Aplicación Web De Atención Para Consultas De Procedimientos Administrativos Y Preguntas Frecuentes Con Integración De Un Chatbot

Juan Felipe Arango Guzmán

Universidad del Valle
Facultad de ingeniería
Escuela de ingeniería de sistemas y computación
Tuluá
2025

Aplicación Web De Atención Para Consultas De Procedimientos Administrativos Y Preguntas Frecuentes Con Integración De Un Chatbot

Juan Felipe Arango Guzmán
Código 2060066
juan.felipe.arango@correounivalle.edu.co

Director
Joshua David Triana Madrid, MSc
joshua.triana@correounivalle.edu.co

Universidad del Valle
Facultad de ingeniería
Escuela de ingeniería de sistemas y computación
Tuluá
2025

Agradezco a Dios por permitirme realizar este sueño que anhelaba desde niño y ahora se vuelve realidad.
Agradezco a mi familia por siempre darme la moral para continuar.
Agradezco a mis amigos y compañeros que formaron parte de nuestro grupo con quienes forjamos grandes lazos y sin ellos esta bonita experiencia no hubiera sido igual.
Agradezco a todos lo profesores, quienes brindaron de forma paciente y con gran disposición algo tan valioso como es el conocimiento.
Agradezco a mi director, el profesor Joshua Triana, por su guía y respaldo en el desarrollo de este proyecto.
Pero en especial debo brindar mis más fervientes agradecimientos a dos grandes personas, primero mi madre, quien lucho junto conmigo durante todos estos años, surcando todos los obstáculos que se presentaron a lo largo del camino y merece tanto como yo este reconocimiento. Y, por último, agradezco al profesor Mauricio López Benítez quien también me acompaño durante casi todo este trayecto, cumpliendo a la perfección los roles de profesor, jefe y amigo, convirtiéndose para mi en un modelo a seguir.

Índice

Índice de cuadros

Índice de figuras

Resumen

En este proyecto de trabajo de grado, se desarrolló una aplicación web con información para atender preguntas frecuentes y consultas acerca de algunos procesos administrativos que puedan tener los estudiantes de ingeniería de sistemas de la sede Tuluá, con el apoyo de un asistente virtual (chatbot) para la resolución de dudas acerca de la información requerida, con el fin de reducir el tiempo que usa el personal administrativo en la atención de estos asuntos, incrementando la productividad de los mismos y a su vez dar una respuesta oportuna a dudas o preguntas frecuentes de los alumnos evitando que se desplacen a la oficina o en contra parte, que envíen correos que debido al gran volumen de estos, la respuesta tardará algo de tiempo en llegar. En el desarrollo del proyecto se utilizó inteligencia artificial (IA) para la creación del chatbot incorporando técnicas de procesamiento de lenguaje natural (PLN) para que los estudiantes tengan una interacción fluida y facilidad para dar respuesta a sus cuestionamientos. Además, se adoptó un enfoque web, considerando que actualmente las personas utilizan cada vez más dispositivos inteligentes. Desarrollar una aplicación nativa para cada plataforma puede ser una tarea ardua. Este problema puede ser mitigado significativamente mediante el uso de tecnologías web, ya que permiten que la aplicación sea accesible desde una amplia gama de dispositivos con el único requisito de contar con un navegador web compatible.

Palabras clave: Chatbot, Aplicación web, PLN, IA, LLM, RAG.

