenización: Proceso mediante el cual una secuencia de entrada se convierte en múltiples tokens para ser procesados.

Modelo de incorporación: Modelo capaz de transformar secuencias de texto en tokens, estos se almacenan para posteriormente compararlos con las entradas del usuario.

Incorporación: Representaciones multidimensionales de documentos obtenidos por un modelo.

Métricas de similitud: Herramientas para evaluar las relaciones entre elementos, escalables a elementos vectoriales n-dimensionales. La distancia euclidiana y similitud del coseno son ejemplos de estas métricas

Back-end: Corresponde a los elementos que contienen la lógica necesaria para que un sistema opere.

Front-End: Corresponde a los elementos que conforman la interfaz o sección observable de un sistema.

API: Interfaz que permite la comunicación intra o inter-sistemas.

Acrónimos

HCI: Human Computer Interaction

NL: Natural Language

SQL: Structured Query Language

AI: Artificial Intelligence

NN: Neural Network

CNN: Convolutional Neural Network

LLM: Large Language Model

BERT: Bidirectional Encoder Representations from Transformers

GPT: Generative Pre-trained Transformers

RAG: Retrieval Augmented Generation

API: Application Programming Interface

Índice de figuras

4.1. Diagrama de contexto	29
4.2. Diagrama de arquitectura	30
5.1. Modelo de navegación	36
5.2. Inicio de sesión	37
5.3. Menú principal	37
5.4. Registros de conversaciones	37
5.5. Métricas de satisfacción	37
5.6. Consultar datos	38
5.7. Inicio de sesión	38
5.8. Menú principal - Chatbot	38
5.9. Registros de conversaciones	39
5.10. Métricas de satisfacción	39
5.11. Vista principal con barra lateral y sección de administrador	39
6.1. Diagrama de flujo del prototipo	51
7.1. Correo para restablecer contraseña	66
7.2. Correo para aviso de creación de usuario	66
7.3. Correo para aviso de actualización de rol	67
7.4. Vista de inicio de sesión	68
7.5. Vista de registro	68
7.6. Vista de solicitud de nueva contraseña	69

7.7. Vista de ingreso de código	69
7.8. Vista de cambio de contraseña	69
7.9. Vista de inicio para usuario con permiso asignado	70
7.10. Vista de inicio para usuario sin permiso asignado	70
7.11. Vista principal de chatbot	71
7.12. Vista de barra lateral con conversaciones	71
7.13. Vista para calificar conversación	71
7.14. Vista de tabla de conversaciones	72
7.15. Vista de tabla de métricas	72
7.16. Vista de modal con preguntas respondidas del usuario	73
7.17. Vista de modal con métricas de la conversación y ejemplo de uso del buscador	73
7.18. Vista de administración de usuarios	74
7.19. Vista de modal de actualización de rol de usuario	74
7.20. Vista de administración de roles	75
7.21. Vista de modal de creación y actualización de roles	75
7.22. Vista de permisos	75
7.23. Barra de navegación	76

Índice de cuadros

1.1. Objetivo general	9
1.2. Objetivos específicos	10
3.1. Perfiles de usuario	22
3.2. Requisitos funcionales	23
3.3. Requisitos no funcionales	24
3.4. Eventos externos	25
3.5. Respuestas del sistema	25
4.1. Módulos y definiciones	33
4.2. Matriz Requisitos funcionales y módulos	34
5.1. Eventos y respuestas prototipo	40
6.1. Comparativa modelos familia Claude	52
6.2. Costos por modelo	53
6.3. Tasa de falsos negativos por modelo	54
6.4. Tasa de falsos positivos por modelo	55
6.5. Tiempos de ejecución por modelo	55
6.6. Costos cada 1000 tokens por modelo	55
7.1. Endpoints API	64
8.1. Riesgos y respectivas medidas de mitigación	80

Índice general

1. Introducción	7
1.1. Objetivos	8
1.1.1. Objetivo general	8
1.1.2. Objetivos específicos	9
1.2. Estructura	10
2. Estado del arte	13
2.1. Marco teórico	13
2.2. Estado del arte	15
3. Planificación del desarrollo	21
3.1. Perfiles de usuarios	21
3.2. Requisitos del sistema	22
3.2.1. Requisitos funcionales	23
3.2.2. Requisitos no funcionales	23
3.2.3. Requisitos de interfaces	24
3.2.4. Requisitos de ambiente	26
4. Diseño de arquitectura	29
4.1. Diagrama de contexto	29
4.2. Diagrama de arquitectura	30
4.3. Definición de módulos	30

4.3.1. Frontend	30
4.3.2. Backend	31
4.3.3. API	31
4.3.4. Modelos de lenguaje / Amazon Bedrock (LLMs)	32
4.3.5. Base de datos (BBDD)	32
4.4. Desglose de módulos	33
4.4.1. Matriz de relación entre módulos y requisitos funcionales	33
5. Diseño de interfaces	35
5.1. Modelo de navegación	35
5.2. Prototipo de interfaces de usuario	36
5.2.1. Primera iteración (V1.1)	37
5.2.2. Segunda iteración (V1.2)	38
5.2.3. Tercera iteración (V1.3)	39
5.2.4. Iteración final (V1.4)	39
5.3. Definición de eventos y respuestas de las interfaces	40
6. Diseño de motor traductor NL-SQL	41
6.1. Estructura de instrucciones	41
6.2. Implementación de memoria en los modelos	42
6.3. RAG y modelo de incorporación	42
6.4. Flujo principal	44
6.4.1. Identificador general	45
6.4.2. Experto SQL	45
6.4.3. Identificador NL/SQL	46
6.4.4. Corrector SQL	46
6.4.5. Traductor Data a NL	47
6.5. Flujo de gráficos	47
6.5.1. Recolector	48
6.5.2. Experto SQL-Gráfico	49

6.5.3. Generador gráfico	49
6.5.4. Limitaciones actuales del flujo	49
6.6. Diagrama de flujo	50
6.7. Comparativa de modelos de lenguaje	52
6.7.1. Diferencias en desempeño	52
6.7.2. Diferencias de costos	52
6.7.3. Diferencias generales	53
6.8. Comparativa de modelos de incorporación	54
6.8.1. Diferencias en desempeño	54
6.8.2. Diferencias de costos	55
6.8.3. Diferencias generales	56
7. Versión actual del prototipo	57
7.1. Esquema general de componentes del prototipo	57
7.1.1. Elementos generales	57
7.1.2. Vista principal	57
7.1.3. Vista conversaciones	58
7.1.4. Vista métricas	58
7.1.5. Vista administrador	58
7.1.6. Vista inicio de sesión	58
7.1.7. Vista crear cuenta	59
7.2. Descripción de componentes	59
7.2.1. Barra de navegación	59
7.2.2. Botones	59
7.2.3. Barra lateral	59
7.2.4. Input para chat	60
7.2.5. Módulo para mensajes de chat	60
7.2.6. Tabla de conversaciones	60
7.2.7. Paginación	60

7.2.8. Filtros	61
7.2.9. Tabla de Métricas	61
7.2.10. Sección para gestión de permisos	61
7.2.11. Cuadro de ingreso de credenciales	61
7.2.12. Formulario de creación	62
7.3. Entornos de desarrollo	62
7.3.1. Entorno de desarrollo local	62
7.3.2. Entorno de producción	62
7.4. Implementación de módulos y funcionalidades	63
7.4.1. Base de datos	63
7.4.2. API	63
7.4.3. Autenticación	64
7.4.4. Notificaciones	65
7.4.5. Almacenamiento de archivos	67
7.4.6. Interfaz gráfica implementada	67
8. Conclusiones	77
8.1. Gestión de riesgos	79
8.1.1. Supuestos	79
8.1.2. Dependencias	79
8.1.3. Restricciones	80
8.1.4. Riesgos	80
8.2. Verificación y validación	80
8.2.1. Verificación	80
8.2.2. Validación	81
8.2.3. Plan de pruebas	81

Capítulo 1

Introducción

Desde la Introducción de los primeros computadores personales a nuestra cotidianidad, la tecnología ha cambiado de gran manera casi todos los aspectos de nuestras vidas. En particular, han facilitado la recopilación y obtención de información de forma exponencial. Procesos que previamente se llevaban a cabo en base a planillas físicas; testimonios; conocimiento general; o incluso de manera no fundamentada, han pasado a ser manejados por una enorme cantidad de información obtenida con el propósito de mejorar la toma de decisiones.

A medida que el tiempo y la tecnología han avanzado, las herramientas utilizadas para almacenar y consumir datos han evolucionado para manejar cantidades cada vez mayores de información. Este cambio nos lleva al mundo que habitamos hoy, uno manejado por volúmenes masivos de información.

Ahora bien, una de las principales problemáticas que esto conlleva es la capacidad del acceso a la información. Para ejemplificar, es de utilidad plantear las bases de datos, una vez que estas se volvieron parte de nuestras vidas, junto a ellas se incluyó la necesidad de conocimiento técnico para acceder a la misma información que antes. Por ende, se generó una dependencia de perfiles técnicos por parte de los otros roles para acceder a datos que previamente estaban al alcance de un documento físico.

Sin embargo, esta evolución tecnológica también ha puesto nuevas herramientas a

nuestra disposición. Tal es el caso de los modelos extensos de lenguaje en el campo de la inteligencia artificial, los cuales han tenido un desempeño notorio en tareas de comprensión y traducción en formato escrito.

Con esto en mente, en este trabajo se propone aprovechar las capacidades de razonamiento de los modelos extensos de lenguaje para facilitar y democratizar el acceso a la información, utilizándolos como mediador entre humano y sistema para eliminar la brecha que existe hoy.

En función de lo planteado, estos modelos serán accesibles a través de un chat, mediante el cual los usuarios tendrán la capacidad de realizar preguntas referentes a la información que buscan para obtenerla como respuesta, sin depender de conocimiento técnico o de terceros.

A su vez, este chat se encontrará en una plataforma digital, segura y de fácil uso para acomodar a todo tipo de usuarios. De esta manera, el sistema será el responsable de interactuar con las bases de datos necesarias para entregar información al usuario.

Finalmente, el presente trabajo se lleva a cabo junto a la Fundación Arturo López Pérez, con el objetivo de entregar independencia a sus funcionarios, además de un acceso expedito a la información. De esta forma, los usuarios serán capaces de realizar análisis que puedan ser beneficiosos en el contexto de la oncología y salud con mayor velocidad.

1.1. Objetivos

1.1.1. Objetivo general

El objetivo de este trabajo consiste en facilitar el acceso a datos en la Fundación Arturo López Pérez, a través de una plataforma digital, la cual aprovecha los modelos extensos de lenguaje para escribir consultas SQL a partir de lenguaje natural y obtener la información pedida.

Objetivo general	
OG	Facilitar el acceso a datos en la Fundación Arturo López Pérez

Cuadro 1.1: Objetivo general

1.1.2. Objetivos específicos

- El modelo de lenguaje debe traducir las consultas en lenguaje natural a lenguaje SQL, el lenguaje de consultas utilizado por la fundación.
- Elaborar una API que permita acceder a las bases de datos del cliente de manera segura.
- El sistema debe recibir y mostrar métricas de satisfacción del usuario respecto a su interacción con el modelo de lenguaje.
- El sistema debe registrar las conversaciones de los usuarios con el modelo de lenguaje.
- El sistema debe ser accesible mediante una plataforma intuitiva y segura.
- El sistema debe contar con diferentes funcionalidades que faciliten o mejoren su uso.
- Las diferentes funcionalidades del sistema deben ser documentadas a cabalidad para facilitar la entrega de la plataforma.
- El desarrollo de la plataforma se debe documentar para mantener un flujo de trabajo organizado.

Objetivo	Descripción	Resultado esperado
Traducir consultas en lenguaje natural a SQL.	El sistema debe recibir una consulta en lenguaje natural, traducirla a SQL, ejecutarla sobre la base de datos y devolver la información al usuario.	Módulo traductor NL —SQL.
Elaborar Una API segura.	La API debe desplegarse en un entorno controlado y seguro que permita la conexión a las bases de datos del cliente.	Módulo API.
Administrar métricas de satisfacción del usuario.	El sistema debe gestionar métricas de satisfacción de las interacciones entre usuario y sistema.	Vista de métricas.
Administrar registros de conversaciones del usuario.	El sistema debe gestionar las interacciones entre usuario y sistema, para facilitar el análisis del uso del sistema.	Vista de historial de uso por usuario.
Elaborar una plataforma intuitiva y segura.	La plataforma debe asegurar la seguridad de la información; contar con manejo de sesiones y vistas para acceder a sus funcionalidades.	Módulos de backend y frontend de la plataforma.
Enriquecer la experiencia del usuario.	El sistema debe ofrecer herramientas que lo hagan más completo y amigable para el usuario.	Funcionalidades que mejoren la usabilidad del sistema.
Documentar el desarrollo.	Elaborar un documento de diseño y bitácora permite trabajar de manera organizada y eficaz.	Documento de diseño y bitácora de desarrollo.
Documentar las funcionalidades.	Cada funcionalidad del sistema debe contar con documentación que permita entender su uso y funcionamiento.	Documentación para la plataforma y sus funcionalidades.

Cuadro 1.2: Objetivos específicos

1.2. Estructura

Este documento se estructura de la siguiente forma: En el Capítulo 2 se tiene el Marco Teórico, donde se presentan y explican conceptos base para el desarrollo realizado;

En el Capítulo 3 se presenta el Estado del Arte, donde se muestra el estado actual de la técnica y las diferentes avenidas que se pueden seguir a la hora de realizar un desarrollo similar al presente texto; En el Capítulo 4 se tiene la Planificación del Proyecto, en este se explicitan los requerimientos del sistema y desarrollo realizados; En el Capítulo 5 se muestra el Diseño de la Arquitectura, desde su concepción como artefacto de alto nivel hasta una delimitación a nivel modular; En el Capítulo 6 se presenta el proceso de Diseño de Interfaces, detallando el proceso inicial de ideación y la evolución de los prototipos desarrollados; En el Capítulo 7 se tiene el proceso de Diseño del Motor Traductor NL-SQL, en esta sección se detalla el proceso de diseño del módulo a cargo de la funcionalidad principal del sistema, además de sus diferentes componentes y las herramientas usadas en su desarrollo; En el Capítulo 8 se da un resumen de la Versión Actual del Prototipo; En el Capítulo 9 se presentan las Conclusiones obtenidas de este desarrollo, las diferentes metas alcanzadas y dificultades encontradas, además del posible trabajo futuro a realizar para obtener un mejor resultado final; Y finalmente, en los anexos se presentan temas relacionados a la Gestión de Riesgos durante el desarrollo y métodos de Verificación y Validación de la plataforma.