1. Introducción

A nivel mundial, las empresas gastan muchos recursos en la atención a los clientes, pues estos son el pilar que conforman las compañías, por ende, es importante procurar que los usuarios tengan una buena experiencia y también optimizar el proceso de atención para ahorrar los costos tanto de dinero como de tiempo. En este contexto, la implementación de chatbots se ha vuelto cada vez más común [1], ya que ofrecen la oportunidad de automatizar la interacción con los clientes y dar respuestas a sus consultas de manera rápida y precisa. La adopción de los asistentes virtuales en distintos sectores ha demostrado ser beneficioso tanto para las entidades que los implementan como para los usuarios, pues entre un 40 % y 80 % de las consultas comunes que en otro caso habrían sido atendidas por personal humano [1]. Por un lado, las empresas pueden reducir sus costos al automatizar procesos tardados como lo es la atención de personas, teniendo una disponibilidad continua. Por otro lado, los usuarios se ven beneficiados con respuestas instantáneas sin esperar una cola, lidiar con tiempos prolongados en teléfono o tener que desplazarse largas distancias. En el ámbito educativo, la implementación de estos asistentes tiene un gran potencial de cambiar la manera en que se da asistencia a los estudiantes. Al brindar respuesta rápida resolviendo consultas y preguntas comunes, los chatbots pueden mejorar de manera significativa la experiencia de los estudiantes y liberar al personal administrativo para enfocarse en tareas más complejas o vitales[2]. La implementación de un asistente de propósito específico da la oportunidad de tener una comunicación asertiva entre los estudiantes y la información que les pueda ser brindada por el chatbot. Este enfoque permite asegurar que las consultas realizadas por los estudiantes sean atendidas de manera lógica y coherente, dando respuestas contextualizadas dentro del ámbito específico en el que se enfoca. El uso de técnicas de PLN permite al chatbot interpretar el significado contextual de las consultas como el significado de siglas o el nombre de una persona en determinado cargo como el asistente de la Universidad de Granada “Elvira” capaz de reconocer las siglas UGR (Universidad de Granada) y dar el nombre del rector en ese momento [3]. En Colombia en los últimos años se ha incrementado el número de estudiantes que se matriculan en instituciones de educación superior, a través de la herramienta SNIES (Sistema Nacional de Información de Educación Superior) en el 2022 se registraron 2.466.228 matriculados de los cuales 2.284.637 son de pregrado [4]. En las instituciones académicas es común que el número de estudiantes aumente, pero el personal administrativo de mantenga constante, lo que puede generar un cuello de botella en la la atención de consultas presenciales o correos electrónicos, a su vez limitando el tiempo y productividad del personal administrativo, de tal forma que la inclusión de un asistente para este tipo de dudas produciría un ahorro de tiempo y esfuerzo humano [2].

2. Formulación del problema

2.1. Descripción del problema

La Universidad del Valle es una institución de renombre con una gran cantidad de estudiantes matriculados en todas las sedes que la componen, se encuentra constantemente con la necesidad de brindar un servicio administrativo ágil y efectivo que satisfaga las demandas de una comunidad estudiantil diversa y en crecimiento. Con más de 33000 estudiantes matriculados en el año 2022 en la sede principal [5] y aproximadamente 12500 estudiantes en las sedes regionales en 2021 [6], la magnitud de la población de estudiantes resalta la importancia de contar con formas de atención eficientes para los estudiantes. En particular, la sede Tuluá de la Universidad del Valle, con 1653 estudiantes matriculados en el año 2020 [7], de los cuales aproximadamente 400 estudiantes forman parte del programa de Ingeniería de Sistemas, la oficina de la coordinación enfrenta desafíos específicos en la gestión de dudas y consultas. Estas consultas, muchas veces repetitivas o redundantes, no solo consumen tiempo y recursos valiosos del personal administrativo, sino que también pueden afectar la calidad de la atención brindada a los estudiantes que presentan problemas urgentes. El volumen de consultas puede aumentar significativamente durante períodos clave del año académico, como el inicio de semestre, los períodos de matrícula, adiciones y cancelaciones. Este aumento en la demanda de atención administrativa puede sobrecargar aún más los recursos disponibles y generar demoras en la respuesta a consultas legítimas. De esta manera, se evidencia la necesidad de implementar soluciones tecnológicas innovadoras que permitan optimizar la atención administrativa y mejorar la experiencia de los estudiantes. La automatización de procesos mediante herramientas tecnológicas puede contribuir significativamente a reducir los tiempos de espera y agilizar la resolución de consultas, liberando recursos y aumentando la eficiencia operativa de la oficina de coordinación de Ingeniería de Sistemas.

2.2. Definición del problema

Como ayuda para el desarrollo de la definición del problema se realizó la práctica de crear un árbol de problemas que se puede evidenciar en la *Figura 1*, esta práctica ayuda a denotar de manera visual las causas y efectos que existen alrededor de la problemática que se desea afrontar. De este modo se llega a la siguiente pregunta problematizadora:

¿Cómo implementar una solución informática que contribuya a reducir las interrupciones en el flujo de trabajo del personal administrativo?

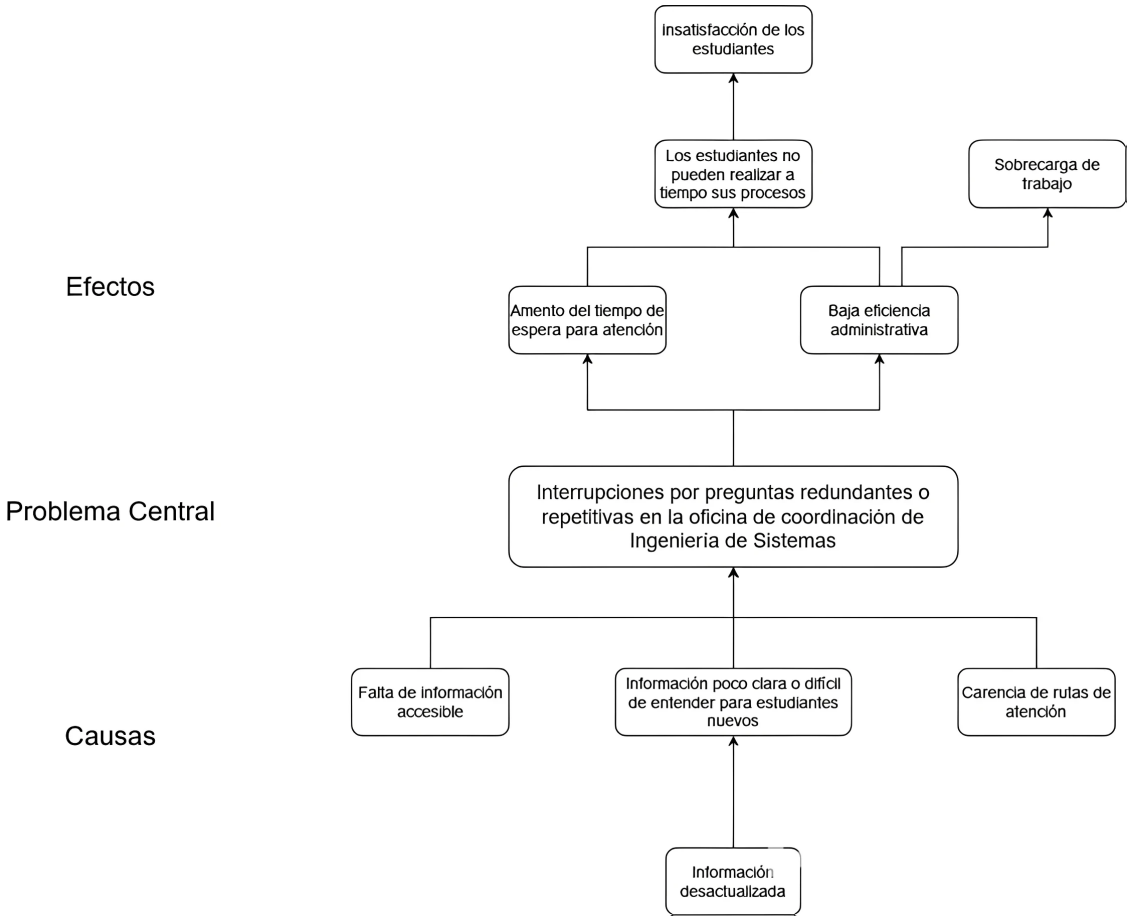

Figura 1: Árbol de problemas
Fuente: Elaboración propia

3. Marco de referencia

3.1. Marco teórico

Teoría General de Sistemas: La Teoría General de Sistemas no es solo una simple teoría, es un enfoque metodológico diseñado para abordar el estudio de un sistema en su totalidad, considerando todas las partes que lo componen e investigando las relaciones e interacciones entre ellas. Su objetivo es alcanzar una comprensión global y generalizada del sistema mediante el uso de estrategias científicas. La Teoría General de Sistemas permite la elaboración de modelos y pronósticos del comportamiento de un sistema antes de implementarlo, a través de procesos de simulación. Todo esto facilita elegir la mejor alternativa para dar solución a una problemática [8].

Lean Manufacturing: La metodología Lean Manufacturing ha sido adoptada por empresas en busca de mejorar la competitividad en el mercado mejorando sus resultados usando menos recursos. Su objetivo principal es eliminar todas las actividades que no agregan valor en el proceso productivo. Aunque surgió para la industria automotriz en Japón, sus principios y técnicas han sido aplicadas en diversos sectores de servicios y manufactura. La metodología Lean tiene aspectos clave como la sobreproducción, las esperas, el inventario, el transporte, los defectos, el desperdicio de procesos, los movimientos innecesarios y la sub-utilización de la capacidad de los empleados. Sin embargo, otro aspecto fundamental de esta metodología es su filosofía empresarial, que valora la comprensión de las personas y los factores que las motivan [9].