Capítulo 2

Estado del arte

2.1. Marco teórico

Para comprender la solución propuesta, es necesario mencionar la disciplina de la Interacción Humano-Computador o HCI, por sus siglas en inglés. Esta área de estudio se centra en el diseño de interfaces, las cuales permiten a usuarios operar diversos sistemas. En el contexto de la obtención de datos, tradicionalmente se requiere de una persona con conocimiento técnico que actúe como intermediario entre usuario y sistema.

Con esto en mente, el presente trabajo busca reemplazar este proceso, introduciendo una nueva capa en la interacción que permita a usuarios sin conocimiento técnico comunicarse con la base de datos directamente, como si se tratara de otra persona.

Para lograr esto, el diseño se apoya en la Inteligencia Artificial, una rama de la informática que busca desarrollar sistemas capaces de realizar tareas que tradicionalmente requieren de una persona, tales como razonar, escribir e incluso procesar imágenes o videos. En el contexto de este trabajo, la Inteligencia Artificial actúa como el motor que permite la traducción de lo que busca el usuario a un lenguaje que la base de datos pueda entender, reemplazando la necesidad de que un humano escriba las consultas SQL.

Dentro de la Inteligencia Artificial, el Procesamiento de Lenguaje Natural o NLP (Natural Language Processing) es un campo que data de fines de la década de los cuarenta [1], el cual se encarga de permitir que un sistema entienda el contenido e intención de un mensaje enviado por un humano.

El desarrollo del NLP ha sido un proceso evolutivo durante el cual han habido diferentes arquitecturas en boga, una de las que destacan durante este proceso son las Redes Neuronales Convolucionales o CNNs (Convolutional Neural Networks). Aunque su función original fue en el campo de la visión artificial, las CNNs fueron adaptadas para el problema de NLP [2], teniendo buenos resultados al identificar patrones y extraer características de las oraciones. Sin embargo, esta estructura tenía dificultades al recordar qué se ha leído y procesado, por lo que la tecnología debió evolucionar hacia modelos capaces de manejar frases completas y de mayor complejidad.

El año 2017 [3] aparece un avance definitivo para el procesamiento de lenguaje, la arquitectura de Transformadores o Transformers. A diferencia de los modelos previos que “leían” de forma secuencial, el Transformador procesa las palabras en paralelo, extrayendo la relevancia de cada palabra en el texto a analizar, a esto se le llama auto-atención o self-attention, una función matemática que permite al modelo evaluar y asignar un peso a cada palabra de una oración en relación con todas las demás, sin importar su posición en la secuencia. Como consecuencia de esto, a diferencia de arquitecturas previas la arquitectura de Transformador puede paralelizar el cómputo de cada palabra, lo que permite reducir el tiempo de procesamiento.

Finalmente, la integración a gran escala de la arquitectura de Transformador y el mecanismo de auto-atención da origen a los Modelos Extensos de Lenguaje o LLM, Large Language Model en inglés. Estos sistemas son entrenados con cantidades masivas de texto y poseen las capacidades avanzadas de razonamiento requeridas para enfrentar los objetivos de este trabajo, tomando el rol del núcleo de la plataforma propuesta. Los LLM han demostrado una gran habilidad en tareas de traducción y síntesis de información. De este modo, los LLM operan como el mediador definitivo entre el humano y el sistema, cumpliendo el objetivo general de democratizar el acceso

a los datos.

2.2. Estado del arte

En [4], se crea un sistema con GUI y manejo de errores, donde el usuario puede ver los contenidos de una tabla e ingresar una query en sql o en lenguaje natural. El usuario debe iniciar sesión en el sistema, puede hacer preguntas que involucren la obtención, ingreso, actualización y eliminación de datos. La frase ingresada por el usuario pasará por una tokenización, lo que significa que cada palabra de la frase será descompuesta en subsecciones, o tokens, y luego pasarán por un análisis léxico, donde la lista tokenizada se mapea con un diccionario, por lo que estas palabras son reemplazadas por las palabras del diccionario relacionadas a la base de datos, luego se analiza sintácticamente, donde cada palabra tokenizada se mapea con atributos de otro diccionario, y por último se pasa por un análisis semántico, donde el sistema encontrará palabras que representen condiciones o símbolos y esas palabras se mapearán con el diccionario. El resultado es una consulta que luego puede ser ejecutada en la base de datos.

Se utilizan distintas estrategias de ingeniería de prompt, en [5] se estudian empíricamente varias LLMs de código abierto para destacar el potencial de estas en in-context learning y supervised fine-tuning con varias representaciones. Se demuestra que similar al aprendizaje in-context, la estrategia de representación es crítica para supervised fine-tuning. También luego de fine-tuning se observa una disminución en las capacidades de aprendizaje in-context. La solución integrada propuesta por el paper, DAIL-SQL, tiene un 86.6 % de exactitud en la ejecución y tiene el primer lugar en el spider leaderboard.

En [6] se plantea que la capacidad de comprensión de un LLM no es suficiente cuando se le proporciona únicamente el esquema de la base de datos y la consulta que se desea realizar, ya que el resultado tiende a ser impreciso. Por esta razón, se propone un framework llamado Knowledge-to-SQL, que utiliza un LLM experto en datos (DELLM). Esto se debe a que el desarrollo de una arquitectura no humana capaz de

generar información experta a partir de un conjunto de datos aporta un gran valor. Los desafíos que se presentan en la generación de conocimiento experto incluyen la especialización de la consulta y la base de datos, la consciencia del contenido de la base de datos y la mejora del rendimiento esperado. Se introduce PLDBF (Preference Learning via Database Feedback), que servirá para proporcionar recompensas a DELLM en función del grado en que el conocimiento generado ayuda a la recuperación de datos precisos y asiste a los LLM en la producción de consultas SQL más precisas. El método propuesto consta de tres módulos principales: el módulo de ajuste fino supervisado de DELLM, que generaría conocimiento basado en la consulta del usuario y el esquema de la base de datos; el módulo de retroalimentación; y el módulo de aprendizaje por preferencias con PLDBF.

En [7] se introduce un framework multiagente basado en LLM, que explota esto con diferentes funcionalidades para que se haga un análisis efectivo de text-to-SQL. MAC-SQL comprende un agente que funciona como descomponedor principal para la generación de text-to-SQL, que, junto a dos agentes auxiliares, selector para el uso de herramientas y refinador para la refinación de SQL. El descomponedor toma una pregunta y la separa en sub-preguntas más simples y las resuelve progresivamente mediante razonamiento encadenado. El selector, cuando el caso lo amerita, reduce el tamaño de la base de datos, donde se crea una sub-base para que la información irrelevante no interfiera. El refinador por último utiliza herramientas externas para la ejecución de SQL, recibe retroalimentación y refina las consultas que considere erróneas. Con GPT-4 como LLM principal para todas las tareas de los agentes y el uso de BIRD y Spider como conjuntos de datos para evaluar el desempeño del framework. Los datos experimentales muestran una precisión de ejecución de 59,59 en el conjunto de pruebas de retención de BIRD al utilizar MAC-SQL-GPT-4, esta es la mejor marca en el momento que se escribe el artículo. El artículo también habla de como afinaron un modelo basado en instrucciones, SQL-Llama, todo con el uso de datos de instrucciones de agentes de MAC-SQL, lo que ayuda en la simplificación de bases de datos, descomposición de preguntas, generación de SQL y corrección de SQL.

En [8] se propone supervised fine tuning (SFT) como otra opción al fine tuning regular para entrenar LLM para una mejor tarea de generación de text-to-SQL. Se utiliza Llama-V2 por sus distintas ventajas arquitectónicas, el modelo alcanza resultados de primer nivel en el conjunto de datos Spider, con una notable precisión de ejecución del 89,6 %, junto con una precisión de coincidencia exacta del 86,8 %. El documento explica las técnicas de entrenamiento para LLMs, como el few shot prompting, fine tuning o el supervised fine tuning, donde se explica por qué las dos primeras opciones se quedan atrás frente a la última en cuanto complejidad de consultas o cuando se trabaja con modelos pequeños respectivamente. Por otro lado, supervised fine tuning utiliza datos etiquetados. Se pasa todo el contexto a la vez, pero la pérdida final se figura solo sobre la etiqueta que el modelo debe generar. Esto permite que el modelo aprenda solo la generación sintáctica de la etiqueta en lugar de toda la declaración.

En [9] se propone un enfoque que aprovecha múltiples prompts para generar diversas respuestas candidatas y las agrega de manera efectiva. El proceso de generación de SQL comprende tres pasos: Schema linking, Multiple SQL Generation y Selection. Schema linking implica identificar las tablas y columnas relevantes de una base de datos para convertir una pregunta en lenguaje natural en una consulta SQL. La introducción de schema linking ha mejorado significativamente el rendimiento tanto de los enfoques basados en ajuste fino como de los basados en ICL. Este paso consta de dos subpasos, extracción de tablas relacionadas con la pregunta en lenguaje natural (Table Linking), para después extraer las columnas necesarias dentro de esas tablas (Column Linking). Se emplean múltiples prompts en ambas fases con el objetivo de lograr un alto nivel de recuperación. Multiple SQL Generation es generar varias consultas SQL basadas en múltiples prompts distintos. Se generan múltiples prompts donde varía tanto el método de selección de los ejemplos de few shot como el orden en que se presentan, lo que asegura una exploración más amplia de posibles consultas SQL, ya que se ha descubierto que hacer esto puede diferir considerablemente en la salida de la LLM. Selection busca la consulta más precisa entre las consultas candidatas. El conjunto de candidatos se filtra en función de los puntajes de confianza, y luego se asigna al LLM la tarea de

seleccionar la consulta más precisa entre el conjunto refinado.

En cuanto a técnicas de entrenamiento, destacan las GAN, o generative adversarial networks, esta técnica se basa en dos modelos que trabajan a la par, los cuales compiten entre ellos.

Estos se tratan de un modelo, el generador, el cual construye información sintética en base a retroalimentación enviada por el otro modelo, el discriminador, este último recibe información real y sintética y decide de cual tipo se trata.

La meta detrás de este método es lograr que el discriminador tenga certeza respecto a cuál tipo de dato con el que interactúa, mientras que el generador busca engañar al discriminador. Por esto, esta técnica se destaca en cuanto a entrenamiento no supervisado o semi-supervisado sin requerir de grandes cantidades de datos de entrenamiento.

En cuanto a alternativas para esta configuración, en [10] se muestran diversas opciones, las cuales posteriormente se comparan. Entre estas, se encuentran FCGANs, GANs convolucionales, LAPGANs, DCGANs, GANs condicionales, infoGANs, y GANs con modelos de interferencia. Estos últimos utilizan dos tipos de redes, encoders (de interferencia) y decoders, ambas cumplen el rol del generador.

Además, se muestra que el entrenamiento se basa en encontrar los parámetros correctos del discriminador para hacer la clasificación mas precisa, y encontrar parámetros del generador para confundir al discriminador.

Por otro lado, se entrega información útil para el entrenamiento, como el orden en que los parámetros de cada modelo se deben actualizar, o que para cada generador hay un discriminador óptimo.

Adicionalmente, en [11] se muestra un caso de uso específico de entrenamiento mediante GANs el cual utiliza modelos de lenguaje. En este trabajo se utilizan las facultades de procesamiento de los modelos de lenguaje para la tarea de recomendar trabajos a postulantes según el currículum que estos ingresan a una plataforma.

Se hace hincapié a la necesidad de extraer más información que la disponible explícitamente en los currículum, mediante el uso generador para refinar documentos de baja calidad a contra partes similares más detalladas. Esto se realiza debido a la

necesidad de los modelos de lenguaje de grandes cantidades de información con el fin de evitar delirios o resultados fabricados, una de las grandes falencias de los LLMs.

Este documento también presenta en detalle el proceso usado para establecer la red, se explicitan parámetros del generador, discriminador y clasificadores, además del proceso de aprendizaje y la metodología de evaluación.

Finalmente, en [12] se muestra un proceso de optimización de consultas SQL generadas a partir de preguntas en lenguaje natural mediante LLMs, se entrega la premisa que el modelo genera diversas consultas SQL a partir de una entrada en NL, y que gran parte del tiempo la consulta seleccionada no es la que mejor se adecua a la pregunta de entrada.

Por esto, se establecen diferentes métricas las cuales son revisadas para cada potencial consulta SQL generada, las cuales una vez son comparadas, resultan en una mejora en el desempeño del modelo utilizado.

Capítulo 3

Planificación del desarrollo

En el siguiente capítulo, se definen los perfiles de los potenciales usuarios que utilizarán el sistema, sus funcionalidades y características, los requisitos de su interfaz y las herramientas utilizadas en su desarrollo.

3.1. Perfiles de usuarios

En el cuadro 3.1 se muestran los distintos perfiles de usuarios que podrían utilizar la plataforma. Todos estos comparten que son personas empleadas por FALP con entre 25 y 65 años de edad.

Perfil	Ocupacio- nal	Psicológico	Otros
Usuario.	Cualquier ocupación dentro de la fundación.	Desea emplear la solución.	Usuario sin privilegios.
Analista.	Parte del área de datos o inteligencia artificial.	Desea emplear la solución y está interesado(a) en su desempeño.	Usuario con algunos privilegios.
Administrador.	Parte del área de inteligencia artificial.	Desea emplear la solución y esta Interesado(a) en su desempeño. Además, participa en su administración.	Usuario con más privilegios.

Cuadro 3.1: Perfiles de usuario.

3.2. Requisitos del sistema

Habiendo considerado los diferentes perfiles de los potenciales usuarios, en la siguiente sección se detallan las características del sistema y su desarrollo necesarias para solventar la problemática.

En particular, se abordarán: requisitos funcionales, los cuales describen las funcionalidades del sistema; requisitos no funcionales, que definen atributos del comportamiento del sistema; de interfaces, que plantean las interacciones entre el usuario y el sistema; y los requisitos de ambiente, los cuales describen las herramientas utilizadas durante la elaboración del proyecto.

3.2.1. Requisitos funcionales

En el cuadro 3.2 se describen los diferentes requisitos funcionales definidos, estos corresponden a las funcionalidades que deben ser parte de la solución final.

Nombre	Descripción
RF1	El sistema debe recibir consultas en lenguaje natural y generar una consulta SQL equivalente.
RF2	El sistema debe generar un documento que contenga la información obtenida a partir de la consulta SQL creada.
RF3	La plataforma y API deben manejar sesiones y credenciales.
RF4	El sistema debe almacenar y desplegar las interacciones entre los funcionarios y el chatbot.
RF5	La plataforma debe ofrecer encuestas opcionales de satisfacción, relativas a la conversacion del chatbot con el usuario.
RF6	La plataforma debe almacenar y desplegar las encuestas de satisfacción y métricas del chatbot.
RF7	La plataforma debe contar con control de privilegios por usuario y permitir a usuarios con permisos la administración de roles.

Cuadro 3.2: Requisitos funcionales

3.2.2. Requisitos no funcionales

En el cuadro 3.3 se describen los diferentes requisitos no funcionales del sistema, estos corresponden a las características que debe tener la plataforma, tales como el tiempo de respuesta y cualidades de su código.

Nombre	Descripción
RNF1	La plataforma debe ser segura y manejar sesiones.
RNF2	El código debe ser escalable, hacer uso de buenas prácticas de programación y tener un formato claro.
RNF3	La plataforma debe tener tiempos de respuesta acordes a la tarea solicitada.
RNF4	La plataforma debe ser de uso intuitivo para cualquier usuario. Esto se logrará mediante una navegación intuitiva y accesible.
RNF5	La plataforma debe ser mantenible. Esto se logrará mediante el control de versiones y una documentación clara.

Cuadro 3.3: Requisitos no funcionales

3.2.3. Requisitos de interfaces

En el cuadro 3.4 se definen las diferentes acciones que puede tomar el usuario, y sus respectivas definiciones, parámetros y respuestas. Estas últimas se definen a más detalle en el cuadro 3.5.

Evento	Descripción	Iniciador	Respuesta
Consulta NL.	El usuario ingresa una consulta en lenguaje natural para obtener información de la base de datos.	Presionar tecla Enter.	Resultado consulta.
Inicio de sesión.	El usuario ingresa sus credenciales para acceder a la plataforma.	Clic en botón “Sign in” una vez llenados los campos.	Resultado inicio sesión.
Ingreso vista de registro de conversaciones.	El usuario desea ingresar al registro de conversaciones.	Clic en botón de navegación “Conversations”.	Resultado cambio de vista.
Ingreso vista de chatbot.	El usuario desea ingresar a la vista de chatbot.	Clic en botón de navegación “Home”.	Resultado cambio de vista.
Ingreso vista de métricas de satisfacción del cliente.	El usuario desea ingresar a la vista de métricas de satisfacción del cliente.	Clic en botón de navegación “Metrics”.	Resultado cambio de vista.
Cierre de sesión.	El usuario desea cerrar sesión.	Clic en botón “Log out”.	Resultado cierre de sesión.

Cuadro 3.4: Eventos externos

Respuesta	Descripción	Parámetros
Resultado consulta.	Se entrega resultado de consulta ingresada por usuario en el formato que este solicite.	Consulta NL.
Resultado inicio sesión.	Verifica las credenciales del usuario y, si son válidas, permite el ingreso a la plataforma.	Credenciales de acceso.
Resultado cambio de vista.	Verifica permisos del usuario. En caso de contar con los permisos necesarios, muestra la vista solicitada.	Tipo de usuario, nombre de vista.
Resultado cierre de sesión.	Cierra la sesión del usuario.	N/A

Cuadro 3.5: Respuestas del sistema

3.2.4. Requisitos de ambiente

En esta sección se definen las herramientas utilizadas en el desarrollo, tanto las especificaciones de los equipos físicos como el software a usar.

Hardware de desarrollo

Los computadores utilizados en el desarrollo tienen, como base, las siguientes especificaciones:

- **OS:** Microsoft Windows 11 Home.
- **Procesador:** Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz.
- **Memoria física instalada (RAM):** 8,0 GB.

Software de desarrollo

Dentro del software de desarrollo utilizado tenemos las siguientes herramientas:

- Python.
- JupyterHub.
- Amazon Bedrock (Modelos fundacionales).
- CSS.
- HTML.
- FastAPI.
- Flask.
- PostgreSQL.
- Docker.

- Auth0.
- Google API.

Capítulo 4

Diseño de arquitectura

Habiendo establecido qué funciones debe ofrecer la solución a desarrollar, en el siguiente capítulo se define la arquitectura del sistema y los módulos que lo comprenden.

4.1. Diagrama de contexto

En la figura 4.1 se muestra el diagrama de contexto donde se presentan las tres entidades que interactúan con el sistema. En esta figura se ve como: el usuario pide y recibe información del sistema; el módulo de LLM es el encargado de realizar la traducción de lenguaje natural a SQL; y la base de datos entrega la información solicitada.

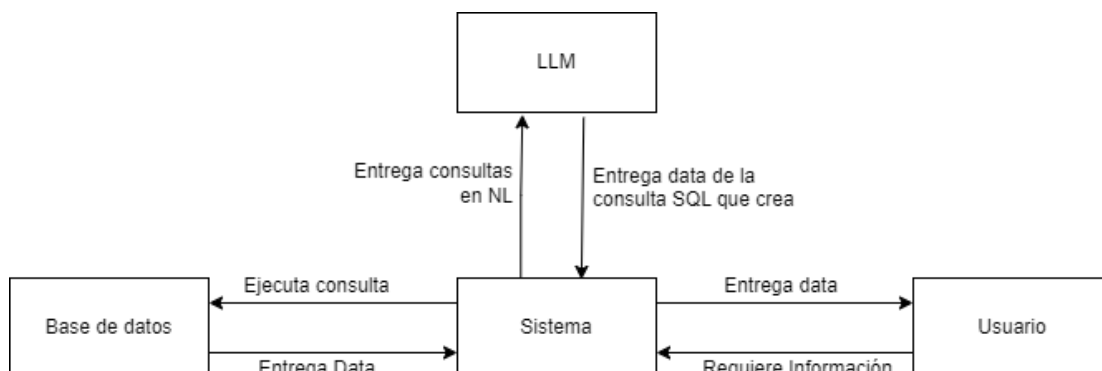


Figura 4.1: Diagrama de contexto

4.2. Diagrama de arquitectura

Habiendo establecido los actores y sus relaciones en el diagrama de contexto, se procede a definir una estructura a más bajo nivel. Esta se muestra en la figura 4.2, donde se ve el diagrama de arquitectura y los distintos módulos que la componen.

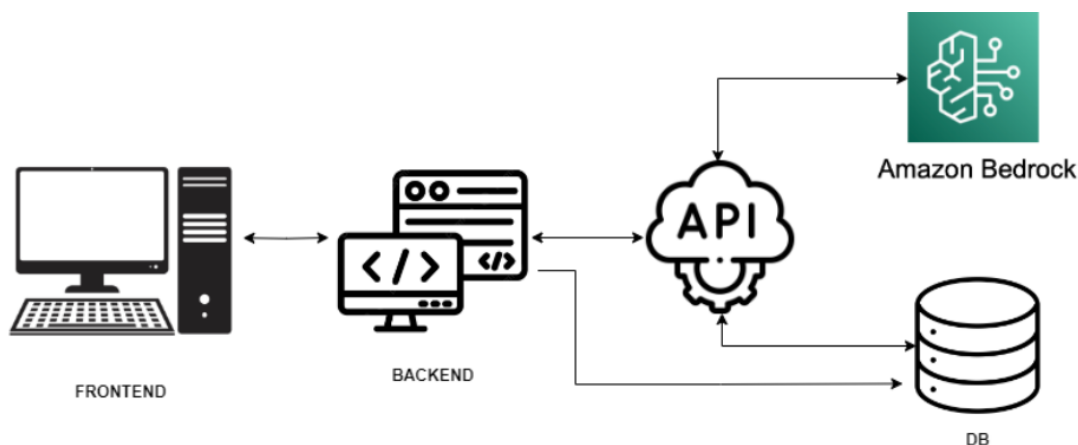


Figura 4.2: Diagrama de arquitectura

4.3. Definición de módulos

4.3.1. Frontend

- **Propósito:** Módulo encargado de manejar las interfaces.
- **Alcance:** En este módulo el usuario ingresa la consulta, la cual es enviada al backend. Además, el frontend recibe información del backend para desplegar vistas dinámicas. Las cinco vistas principales de la plataforma son: inicio de sesión, vista de consulta, vista de satisfacción de usuario, vista de conversaciones y vista de administración. Se realizará en Python, y se hará uso de Flask, un framework flexible, el cual es adaptable a distintos tipos de aplicaciones web además de ser ligero y escalable.
- **Dependencias:** Únicamente dependiente del backend.

4.3.2. Backend

- **Propósito:** Módulo encargado de manejar la lógica del sistema, el cual recibe y retorna información recibida desde los módulos adyacentes.
- **Alcance:** Este módulo se encarga de recibir las distintas peticiones por parte del usuario a través del frontend, logra esto al hacer un llamado a la API o base de datos para obtener la data requerida. Su responsabilidad radica en el inicio de sesión, administración de usuarios, manejo de privilegios y solicitud de datos a la API. Se realizará en Python y también utilizará el framework Flask, de manera que este será utilizado tanto para el backend como el frontend.
- **Dependencias:** Entradas provenientes del frontend, conexión con la API.

4.3.3. API

Definición del módulo

- **Propósito:** Módulo encargado de la interacción entre backend, LLMs y BBDD.
- **Alcance:** Este módulo se encarga de interactuar de manera segura con las BBDD, mantener una conexión con las LLMs y mantener conexión con el backend, de forma que permite la interacción entre los tres módulos. Se estiman aproximadamente 100 conexiones simultáneas como máximo. Tendrá autenticación y límite de conexiones en un intervalo de tiempo. Se desarrollará en python y se utilizará el framework FastAPI, debido a que el trabajo se realizará en un entorno de JupyterLab entregado por el cliente. Este framework facilita la generación de documentación, permite un alto rendimiento y trabajo asíncrono, lo que permite escalar. Utiliza modelos de datos que son validados o convertidos en caso de no ser el tipo de dato requerido, tiene una documentación extensa y es flexible.
- **Dependencias:** Este módulo depende del buen desempeño de módulos como la LLM, BBDD y el backend.

4.3.4. Modelos de lenguaje / Amazon Bedrock (LLMs)

Definición del módulo

- **Propósito:** Este módulo se encarga de la lógica con la que se manipulan los mensajes del usuario. Identificando la naturaleza de los mensajes; Generando o corrigiendo consultas SQL; Traduciendo información de la base de datos a lenguaje natural; Generando archivos o gráficos; Y orientando al usuario a realizar preguntas sobre la base de datos del cliente. Para esto se utilizan modelos de la familia Claude los cuales según el tipo de tarea, trabajan con un conjunto de instrucciones predefinido para resolver lo que les llegue como entrada, ya sea del usuario o desde otro modelo. La estructura de comunicación de los modelos depende de las salida de los mismos, donde se tienen salidas que se retornan de inmediato al usuario, comunicación secuencial entre modelos o bucles con intentos definidos de manera interna. El enfoque de trabajo en desarrollo de prompts permite obtener una comparación en el desempeño de modelos frente a una misma tarea, lo cual facilita identificar cuellos de botella o fallos en la lógica de interacciones, facilitando el desarrollo.
- **Dependencias:** Este módulo depende de su interacción con la base de datos y con la API.

4.3.5. Base de datos (BBDD)

Definición del módulo

- **Propósito:** Módulo que almacena los datos utilizados en las consultas, además de la información referente a la satisfacción con las misma.
- **Alcance:** Este módulo se utiliza para la interacción con los modelos, para que estos conozcan la estructura de las tablas. Además, se le realizan las consultas generadas por los modelos de lenguaje. Se almacenará también la información de

sesiones para la plataforma, el almacenamiento de conversaciones y de métricas. Se utilizará el gestor PostgreSQL para el almacenamiento e interacción con los datos.

- **Dependencias:** Este módulo no presenta dependencias.

4.4. Desgloce de módulos

En el cuadro 4.1 se resume la información referente a los módulos descritos en la sección anterior. Además, se vincula la sección del documento donde cada uno se define a más detalle.

Módulo	Función	Sección
Frontend	Encargado de manejar las interfaces mostradas al usuario.	Sección 4.3.1
Backend	Encargado de manejar la lógica del sistema, gestiona la información que se envía o llega desde los módulos adyacentes.	Sección 4.3.2
API	Encargado de la interacción entre backend, LLMs y BBDD.	Sección 4.3.3
Modelos de lenguaje / Amazon Bedrock (LLM).	Encargado de la traducción de lenguaje natural a consulta SQL.	Sección 4.3.4
Base de datos (BBDD).	Encargado de almacenar la información a pedir, los datos referentes a las consultas y las respuestas a encuestas de satisfacción.	Sección 4.3.5

Cuadro 4.1: Módulos y definiciones

4.4.1. Matriz de relación entre módulos y requisitos funcionales

Por último, en el cuadro 4.2 se vincula cada módulo con los requisitos funcionales previamente descritos en el cuadro 3.2.