Inteligencia Artificial (IA): La inteligencia artificial (IA) se define como un campo interdisciplinario que se centra en el estudio y desarrollo de sistemas inteligentes capaces de realizar tareas que normalmente requerirían la inteligencia humana. La IA abarca una amplia gama de enfoques, como algoritmos de búsqueda simples o modelos de aprendizaje automático. En el campo de la inteligencia artificial se busca replicar la forma humana de aprender, adaptarse y tomar decisiones utilizando técnicas computacionales [10].

PLN (Procesamiento de Lenguaje Natural): El procesamiento de lenguaje natural (PLN) es una disciplina de la informática que forma parte de la inteligencia artificial, enfocada en que las computadoras puedan comprender y procesar el lenguaje humano. El PLN combina técnicas de lingüística computacional, que se basan en reglas para modelar el lenguaje humano, con modelos estadísticos, de machine learning y de deep learning. Con estas técnicas las computadoras puedan comprender el significado completo del lenguaje humano, incluyendo la intención y el sentimiento del hablante o escritor. El PLN impulsa gran variedad de aplicaciones, como traductores o sistemas de reconocimiento de comandos de voz [11].

Retrieval-Augmented Generation (RAG): La generación aumentada por recuperación (RAG) es una técnica que permite enriquecer un modelo de lenguaje con una base de conocimiento específica, distinta de los datos utilizados en su entrenamiento. Los mo-

delos de lenguaje grandes (LLM) se entrenan con enormes volúmenes de datos generales, lo que puede llevarlos a generar información imprecisa o inventada cuando se les consulta sobre datos específicos a los que no tuvieron acceso. Además, reentrenar un modelo para incorporar nuevo conocimiento requiere un tiempo considerable y una gran capacidad de hardware, esta técnica aborda estas limitaciones al integrar información externa de manera eficiente, mejorando la precisión y relevancia de las respuestas [12].

3.2. Marco conceptual

Chatbot: Un chatbot es un programa informático que emplea inteligencia artificial y procesamiento del lenguaje natural para entender y responder a las preguntas de los usuarios, imitando la interacción humana. Inicialmente, se basaban en texto y ofrecían respuestas predefinidas para consultas simples, pero con el tiempo han evolucionado hacia sistemas más sofisticados que pueden mantener conversaciones complejas. Los chatbots más modernos pueden reconocer el contexto y aprender de las interacciones con los usuarios [13].

Productividad laboral: La productividad laboral se define como la relación entre los bienes o servicios producidos por un trabajador y los recursos utilizados para obtener esa producción. Es una medida de eficiencia que puede aplicarse tanto a una empresa específica como a toda una economía. Calcular la productividad laboral es esencial para las empresas, ya que les proporciona información sobre su desempeño y les permite comprender mejor su funcionamiento, identificar causas de baja productividad y evaluar el impacto de sus políticas y decisiones en la productividad. La productividad laboral aumenta cuando se incrementa el valor agregado mediante la mejora en el uso de los factores de producción [14].

Aplicación web: : Una aplicación web es una aplicación cliente-servidor que generalmente usa el navegador web como cliente. Los navegadores envían peticiones a los servidores, que generan y devuelven respuestas. A diferencia de las antiguas aplicaciones cliente-servidor, las aplicaciones web usan un programa cliente común: el navegador web. Los navegadores se encuentran disponibles en prácticamente la totalidad de los ordenadores, estos evitan instalar múltiples programas de cliente especializados [15].

LLM (Large Language Model): Los modelos de lenguaje de gran escala son redes neuronales basadas en la arquitectura de transformadores, compuestas por un codificador y un decodificador. Estos procesan los datos de entrada y generan vectores multidimensionales conocidos como embeddings, que permiten al modelo identificar patrones y relaciones entre los datos al medir la proximidad entre dichos vectores. Los LLM son altamente versátiles y pueden entrenarse para diversas tareas, como la generación o completación de texto, la creación de código, imágenes, audio y más. Durante el entrenamiento, los datos se dividen en unidades más pequeñas llamadas tokens, el modelo predice el siguiente token basándose en los tokens previos, lo que le permite aprender y generar secuencias coherentes [16].

Embedding: Los embeddings o incrustaciones son representaciones de datos, como texto, imágenes o audio, en forma de puntos en un espacio vectorial, la posición de estos puntos refleja las relaciones y patrones entre los datos, estas representaciones se generan mediante redes neuronales que producen vectores multidimensionales, capturando en sus dimensiones conexiones y características que serían imperceptibles para un humano, por ejemplo, en el procesamiento de texto, los embeddings permiten a los modelos identificar relaciones entre palabras, comprender el contexto y generar respuestas coherentes y relevantes[17].

3.3. Antecedentes

Introducing Martha: Leveraging Cognitive Automation to Create an Exceptional Student Experience: En la publicación se habla sobre la implementación exitosa de un chatbot llamado Martha en la Universidad George Washington (GW). Este chatbot se dio como respuesta a la necesidad de simplificar el acceso a los sistemas y servicios educativos para los estudiantes. Con una comunidad estudiantil diversa compuesta por 26,000 personas de 50 estados y 130 países, se enfrentaba el desafío de mantenerse al día con la demanda creciente, para abordar esta situación, GW IT adoptó la tecnología de chatbot como estrategia de servicio. Martha fue desarrollada utilizando BMC Helix Platform y BMC Helix Chatbot, se convirtió en un recurso valioso para los estudiantes dando asistencia las 24 horas del día. Durante el piloto, Martha atendió 4,581 interacciones, destacando su eficacia y preferencia entre los estudiantes. Este éxito se atribuye a la capacidad de Martha para resolver una variedad de solicitudes, desde ayudar a los estudiantes a conectarse a la red inalámbrica hasta registrar dispositivos y responder preguntas frecuentes. La implementación de Martha no solo mejoró la experiencia del estudiante, sino que también tuvo un impacto positivo en la eficiencia operativa de GW IT. Al reducir la carga de trabajo del personal del servicio de asistencia, Martha permitió que estos se enfocaran en resolver problemas más complejos y de mayor prioridad [18].