	Frontend	Backend	API	LLM	BBDD
RF1	X	X	X	X	
RF2	X	X	X		X
RF3	X	X	X		X
RF4			X		X
RF5	X	X	X	X	X
RF6	X	X	X		X
RF7	X	X	X		X

Cuadro 4.2: Matriz Requisitos funcionales y módulos

Capítulo 5

Diseño de interfaces

Habiendo aclarado los módulos que componen el sistema, se procede con el diseño e implementación de sus vistas relacionadas, comenzando con el flujo de las mismas y posteriormente el detalle de las diferentes versiones de la plataforma.

5.1. Modelo de navegación

En la figura [5.1](#) se muestra el modelo de navegación, este representa de forma visual cómo se navega la plataforma. En este, se incluyen los principales requerimientos del sistema, los cuales son: consultar datos con el chatbot (RF1 y RF2), gestionar sesiones (RF3 y RF7) la consulta de registros de conversaciones (RF4) y la consulta de métricas de satisfacción de usuarios (RF5 y RF6).

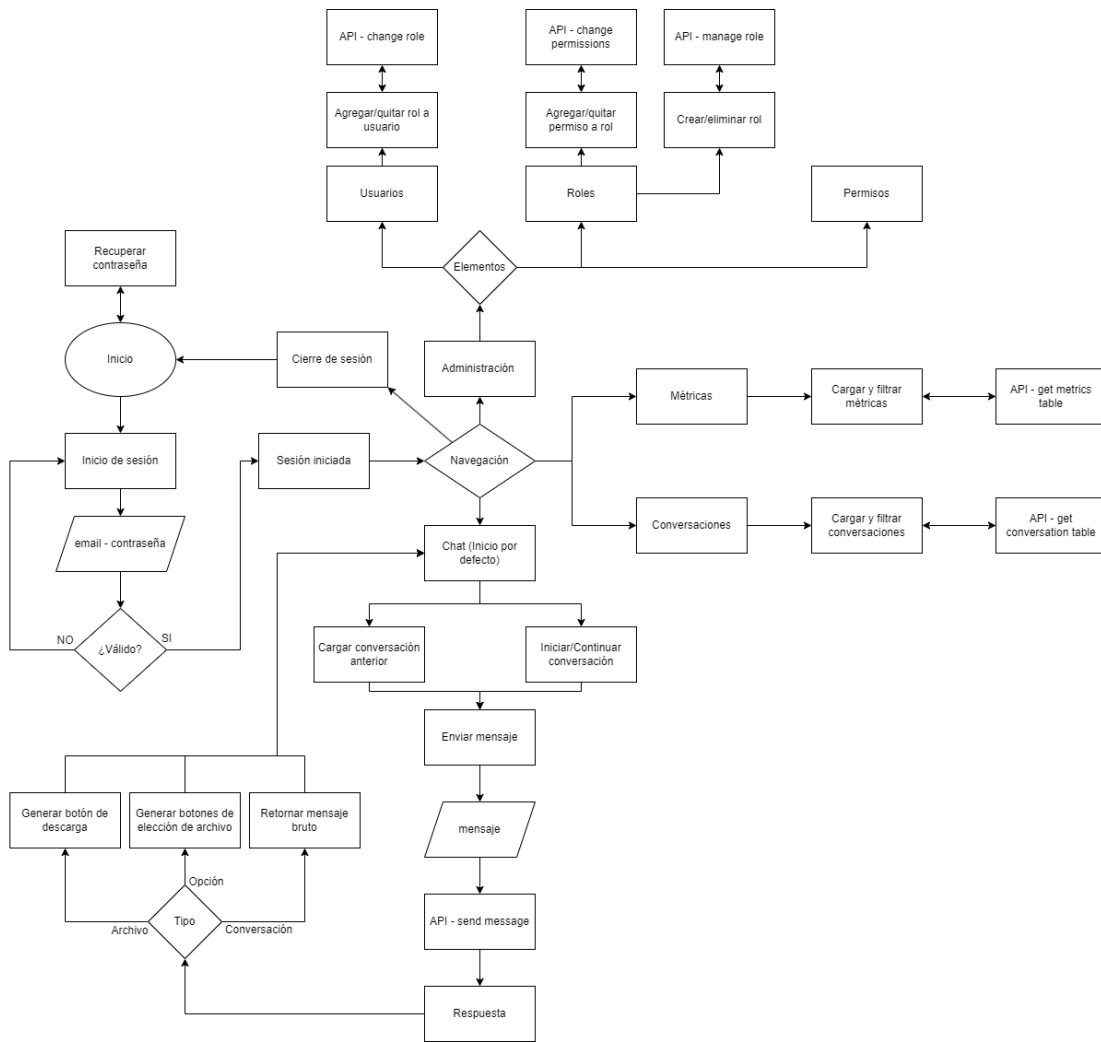


Figura 5.1: Modelo de navegación

5.2. Prototipo de interfaces de usuario

En esta sección se muestran las diferentes iteraciones de las interfaces. Los cambios a realizar entre cada versión se definen según reuniones con representantes de la fundación Arturo López Perez, los requerimientos previamente identificados y buenas prácticas de diseño de interfaces de usuario.

5.2.1. Primera iteración (V1.1)

En esta primera iteración se tiene la vista de inicio de sesión, que se muestra en [5.2](#), en esta el usuario puede ingresar sus credenciales para acceder a la plataforma. En la figura [5.3](#) se encuentra el menú principal, donde el usuario puede seleccionar qué función desea utilizar, en las figuras [5.4](#) y [5.5](#) se tienen las vistas de registros de conversaciones y de métricas de satisfacción, respectivamente. Por último, en la figura [5.6](#) se muestra la vista con el chatbot encargado de la traducción NL-SQL.

The image shows the FALP logo at the top, which consists of the word "falp" in a dark blue, lowercase, sans-serif font. Above the letter "a" is a yellow circular icon containing a stylized blue figure with arms raised. Below the logo is a light blue rectangular box with rounded corners. Inside this box, the word "Email" is followed by a white input field containing the text "email@example.com". Below that, the word "Password" is followed by a white input field containing a series of dots. A yellow "Sign in" button is positioned below the password field. At the bottom of the box, the text "Forgot password?" is displayed as a link.

Figura 5.2: Inicio de sesión

El diagrama muestra la interfaz de usuario de la aplicación, organizada en una estructura de contenedores. En la parte superior, hay una barra de navegación de color verde azulado que contiene un botón "CERRAR SESIÓN" en la esquina superior derecha. Debajo de esta barra, el contenido principal se encuentra dentro de un contenedor gris claro con bordes redondeados. Este contenedor principal está dividido en tres secciones rectangulares blancas con bordes redondeados y sombreado, distribuidas horizontalmente. La primera sección a la izquierda contiene el texto "CONSULTAR DATOS". La sección central contiene el texto "REGISTROS DE CONVERSACIONES". La tercera sección a la derecha contiene el texto "MÉTRICAS DE SATISFACCIÓN".

Figura 5.3: Menú principal

[illegible]

Figura 5.4: Registros de conversaciones

[illegible]

Figura 5.5: Métricas de satisfacción

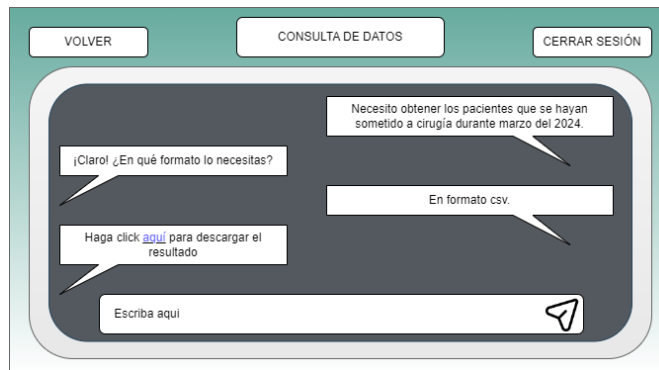


Figura 5.6: Consultar datos

En el cuadro [5.1](#) se aprecian las principales interacciones con el prototipo de plataforma.

5.2.2. Segunda iteración (V1.2)

Similar a la primera versión, se tienen la vista de inicio de sesión (figura [5.7](#)), la vista de registros de conversaciones (figura [5.9](#)) y la vista de métricas de satisfacción (figura [5.10](#)). Sin embargo, en esta iteración el menú principal se fusiona con la vista de chatbot, esto resulta en la vista de menú principal - chatbot que se ve en la figura [5.8](#).

En el siguiente [enlace](#) se puede acceder a una versión interactiva del prototipo.



Figura 5.7: Inicio de sesión



Figura 5.8: Menú principal - Chatbot

Id conversación	Id usuario	Conversación	Mensaje inicio	Consulta generada	Otros atributos
1	456	[{"actor": "user", "m...	¿Cuántos pacientes	SELECT COUNT(*)	...

Figura 5.9: Registros de conversaciones

Id métrica	Puntuación	Comentario	Fecha	Id usuario	Otros atributos
1	8.4	Le falta entregar...	2024-09-10	456	...

Figura 5.10: Métricas de satisfacción

5.2.3. Tercera iteración (V1.3)

En esta iteración se ha añadido una barra lateral que permite ver los mensajes anteriores dentro de una misma sesión. También, se agrega una ventana de administración que aun está en desarrollo, pero tiene como objetivo gestionar los permisos de los distintos usuarios para manejar quién tiene acceso a métricas y conversaciones de otros usuarios. Estos cambios se evidencian en la figura [5.11](#).

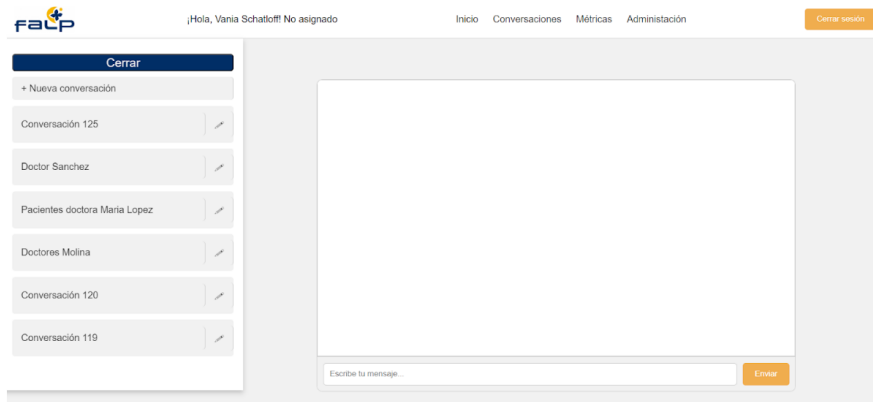


Figura 5.11: Vista principal con barra lateral y sección de administrador

5.2.4. Iteración final (V1.4)

La última versión implementada de la interfaz gráfica se muestra en la sección [7.4.6](#), referente a la GUI final del sistema.

5.3. Definición de eventos y respuestas de las interfaces

En el cuadro 5.1 se describen las principales interacciones con el prototipo de plataforma y sus respuestas.

Evento	Acción y objeto afectado	Respuesta
Inicio de sesión	Hacer clic al botón “Sign In”.	Redirección a vista principal.
Recuperación de contraseña.	Hacer clic al botón “Forgot Password?”.	Redirección a vista de recuperación de contraseña.
Cierre de sesión.	Hacer clic al botón “Log Out”.	Redirección a vista de inicio de sesión.
Consulta de datos.	Hacer clic al botón “Home”.	Redirección a vista de chatbot.
Consulta registros de conversaciones.	Hacer clic al botón “Conversations”.	Redirección a vista de consulta de registros de conversaciones.
Consulta de métricas de satisfacción.	Hacer clic al botón “Metrics”.	Redirección a vista de consulta de métricas de satisfacción.
Filtración de datos.	Hacer clic al botón “Filtrar”.	Filtrado de resultados de registros de conversación o de métricas de satisfacción.

Cuadro 5.1: Eventos y respuestas prototipo

Capítulo 6

Diseño de motor traductor NL-SQL

Para el desarrollo del chatbot y las respuestas de este, se hace uso de diferentes instancias de modelos de lenguaje de gran tamaño, los cuales cumplen distintas funcionalidades para completar diversos flujos, estos se traducen en soluciones para las necesidades y requerimientos del cliente.

Con este fin, se utilizan los modelos provistos por AWS-Bedrock que se tienen a disposición durante el desarrollo, con la ayuda de Langchain como framework para la integración, configuración y llamado de los modelos. Así, se implementan diferentes invocaciones para las distintas tareas que se requieren para generar las respuestas para cada uno de los casos que cubre el chatbot.

6.1. Estructura de instrucciones

Las instrucciones dadas a los modelos de lenguaje utilizados se entregan mediante prompts, conjuntos de directivas que se dividen en dos secciones en Langchain: SYSTEM, en la cual se entrega información de la naturaleza de las respuestas que el modelo debe generar; y USER, sección en que se entregan las preguntas del usuario.

Los prompts en cuanto al sistema siguen una misma estructura basada en la investigación previa realizada: Estos comienzan con información de contexto para el modelo, donde se explicitan el rol y las tareas que va a cumplir. A continuación, se utiliza

aprendizaje en contexto, conocido en inglés como in-context learning, para preparar al modelo con ejemplos de preguntas esperables y sus respectivas salidas. Finalmente, se indican instrucciones adicionales o restricciones que el modelo debe seguir.

Todos los modelos utilizan esta lógica con leves modificaciones para casos particulares donde el modelo generaba respuestas de manera imprecisa, en estos casos se recalcan instrucciones dadas previamente.

6.2. Implementación de memoria en los modelos

Los modelos cuentan con una memoria basada en el historial de la conversación, de esta manera, el LLM es capaz de entender preguntas que no propician toda la información necesaria, ya que se basa en el contexto de la conversación para seguir respondiendo al usuario.

Con este fin, se utilizan las herramientas entregadas por Langchain para manejar la memoria del modelo. Cabe destacar que durante el desarrollo se utilizó un método actualmente obsoleto, este entrega la conversación en su totalidad como la memoria del modelo. Actualmente, existen diferentes formas de manejar la memoria dentro de Langchain, siendo la más atractiva el grupo de funciones de memoria de resumen o summary memory, estas hacen uso de un modelo de lenguaje secundario que disminuye el volumen de información, pero mantiene los aspectos más relevantes en la memoria.

Por esto, si bien la memoria utilizada es funcional y brinda un comportamiento a los modelos que mejora la experiencia del usuario, en su estado actual hace uso de más tokens de los necesarios, lo que se traduce en un mayor costo operacional.

6.3. RAG y modelo de incorporación

Debido a que durante el desarrollo se trabaja con un gran número de tablas de la base de datos del cliente, las cuales se entregan al modelo como archivos de texto, se opta por implementar en varias de las invocaciones a modelos lo que se conoce como

generación aumentada por recuperación o RAG, por sus siglas en inglés.

Su función consiste en identificar los documentos más relevantes a la consulta entre los diferentes documentos que se tienen, con el fin de que el LLM obtenga únicamente la información necesaria para poder generar una respuesta coherente a la pregunta recibida, disminuyendo los tokens de entrada y asegurando que el modelo cuente con toda la información necesaria.

La obtención de documentos se realiza mediante un modelo de incorporación o de embeddings, este se encarga de convertir los documentos en múltiples vectores n-dimensionales, los cuales compara con la información entregada por el usuario, mediante similitud entre vectores – en particular semejanza del coseno – para identificar los documentos más similares y, por ende, relevantes.

Esto otorga múltiples beneficios, en particular:

1. El prompt es mucho más preciso, lo que disminuye la probabilidad de obtener una respuesta no deseada.
2. La cantidad de tokens de entrada disminuye al no incluir toda la información de los documentos cada vez que se hace un llamado al modelo, lo que se traduce en un ahorro a nivel computacional y económico.
3. Una amplia capacidad de adaptabilidad y actualización de la información, ya que se pueden agregar documentos sin la necesidad de entrenar o agregar más detalles en las instrucciones.

En el desarrollo, esto se logra gracias a que se encuentra implementado en un índice de la biblioteca Faiss, además de ser uno de los métodos que mejor se ajusta a nuestro caso, debido a la necesidad de procesamiento rápido y exacto, y la cantidad relativamente reducida de vectores a comparar.

Más a profundidad, la similitud coseno es un método que permite determinar la similitud de dos vectores n-dimensionales, esta se calcula $\frac{u \cdot v}{|u||v|}$ con u y v vectores. Uno de estos corresponde a los embeddings de uno de los documentos a comparar y el otro

al vector de embeddings de la pregunta. Debido a que la fórmula utiliza los módulos de los vectores, trabajar con estos normalizados nos entrega una métrica de comparación independiente del tamaño de estos. Por esto, su valor se encuentra en el intervalo $[-1, 1]$, con 1 siendo un vector idéntico, 0 un vector ortogonal y -1 un vector opuesto.

En cuanto al radio de búsqueda utilizado por el índice de Faiss, este entrega vectores que se encuentren dentro del radio de búsqueda entregado. A modo de ejemplo, asumiendo un caso en dos dimensiones, el radio de búsqueda se traduce en una circunferencia con centro en el vector al que se le buscan documentos relacionados, o, análogamente en un caso tridimensional, sería un cono de búsqueda, esta idea se extiende al caso n-dimensional de los embeddings.

6.4. Flujo principal

El flujo principal de la aplicación tiene como objetivo el responder las preguntas relacionadas a la base de datos del cliente que el usuario haga por medio del chat en la vista principal.

En el desarrollo se hace uso de la estrategia divide y conquistarás, la cual se traduce en un sistema de varios modelos comunicándose unos con otros de manera secuencial, cada uno de los cuales cuenta con una tarea específica, lo que resulta en respuestas más precisas con respecto a respuestas.

El sistema también contempla escenarios alternativos frente a preguntas que no son las esperadas, teniendo así salidas para casos en donde la pregunta no tiene relación, se está pidiendo algo que no existe en la base de datos, la consulta SQL genera un error o se solicitan archivos o gráficos cuando la conversación aún no ha generado información, entre otros.

A continuación, se describen los diferentes modelos que conforman el módulo de traducción:

6.4.1. Identificador general

Este identificador tiene como función categorizar el mensaje inicial del usuario, para así saber qué hacer con el mensaje. Esta invocación utiliza un RAG para mejorar la precisión en la identificación.

- Si categoriza como Consulta, significa que el modelo entiende que se está realizando una pregunta relacionada con los datos que tiene la base de datos del cliente, este arroja como salida “SQL”, lo que de manera interna envía el mensaje al modelo experto SQL.
- Si categoriza como archivo, significa que el mensaje que recibió fue una solicitud para obtener un archivo en formato CSV o XLSX. Esto genera como salida el nombre del formato que indicó el usuario, y de manera interna se genera el archivo, el cual llega al usuario para poder descargar, además de ser almacenado en la base de datos.
- Si categoriza como gráfico, significa que el mensaje que recibió fue una solicitud para obtener un gráfico con la información que se recibió previamente. Lo que arroja como salida es la palabra “graph” para entrar al flujo de la generación de gráficos, el cual se retorna al usuario.
- Si categoriza como conversación, el chatbot responde con lenguaje natural de manera directa al usuario, orientando a hacer una pregunta relacionada con la base de datos del cliente.

6.4.2. Experto SQL

El modelo experto SQL tiene como función generar una consulta utilizando sintaxis de PostgreSQL en base a la pregunta que hizo el usuario.

Para construir una consulta SQL exitosa utilizando el nombre de tablas, columnas y valores de la base de datos, se implementa este modelo con un RAG, en el cual

cada documento corresponde a una tabla de la base de datos del cliente, en la cual se indica cada columna y ejemplos de los valores que contienen estas. La salida de esta invocación es una consulta SQL o texto en lenguaje natural indicando que la consulta no tiene relación con los datos del cliente. Esta respuesta es entregada al siguiente identificador.

6.4.3. Identificador NL/SQL

El identificador NL/SQL tiene como función clasificar el mensaje que entrega el Experto SQL

- Si el modelo responde “NL”, significa que la respuesta del Experto SQL fue texto en lenguaje natural, lo que indica que no era posible hacer una consulta relacionada con la base de datos del cliente.
- Si del modelo responde “SQL”, significa que la respuesta del Experto SQL fue exclusivamente una consulta SQL sin ningún tipo de texto en lenguaje natural.

Esta invocación no utiliza un RAG al tener una función simple y no requiere de nada más que la salida del modelo anterior como parámetro de entrada.

6.4.4. Corrector SQL

El corrector SQL tiene como función arreglar las consultas SQL generadas por el experto SQL que generan un error en la base de datos. Esta invocación tiene un prompt similar al Experto SQL, salvo que este tiene como argumento el error generado por la base de datos, por lo que tiene una noción de qué es lo que falla en la consulta.

La llamada a este modelo se encuentra dentro de un bucle que se repite mientras la consulta resulte en errores y no se haya completado un número de intentos fallidos, esto con el fin de no utilizar recursos en una tarea que no se podrá completar. En caso de que el modelo no sea capaz de arreglar el problema en este número de iteraciones,

el flujo indica que la respuesta no pudo ser respondida e invita al usuario a contactar a un administrador.

6.4.5. Traductor Data a NL

El traductor es el último modelo del flujo principal, su función es traducir la información que se recibe de la base de datos a texto en lenguaje natural, que es lo que finalmente llega al usuario. La respuesta de este modelo depende de la cantidad de tokens que tenga la respuesta de la base de datos, esto para evitar un envío de información masiva que genere un alto costo y una respuesta poco amigable para el usuario.

- Si la respuesta es nula por parte de la base de datos, se indica que no se encontró información sobre la pregunta hecha
- Si la información de la base de datos es mayor a una cantidad arbitraria de tokens (500 en el prototipo), se responde indicando que se obtuvo la información, pero que esta es muy extensa, por lo que no se puede mostrar por mensaje. Sin embargo, el usuario puede solicitarla en formato CSV, XLSX o generar un gráfico
- Si la información es menor a una cantidad de tokens, 500 en el prototipo, el modelo responde en lenguaje natural haciendo un análisis de la pregunta, la consulta y la data obtenida para así generar una respuesta lo más congruente y completa posible, indicando de todas maneras que puede solicitar obtener la información en un archivo XLSX o CSV, así como también generar un gráfico.

6.5. Flujo de gráficos

El flujo de gráficos corresponde a un caso del flujo principal que es utilizado cuando el usuario solicita un gráfico a partir de información obtenida por el flujo principal. Este flujo hace uso de una o dos nuevas invocaciones para poder generar un gráfico básico, dependiendo del caso.

Para la implementación de este subflujo se consideran 3 posibles casos:

1. Cuando no hay información previa en la conversación, donde el identificador responde directamente al usuario indicando que no hay información que generar un gráfico
2. Cuando se posee toda la información para generar el gráfico que el usuario solicita, esto de manera interna llama al “reconocedor gráfico” el cual extrae toda la información para entregarla al “generador gráfico” con el propósito de construir la figura.
3. Cuando no se posee toda la información para poder generar el gráfico, en este caso se llama al generador gráfico internamente, pero este en su campo de “falta de datos” envía un valor “True”, lo que dirige la consulta al modelo Experto SQL orientado a gráficos, el cual genera una nueva consulta que es procesada como si fuese el flujo principal para obtener, así, la información necesaria para generar el gráfico. Posterior a esto, se hace un nuevo llamado al reconocedor gráfico, el cual ahora cuenta con toda la información y le pasa los parámetros a la función para generarlo.

6.5.1. Recolector

El recolector gráfico tiene como función generar un archivo JSON con toda la información importante para generar el gráfico que solicita el usuario, si es que el usuario no provee todos los parámetros del gráfico, estos son asumidos a juicio del modelo u omitidos según su relevancia.

Las claves que conforman el archivo JSON son:

- Color.
- Título.
- Nombre eje x.

- Nombre eje y.
- Tipo de gráfico.
- Falta de datos.
- Explicación.

Como se puede ver en las claves, existe un parámetro de 'Falta de datos', el cual clasifica si el gráfico puede ser generado directamente o si es necesario obtener más información de la base de datos para poder generarlo.

6.5.2. Experto SQL-Gráfico

El Experto SQL-Gráfico tiene la misma funcionalidad que el experto SQL original, la cual es generar una consulta SQL en base a la pregunta del usuario. Este modelo es solo accedido cuando el usuario solicita un gráfico con información que no se encuentra en la data obtenida en la misma conversación.

La diferencia entre este modelo y el Experto SQL del flujo principal es que este considera toda la conversación y, pese a recibir la petición de generar un gráfico, solo considera la información que puede ser traspasada a consulta SQL.

6.5.3. Generador gráfico

Esta es una función que toma el archivo JSON generado por el reconocedor gráfico para construir el gráfico que solicita el usuario. En este se considera el tipo de gráfico, como: de barras, de líneas, de dispersión o histograma; el valor de los ejes; el título; y el color de las barras, puntos o líneas.

6.5.4. Limitaciones actuales del flujo

Actualmente, el modelo tiene problemas de memoria referente a recibir peticiones de manera fraccionada, por lo que si se pide información en diferentes mensajes del

usuario, no siempre se obtiene el gráfico con todas las indicaciones.

6.6. Diagrama de flujo

A modo de resumen, la lógica seguida por las instancias de modelos de lenguaje se muestra en la figura [6.1](#), esta corresponde al diagrama de flujo del módulo y muestra el orden a seguir durante la comunicación entre modelos.

6.7. Comparativa de modelos de lenguaje

6.7.1. Diferencias en desempeño

En el cuadro 6.1 se presenta una comparativa entre los tres modelos de la familia Claude disponibles para el equipo.

Tarea	Claude 3.5	Claude 3.0	Claude 3 Haiku
Identificador General	Sin fallos	Falla en archivos	Falla en archivos y consulta
Experto SQL	Sin fallos	Con fallo en obtención de tablas	Con fallo en parámetros a obtener
Identificador SQL/NL	Sin fallos	Sin fallos	Sin fallos
Corrector SQL	Sin fallos	Reconoce lo que falta, falla en comprender que la pregunta no puede ser respondida	Cae en el mismo error de incluir información que no existe en las tablas
Traductor data a NL	Sin fallos	Sin fallos	Sin fallos
Reconocedor Gráfico	Sin fallos	Sin fallos	Sin fallos
Experto SQL gráfico	Sin fallos	Sin fallos	Fallo en entendimiento de tablas

Cuadro 6.1: Comparativa modelos familia Claude

6.7.2. Diferencias de costos

Promediando los casos de uso utilizados durante el plan de pruebas de la plataforma, se tiene que se utilizan alrededor de 4500 tokens de entrada y 200 tokens de salida en una conversación que contempla preguntas y solicitudes para generar archivos.

El cobro de los distintos modelos está dado en dólares por cada millón de tokens, por lo que se puede tener una cantidad estimada cercana a 222 conversaciones antes de llegar al millón de tokens de entrada. Esta cantidad de conversaciones resultaría en, aproximadamente, 44.000 tokens de salida.

El cuadro 6.2 muestra la diferencia de precios para cantidad de conversaciones en base a la cantidad de tokens promedio de la plataforma.

Modelo	Costo por millón de tokens de entrada	Costo por millón de tokens de salida	Costo entrada + salida cada 100 conversaciones	Costo entrada + salida cada 222 conversaciones
Claude 3.5	\$3	\$15	\$1.65	\$3.66
Claude 3	\$3	\$15	\$1.65	\$3.66
Claude 3 Haiku	\$0.25	\$1.25	\$0.13	\$0.31

Cuadro 6.2: Costos por modelo

6.7.3. Diferencias generales

Si bien Claude 3.5 tiene un desempeño mejor que Claude 3.0 y Claude 3 Haiku, este trae consigo un precio igual o mayor, tanto de entrada como de salida, respecto a ambos modelos. Por esto, es de utilidad saber en cuáles tareas los modelos menos avanzados pueden reemplazar a Claude 3.5, con el objetivo de reducir el gasto asociado a la operación del sistema.

A modo de resumen, las diferencias generales entre los distintos modelos, en cuanto a respuestas, son:

- Claude 3 Haiku se especializa en tareas más simples y que requieran de velocidad, es útil en escenarios donde los recursos son limitados.
- Claude 3, en comparación con modelos como Claude 2, es mucho más hábil para responder de manera más precisa a tareas complejas, pero es más lento comparado con los otros dos modelos.
- Claude 3.5 es más rápido que Claude 3 y tiene mejor desempeño en interacciones y contextos más prolongados, lo que hace que sea sumamente efectivo para este proyecto y se mantenga como solución definitiva para las partes del flujo que requieren tareas complejas.

6.8. Comparativa de modelos de incorporación

En las siguientes secciones se presentan los resultados obtenidos a partir de los diferentes modelos de embeddings probados por el equipo, estos son: titan-embed-text-v1 y titan-embed-text-v2 de Amazon, además de uno de los mejores modelos multilingües disponibles en HuggingFace, multilingual-e5-large, y uno que está al borde del top 100, all-mpnet-base-v2, el cual se usa como caso base.

6.8.1. Diferencias en desempeño

En cuanto al desempeño de los modelos, en los cuadros 6.3 y 6.4 se muestra el desempeño de los diferentes modelos en cuanto al número de falsos negativos (porcentaje de documentos que debieron ser retornados, pero no lo fueron) y el número de falsos positivos (porcentaje de documentos que no debieron ser retornados, pero lo fueron), respectivamente.

Cabe destacar que las columnas con nombre “Tasa de falsos X*” corresponden a los valores alcanzados una vez se ajusta el radio de búsqueda para lograr que los falsos negativos se vuelvan 0.

Modelo	Tasa falsos negativos	Tasa falsos negativos*
titan-embed-text-v1	14.29 %	0 %
titan-embed-text-v2	21.43 %	0 %
all-mpnet-base-v2	0 %	-
multilingual-e5-large	0 %	-

Cuadro 6.3: Tasa de falsos negativos por modelo

Modelo	Tasa falsos positivos	Tasa falsos positivos*
titan-embed-text-v1	47.83 %	69.57 %
titan-embed-text-v2	31.25 %	43.75 %
all-mpnet-base-v2	60 %	-
multilingual-e5-large	58.82 %	-

Cuadro 6.4: Tasa de falsos positivos por modelo

Por otro lado, en el cuadro 6.5 se muestran los tiempos de ejecución total por modelo.

Modelo	Promedio tiempos de ejecución	Desviación estándar tiempos de ejecución
titan-embed-text-v1	4.2511	0.8869
titan-embed-text-v2	4.6428	1.6849
all-mpnet-base-v2	4.8374	1.1615
multilingual-e5-large	4.8665	1.1793

Cuadro 6.5: Tiempos de ejecución por modelo

6.8.2. Diferencias de costos

A continuación, se describen los precios por modelo cada 1000 tokens transformados a embeddings, esto se muestra en el cuadro 6.6.

Modelo	Costo por 1000 tokens
Titan Text Embeddings	0.0001 USD
Titan Text Embeddings V2	0.00002 USD
multilingual-e5-large	Gratuito
all-mpnet-base-v2	Gratuito

Cuadro 6.6: Costos cada 1000 tokens por modelo

6.8.3. Diferencias generales

Con la información entregada, se ve que ambas versiones de Titan Text Embeddings de Amazon presentan un buen desempeño y, junto a multilingual-e5-large, destacan en aspectos diferentes.

En el caso de que los tiempos de ejecución sean la mayor prioridad en el desarrollo, se recomienda el uso de Titan Text Embeddings v1. Mientras que, si se tiene conocimiento de valores esperados en cuanto a distancias, la segunda versión del modelo de Amazon es una mejor alternativa. Esto se debe a que este retorna documentos con más precisión, evitando introducir ruido al modelo, lo cual se traduce en un contexto más reducido y respuestas más precisas. Por otro lado, si el presupuesto para la generación de embeddings es una restricción, se recomienda el uso de multilingual-e5-large, el cual tiene buenos resultados de manera gratuita.

Para el desarrollo, se toma el modelo titan-embed-text-v2, ya que este permite obtener resultados más ceñidos a lo pedido por el usuario, a cambio de un aumento marginal en los costos.

Cualquiera sea el caso que se escoja, los embeddings generados se deben almacenar con el fin de no transformar más tokens de lo necesario y malgastar recursos tanto computacionales como económicos. Existen diversas alternativas para esta tarea, siendo el almacenamiento de vectores, o vector stores, el método más común. Esto es un trabajo futuro de interés para el proyecto.

Capítulo 7

Versión actual del prototipo

En este capítulo se detalla la última versión de la plataforma obtenida del proceso de desarrollo. En particular, se describen sus funcionalidades, características y vistas.

Como recordatorio, este corresponde a un sistema de obtención de información, el cual evita utilizar horas hombre en traducir consultas SQL manualmente, entregando una alternativa de comunicación directa entre usuario y sistema.

7.1. Esquema general de componentes del prototipo

A continuación, se describen los distintos componentes de las vistas de la aplicación, ordenados por las diferentes vistas que la conforman.

7.1.1. Elementos generales

- Barra de navegación.
- Botones.

7.1.2. Vista principal

- Barra lateral de conversaciones antiguas.

- Input para chat.
- Módulo para mensajes de chat.

7.1.3. Vista conversaciones

- Tabla con conversaciones.
- Paginación.
- Filtros.

7.1.4. Vista métricas

- Tabla con métricas.
- Paginación.
- Filtros.

7.1.5. Vista administrador

- Sección para gestión de permisos.
- Tabla con usuarios.
- Tabla con roles.
- Paginación.

7.1.6. Vista inicio de sesión

- Cuadro de ingreso de credenciales.
- Botones.

7.1.7. Vista crear cuenta

- Formulario de creación.

7.2. Descripción de componentes

7.2.1. Barra de navegación

- **General:** Permite el cambio entre vistas, así como cerrar sesión y dar la bienvenida al usuario.
- **Interacción:** El usuario puede hacer clic en distintos botones que lo dirigirán a las funcionalidades.
- **Funciones:** Permitir el cambio de vistas de manera sencilla.

7.2.2. Botones

- **General:** Permite la interacción del usuario con algunas funciones del sistema.
- **Interacción:** El usuario hace clic en el componente.
- **Funciones:** Cada botón tiene una función en particular, como iniciar sesión, cerrar sesión, enviar mensaje, cambiar de vista, abrir nueva conversación, etc.

7.2.3. Barra lateral

- **General:** Permite ver y acceder a conversaciones anteriores de un mismo usuario con el modelo.
- **Interacción:** El usuario hace clic en un botón para que se muestre el componente, de esto se puede hacer clic en las conversaciones anteriores del usuario.
- **Funciones:** Tener acceso a las conversaciones antiguas del usuario.

7.2.4. Input para chat

- **General:** Es un formulario que actúa como input para chat. Permite al usuario escribir mensajes.
- **Interacción:** El usuario escribe un mensaje luego de hacer clic en el input.
- **Funciones:** Permite al usuario escribir un mensaje y enviarlo al chatbot.

7.2.5. Módulo para mensajes de chat

- **General:** Permite visualizar los mensajes anteriores en la conversación.
- **Interacción:** El usuario puede ver los mensajes que se han tenido durante la conversación con el sistema.
- **Funciones:** Mostrar los mensajes entre el chatbot y el usuario de una conversación específica.

7.2.6. Tabla de conversaciones

- **General:** Una tabla de conversaciones que muestra los atributos más importantes de cada conversación, como el primer mensaje, la consulta generada, entre otros datos.
- **Interacción:** El usuario puede ver los datos y filtrarlos.
- **Funciones:** Ver los datos más importantes de las conversaciones.

7.2.7. Paginación

- **General:** Permite navegar entre una gran cantidad de datos.
- **Interacción:** El usuario puede hacer clic en distintos botones.
- **Funciones:** Permite cambiar la página actual de los datos.

7.2.8. Filtros

- **General:** Permite el filtrado información.
- **Interacción:** El usuario indica un filtro y genera una llamada distinta desde el backend a la API.
- **Funciones:** Filtrar información según el usuario lo indique.

7.2.9. Tabla de Métricas

- **General:** Una tabla de métricas que muestra las métricas de cada conversación, como la satisfacción del usuario, comentarios, entre otros datos.
- **Interacción:** El usuario puede ver los datos y filtrarlos.
- **Funciones:** Ver los datos de las métricas.

7.2.10. Sección para gestión de permisos

- **General:** Vista que permite al usuario administrador gestionar permisos.
- **Interacción:** El usuario puede seleccionar usuarios o roles y administrarlos al hacer clic en distintas opciones y botones.
- **Funciones:** Permite cambiar los permisos de los usuarios y roles.

7.2.11. Cuadro de ingreso de credenciales

- **General:** Es un formulario que recibe los datos para el inicio de sesión en la plataforma.
- **Interacción:** El usuario rellena el formulario con sus credenciales para acceder a la plataforma.

- **Funciones:** Permitir al usuario ingresar sus credenciales para acceder a la plataforma.

7.2.12. Formulario de creación

- **General:** Permite crear un nuevo usuario en el sistema.
- **Interacción:** El usuario rellena el formulario y aprieta el botón para crear el usuario.
- **Funciones:** Crear nuevos usuarios para acceder a la plataforma.

7.3. Entornos de desarrollo

7.3.1. Entorno de desarrollo local

Para el entorno de desarrollo local, se cuenta con un IDE y la versión 3.9 de Python para ejecutar el código. Cada repositorio se puede iniciar en su contenedor Docker respectivo o en un entorno virtual.

En este entorno, se deben instalar los requerimientos de cada repositorio desde el archivo requirements.txt. Una vez completada la instalación, se debe ejecutar el archivo entrypoint.py para iniciar la API y la plataforma. Una vez iniciada, la interfaz de la plataforma es accesible desde un navegador web, ingresando al puerto local donde se haya desplegado.

7.3.2. Entorno de producción

En el entorno de producción es necesario configurar un servidor adecuado con Python 3.9 instalado. Cada repositorio debe desplegarse dentro de su contenedor Docker correspondiente, o bien en un entorno de producción virtualizado.

Es crucial asegurar la instalación de todas las bibliotecas detalladas en los archivos de requerimientos de cada repositorio. Luego, se debe ejecutar el archivo `entrypoint.py` para iniciar la API y la plataforma.

Una vez se hace esto, la interfaz de la plataforma estará disponible a través del navegador web, accediendo a la dirección IP del servidor y al puerto configurado para el despliegue.

7.4. Implementación de módulos y funcionalidades

7.4.1. Base de datos

La base de datos utilizada en el sistema utiliza la versión 16.3 de PostgreSQL, un gestor para base de datos relacionales el cual utiliza el equipo de trabajo para almacenar información de lo necesario para las diferentes pruebas de las funcionalidades del sistema (Tablas con información accedida por los flujos), como información referente a la data que se maneja dentro de la aplicación, tales como conversaciones, métricas, entre otras.

7.4.2. API

En el cuadro [7.1](#) se muestran los diferentes endpoints de la plataforma, el tipo de solicitud HTTP y cuerpo que se les envía y también la o las respuestas con las que puede responder.

Endpoint	Tipo de solicitud	Respuestas	Cuerpo
/ping/	GET	[200]	
/chat/sendMessage/	POST	[200-422]	prompt: string, conversation_id: int, user_id: int
/chat/getConversations/	GET	[200-422]	user_id: int
/chat/getConversationMessages/	GET	[200-422]	conversation_id: int
/chat/changeConversationName/	POST	[200-422]	conversation_id: int name: string
/chat/getConversationTable/	GET	[200-422]	limit: int, offset: int, order_by: string, order_way: string
/files/download/	GET	[200-422]	file_id: int
/files/check/	GET	[200-422]	file_id: int
/metrics/send/	POST	[200-422]	conversation_id: int, calification: int, questions: dict
/metrics/getTable/	GET	[200-422]	limit: int, offset: int, order_by: string, order_way: string

Cuadro 7.1: Endpoints API

7.4.3. Autenticación

Para la autenticación de la API, se utiliza el protocolo OAuth 2.0, que es un estándar ampliamente usado para la autorización segura. En este caso, dado que la API será accedida mediante un esquema Machine-to-Machine (M2M), se implementa Auth0 como proveedor de autenticación. Auth0 actúa como un servidor de autorización, encargado

de recibir las solicitudes de tokens y emitir los tokens necesarios para acceder a la API.

Por otro lado, se configura un cliente dentro de Auth0, el cual estará asociado a un `client_id` y un `client_secret`. Estos datos son credenciales únicas que se pasarán al backend como variables de entorno para mantener la seguridad. El backend utilizará estas credenciales para solicitar un token de acceso a Auth0, el cual será utilizado para hacer llamadas a la API.

Cuando la API recibe una solicitud acompañada de un token, valida su autenticidad y permisos antes de permitir el acceso a la información solicitada. De esta manera, se asegura que sólo las máquinas autorizadas puedan acceder a la API de forma segura y controlada.

7.4.4. Notificaciones

El sistema de notificaciones utiliza como base la API de Google, con la cual vía correo se notifican tanto los cambios de contraseñas como de permisos en la plataforma a los usuarios respectivos.

Casos de uso:

- Restablecer contraseña: Le llega a todos los usuarios, independiente de su rol, que necesitan cambiar su contraseña. Un ejemplo de este mensaje se muestra en la figura [7.1](#).

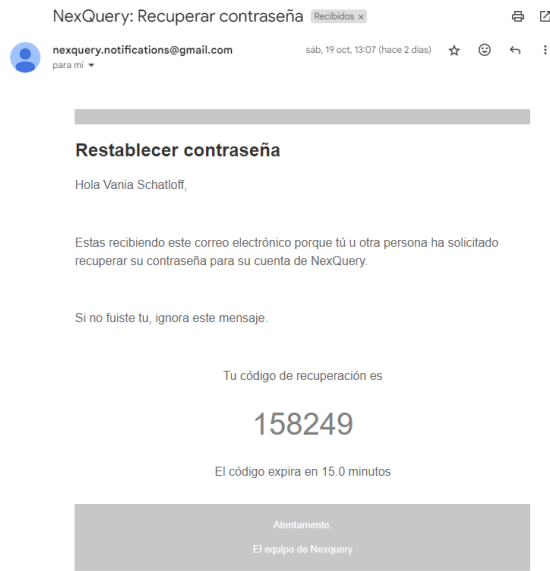


Figura 7.1: Correo para restablecer contraseña

- Nuevo usuario: Le llega a los usuarios administradores. Un ejemplo de este mensaje se muestra en la figura 7.2.

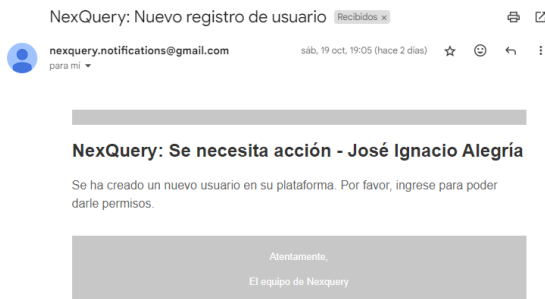


Figura 7.2: Correo para aviso de creación de usuario

- Cambio de rol: Le llega a los usuarios cuyo rol fue actualizado. Un ejemplo de este mensaje se muestra en la figura 7.3.

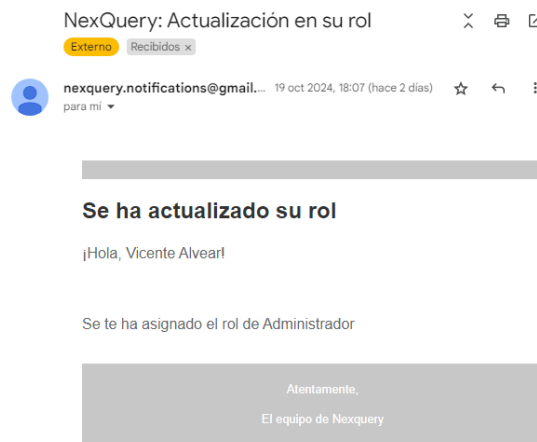


Figura 7.3: Correo para aviso de actualización de rol

7.4.5. Almacenamiento de archivos

Para el almacenamiento de archivos, estos se manejan de forma local dentro del proyecto, permaneciendo ahí hasta que son procesados por diferentes tareas programadas mediante la biblioteca Cron. Estas se encargan de recorrer los archivos almacenados localmente y para eliminar todos los archivos que tengan un tiempo de vida determinado. Vale decir, una vez que los archivos alcanzan un tiempo específico desde su creación, la tarea Cron los identifica y elimina automáticamente.

7.4.6. Interfaz gráfica implementada

La interfaz gráfica comprende una aplicación con múltiples funcionalidades a través de varias vistas en las que se puede navegar e interactuar de manera intuitiva y amigable.

Contando con:

- Un inicio de sesión en el cual se encuentra implementado todo el sistema de crear una cuenta como el de cambiar contraseña.

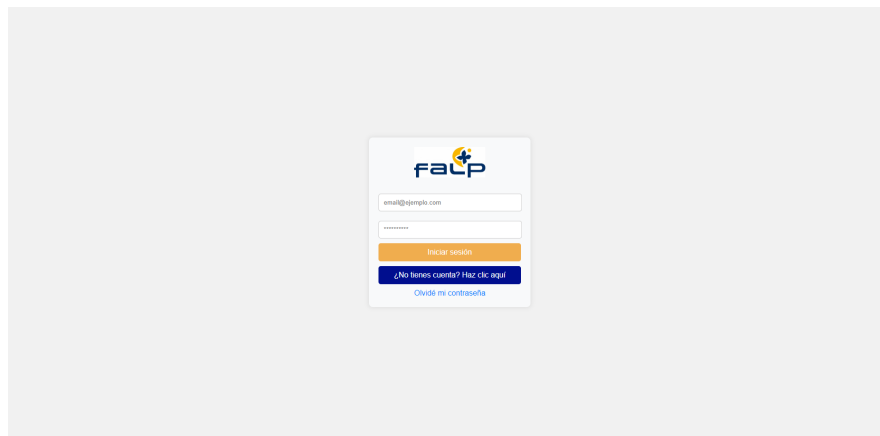


Figura 7.4: Vista de inicio de sesión

A registration form for the 'falp' system. It features the 'falp' logo at the top. Below the logo are two input fields for 'Nombre' (containing 'Pablo') and 'Apellido' (containing 'Torres'). This is followed by an 'Ingrese su email:' label and an email input field with 'email@ejemplo.com'. Then, there is an 'Ingrese su contraseña:' label and a password input field with '*****'. Below that is another 'Ingrese su contraseña nuevamente:' label and a second password input field with '*****'. At the bottom, there are two buttons: an orange one labeled 'Me quiero registrar' and a blue one labeled '¿Tienes cuenta? Haz clic aquí'.

Figura 7.5: Vista de registro

- Vistas para el cambio de contraseña, comenzando con la solicitud, la verificación de código y finalizando con el cambio de contraseña.



falcp

Recuperar contraseña

Correo electrónico:

email@ejemplo.com

Quiero recuperar mi contraseña

¿Quieres iniciar sesión? Haz clic aquí

Figura 7.6: Vista de solicitud de nueva contraseña



falcp

Recuperar contraseña

Ingrese código de recuperación:

Enviar código

¿Quieres iniciar sesión? Haz clic aquí

Figura 7.7: Vista de ingreso de código



falcp

Recuperar contraseña

Ingrese nueva contraseña:

.....

Ingrese contraseña nuevamente:

.....

Cambiar contraseña

Figura 7.8: Vista de cambio de contraseña

- Una vista de bienvenida dinámica en base a los permisos que el usuario posea.

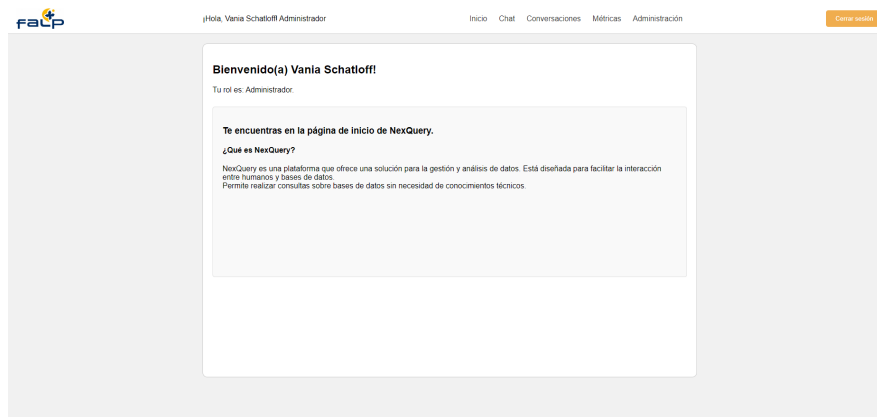


Figura 7.9: Vista de inicio para usuario con permiso asignado



Figura 7.10: Vista de inicio para usuario sin permiso asignado

- Una vista de chat donde se puede tener conversaciones con el chatbot especializado en obtener respuestas de la base de datos en NL. Esta vista cuenta además con una barra lateral desplegable donde se puede ver el historial de conversaciones de la cuenta que se este utilizando, estas conversaciones también cuentan con un sistema de valoración para así obtener un feedback directo de los usuarios y saber en que tipo de preguntas puede estar fallando el sistema.

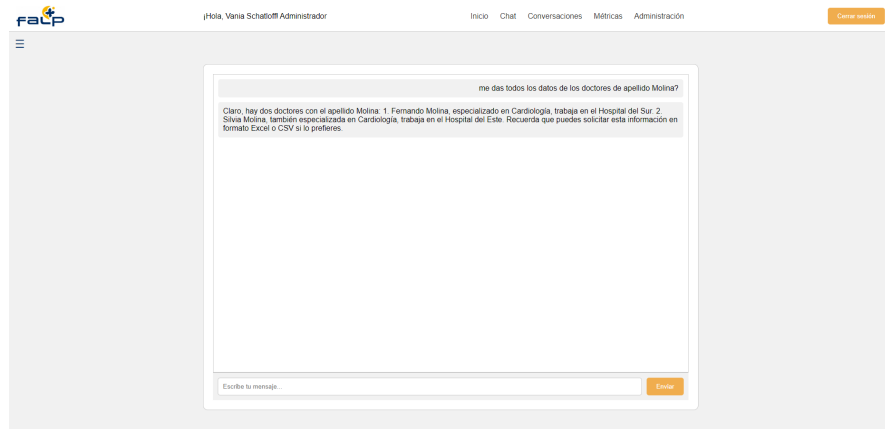


Figura 7.11: Vista principal de chatbot

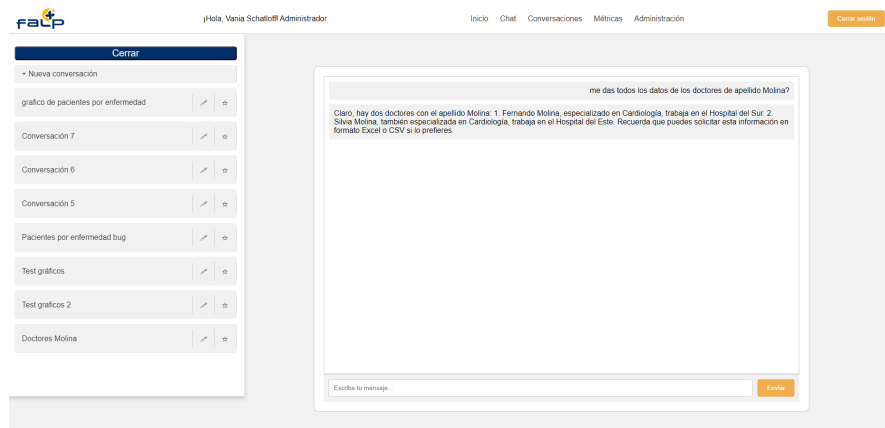


Figura 7.12: Vista de barra lateral con conversaciones

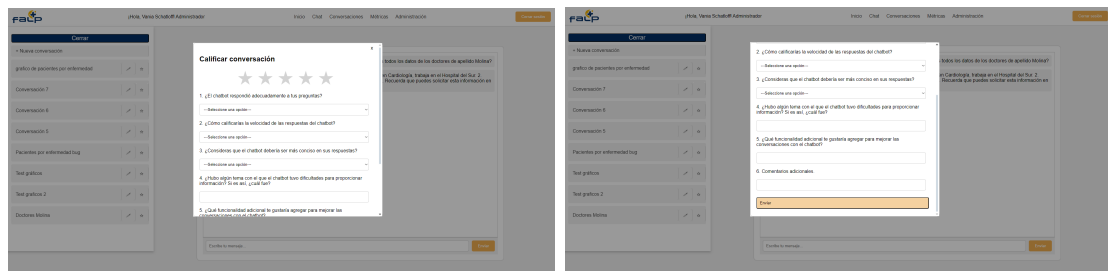


Figura 7.13: Vista para calificar conversación

- Una vista de conversaciones donde se pueden ver todas las conversaciones que han tenido los usuarios. Esta cuenta con un sistema de filtrado para facilitar su uso y generar un uso mas amigable para los usuarios con acceso a esta vista.

Id conversación	Id usuario	Mensaje inicial	Consulta generada
8	1	puedes hacer el conteo de pacientes con cancer de mama por hospital?	<pre>SELECT h.hospital, COUNT(p.id) AS cantidad_pacientes FROM public.pacientes p JOIN public.atenciones a ON p.id =a.id_paciente JOIN public.doctores h ON a.id_dotor = h.id WHERE p.tipo_cancer = 'Cáncer de Mama'; GROUP BY h.hospital;</pre>
8	1	puedes darme la información sobre los pacientes que tienen cancer de mama? Necesito saber el hospital y en realidad quiero toda la informacion	<pre>SELECT p.id, p.nombre, p.sur, p.tipo_cancer, p.categoria_cancer, d.hospital FROM public.pacientes p JOIN public.atenciones a ON p.id =a.id_paciente JOIN public.doctores d ON a.id_dotor = d.id WHERE p.tipo_cancer = 'Cáncer de Mama'; SELECT DISTINCT p.* FROM public.pacientes p</pre>

Figura 7.14: Vista de tabla de conversaciones

- Una vista de métricas donde se pueden observar las respuestas de los usuarios y la configuración básica del chatbot con la que este esta trabajando para responder esa conversación en particular, esta vista también cuenta con un sistema de filtrado dentro de la pestaña de "ver métricas" y "ver preguntas" para acceder de manera rápida a la información que al administrador le interese.

Id métrica	Id conversación	Id usuario	Calificación (valor máximo 5)	Preguntas	Métricas
6	6	1	1	Ver preguntas	Ver métricas
5	7	1	1	Ver preguntas	Ver métricas
4	8	1	1	Ver preguntas	Ver métricas
3	7	1	3	Ver preguntas	Ver métricas
2	8	1	3	Ver preguntas	Ver métricas
1	8	1	5	Ver preguntas	Ver métricas

Figura 7.15: Vista de tabla de métricas

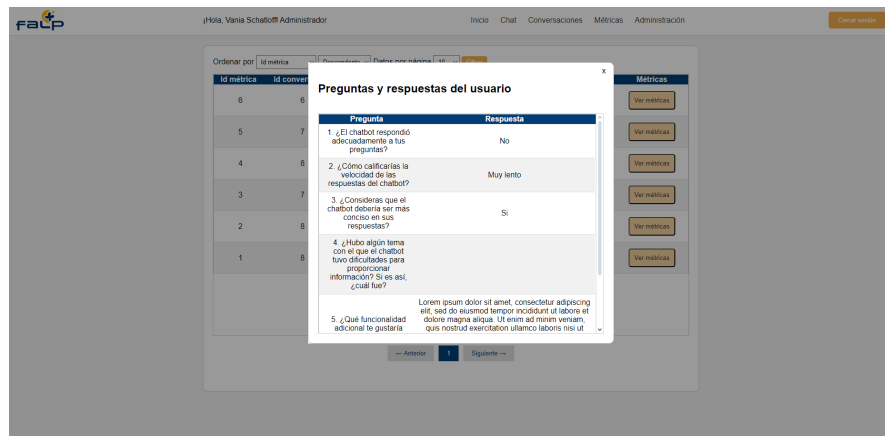


Figura 7.16: Vista de modal con preguntas respondidas del usuario

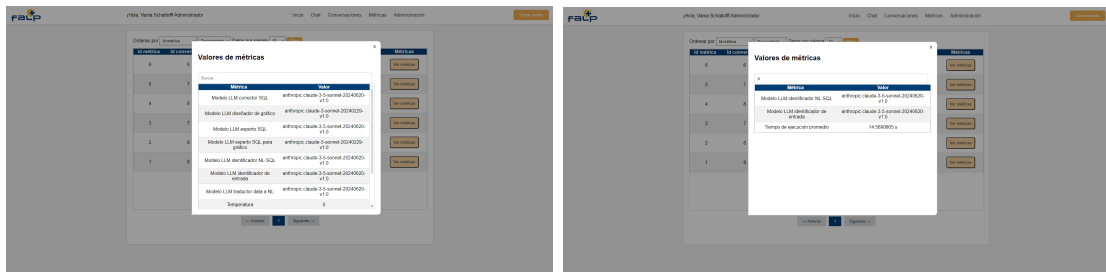


Figura 7.17: Vista de modal con métricas de la conversación y ejemplo de uso del buscador

- Una vista de administrador donde se pueden crear y manejar los permisos de los distintos usuarios de la plataforma para otorgarles los distintos tipos de acceso a la misma.
 - Usuarios: Permite administrar los roles de cada usuario.

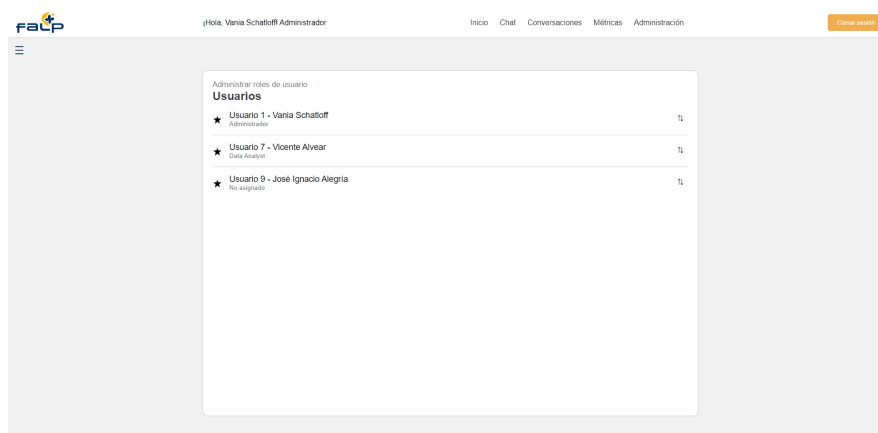


Figura 7.18: Vista de administración de usuarios

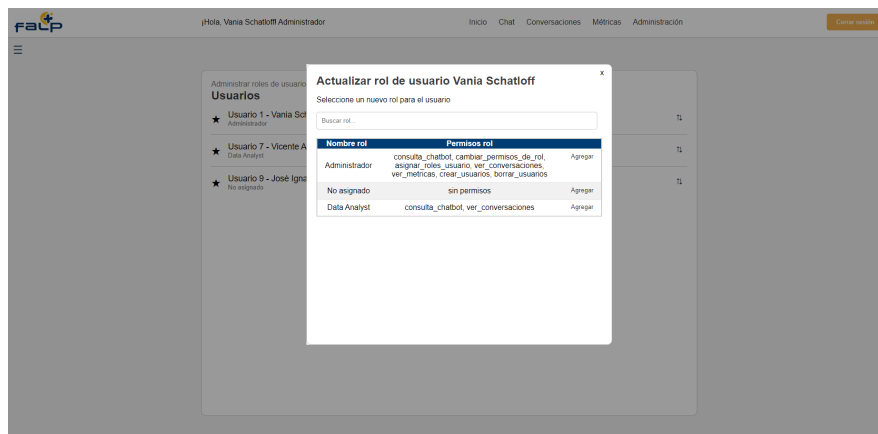


Figura 7.19: Vista de modal de actualización de rol de usuario

- Roles: Permite la creación y actualización de roles.

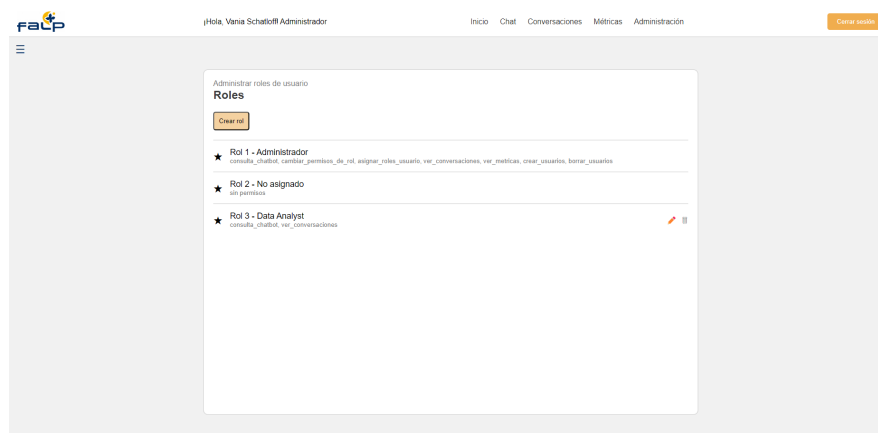


Figura 7.20: Vista de administración de roles

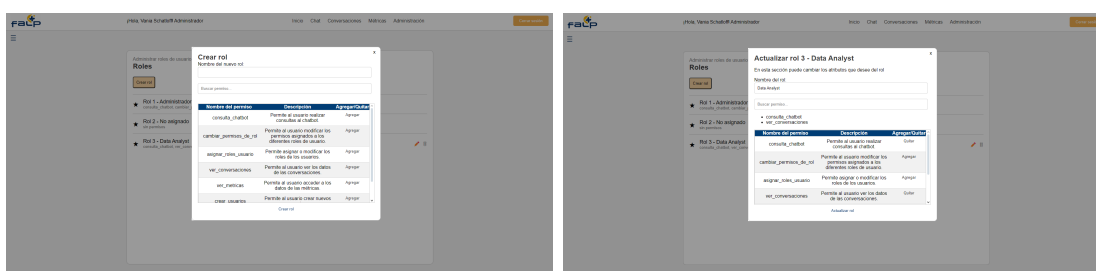


Figura 7.21: Vista de modal de creación y actualización de roles

- **Permisos:** Permite ver los permisos existentes para la plataforma.

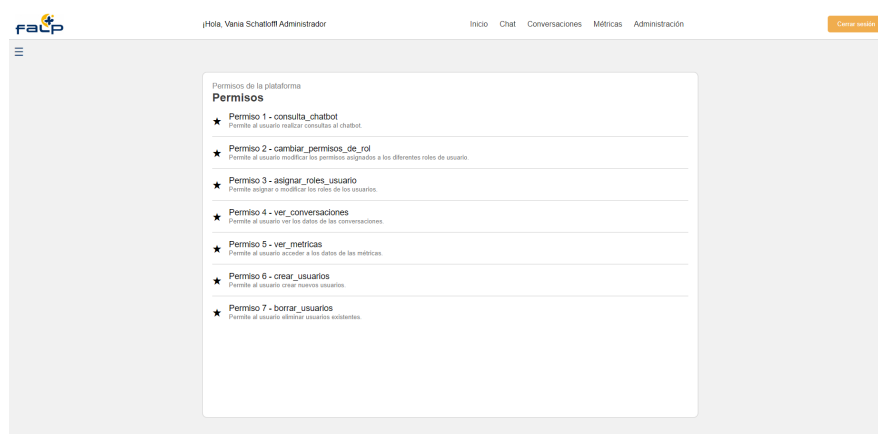


Figura 7.22: Vista de permisos

- Todas las vistas cuentan con una barra de navegación global en la sección superior de la vista para acceder a la distintas vistas previamente mencionadas, así

como también un botón para cerrar sesión. Esto permite una navegación mas rápida y sencilla para el usuario.

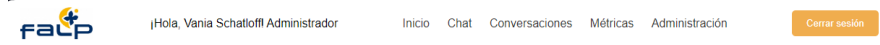


Figura 7.23: Barra de navegación

Capítulo 8

Conclusiones

A modo de cierre, el desarrollo realizado culminó en una plataforma amigable y clara, la cual permite a sus usuarios obtener información de las bases de datos que requieran mediante comunicación “directa” con el sistema. Una consecuencia directa de esto es la solución de los principales dolores a enfrentar por el proyecto: Liberar horas hombre de personal técnico, previamente utilizadas en obtener información necesaria para las tareas de su organización y, como respuesta, dar acceso a información a personal no especializado.

Esto demuestra que el uso de modelos de lenguaje ya existentes permite anular la brecha existente en el problema de la comunicación humano –datos. Ya que, al utilizar un acercamiento divide y conquistarás junto a encadenamiento de modelos, se obtiene un sistema de sencilla depuración, adaptable a innumerables entornos, y que permite democratizar el acceso a la información de manera efectiva.

Anexos

8.1. Gestión de riesgos

8.1.1. Supuestos

1. Se supone disponibilidad de los distintos modelos de lenguaje.
2. Se supone la existencia de diferentes métricas relacionadas al desempeño de los modelos.
3. Se supone que las métricas referentes al desempeño de los modelos son comparables.
4. Se supone la capacidad de creación de nuevos modelos en la base de datos del cliente para el almacenamiento de las diferentes métricas de las interacciones usuario - chatbot.

8.1.2. Dependencias

1. Se depende de la entrega del documento de confidencialidad para obtener acceso a los datos del cliente.
2. Se depende de la entrega de datos por parte de la fundación.
3. Se depende del acceso a servicios como JupyterHub y Amazon Bedrock con sus respectivos modelos fundacionales.

4. Se depende de la comunicación con el cliente para cualquier eventualidad que se considere pertinente.

8.1.3. Restricciones

1. Las bases de datos del cliente utilizan PostgreSQL y MySQL como gestores y SQL como lenguaje, por lo que se trabajará dentro de un paradigma relacional.

8.1.4. Riesgos

1. Riesgos asociados al desconocimiento tecnológico propios de un proyecto que involucra nuevas tecnologías.
2. Riesgos asociados a no obtener información por parte del cliente.

En el cuadro 8.1 se muestran estos riesgos junto a las medidas de mitigación acordadas por el equipo de desarrollo.

Riesgo	Medidas de mitigación
Desconocimiento tecnológico.	Estudio e investigación por parte del equipo.
No obtención de la información del cliente.	Crear y poblar bases de datos sintéticas que se asemejen a las utilizadas por el cliente.

Cuadro 8.1: Riesgos y respectivas medidas de mitigación

8.2. Verificación y validación

8.2.1. Verificación

Actualmente, las pruebas se han limitado a pruebas simples de comportamiento de los distintos modelos en su contexto y pruebas del flujo principal. En particular, el

equipo ha hecho pruebas unitarias, de integración y de extremo a extremo para verificar el correcto funcionamiento de los modelos y el sistema respectivamente.

8.2.2. Validación

Si bien el proceso de validación se ha realizado por medio de distintas reuniones, se planea que después de mostrar el prototipo inicial al cliente, estas puedan ser mas frecuentes con el fin de recopilar mas retroalimentación para añadir funcionalidades, mejoras y ajustes en la aplicación. Mediante este proceso, se busca generar conocimiento sobre las potenciales dificultades que el cliente o los usuarios finales tengan con el prototipo.

La retroalimentación recibida ha estado enfocada en añadir funcionalidades al flujo de modelos con el fin de aumentar la calidad de uso del sistema.

8.2.3. Plan de pruebas

Como plan de pruebas, se busca ir más allá de las pruebas realizadas por el equipo, para esto se pretenden hacer pruebas mas contundentes con participación del cliente y personas ajenas al desarrollo, entre los que idealmente se encuentran los usuarios finales de la plataforma.

Bibliografía

- [1] Claude E. Shannon and Warren Weaver. *The mathematical theory of information*. University of Illinois Press, Champaign, Illinois, 1949.
- [2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research* 3, February 2003.
- [3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv:1706.03762*, June 2017.
- [4] Abhilasha Kate, Satish Kamble, Aishwarya Bodkhe, and Mrunal Joshi. Conversion of natural language query to sql query. In *Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, March 2018.
- [5] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv:2308.153*, November 2023.
- [6] Zijin Hong, Zheng Yuan, Hao Chen, Qinggang Zhang, Feiran Huang, and Xiao Huang. Knowledge-to-sql: Enhancing sql generation with data expert llm. *arXiv:2402.11517v2*, March 2024.

- [7] Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Linzheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. Mac-sql: A multi-agent collaborative framework for text-to-sql. *arXiv:2405.07467v1*, February 2024.
- [8] Puneet Kumar Ojha, Abhishek Gautam, Ankit Agrahari, and Parikshit Singh. Sft for improved text-to-sql translation. *International Journal of Intelligent Systems and Applications in Engineering*, February 2024.
- [9] Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation. *arXiv:2312.11242v3*, May 2024.
- [10] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, January 2018.
- [11] Yingpeng Du, Di Luo, Rui Yan, Xiaopei Wang, Hongzhi Liu, Hengshu Zhu, Yang Song, and Zhang. Enhancing job recommendation through llm-based generative adversarial networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(8):8363–8371, March 2024.
- [12] Zhenwen Li and Tao Xie. Using llm to select the right sql query from candidates. *arXiv:2401.02115v1*, January 2024.