Asistentes virtuales en plataformas 3.0: En la Universidad de Granada (UGR), se implementó un asistente virtual llamado Elvira, diseñado para simplificar el acceso a los servicios y sistemas educativos para los estudiantes. Se desarrolló utilizando técnicas de inteligencia artificial (IA) y procesamiento del lenguaje natural (PLN), Elvira se despliega en la página web de la UGR, permitiendo a los usuarios formular consultas en lenguaje natural y recibir respuestas relevantes. Elvira fue integrada en la página web de la UGR y ofrece respuestas precisas y contextuales a las consultas de los estudiantes sobre una variedad de temas relacionados con la universidad. Su implementación ha sido un éxito notable, con una alta aceptación por parte de los estudiantes. Durante el periodo piloto, Elvira gestionó miles de consultas, evidenciando su eficacia como herramienta de apoyo para los estudiantes. Además de su presencia en la página web, Elvira se implementó en dispositivos móviles y en la red social Facebook. Su capacidad para ofrecer soporte las 24 horas del día, sin necesidad de aumentar el personal de atención al cliente, ha sido especialmente valorada [3].

Asistente virtual académico utilizando tecnologías cognitivas de procesamien-

to de lenguaje natural: El Politécnico Colombiano Jaime Isaza Cadavid implementó un asistente virtual académico utilizando tecnologías cognitivas tipo chatbot para mejorar la experiencia y tiempos de atención en procesos académicos. Este chatbot proporciona a los estudiantes información instantánea, optimizando el tiempo y reduciendo el esfuerzo humano a un bajo costo. El Politécnico cuenta con diversas plataformas de atención, como el sistema PQRS y redes sociales, que presentan problemas comunes en instituciones universitarias, como largos tiempos de respuesta debido a problemas de comunicación y saturación de los puntos de atención. Para solucionar esta problemática, se propuso un chatbot diseñado para simular conversaciones, entender lo que los usuarios escriben y responder acertadamente. Su funcionamiento fue bien recibido dado que, mediante una encuesta con 10 estudiantes de diferentes carreras, obtuvo más del 80 % de aprobación [2].

RefAI: a GPT-powered retrieval-augmented generative tool for biomedical literature recommendation and summarization: Para abordar las dificultades que enfrentan los profesionales biomédicos al buscar y sintetizar la gran cantidad de literatura científica, se desarrolló RefAI. Esta herramienta potenciada por GPT busca superar las limitaciones de los modelos de lenguaje actuales, que a menudo inventan referencias o se equivocan citando fuentes específicas. RefAI integra PubMed para la recuperación de artículos, emplea un novedoso algoritmo multivariable para la recomendación de literatura pertinente y utiliza GPT-4 turbo, dando una recomendación inicial de fuentes que son verificadas para comprobar su veracidad, aumentar las palabras clave para la recuperación de la información y también generar resúmenes coherentes. El sistema está diseñado para proporcionar recomendaciones de alta calidad y resúmenes fiables, asegurando la autenticidad de las fuentes y la precisión de los metadatos, a diferencia de otros sistemas que pueden incluir artículos falsos o información incorrecta. La efectividad de RefAI fue evaluada por 10 expertos del dominio, comparándola con herramientas como ChatGPT-4, ScholarAI y Gemini, los resultados indicaron que RefAI superó significativamente a las alternativas en puntos clave como la relevancia y calidad de la recomendación, así como la precisión, exhaustividad y correcta integración de referencias en los resúmenes [19].

Desarrollo de un chatbot que ayude a responder a preguntas frecuentes referentes a becas en la Universidad Técnica Particular de Loja: En la publicación se detalla el desarrollo de un chatbot para la Universidad Técnica Particular de Loja (UTPL), diseñado como una solución innovadora para simplificar y agilizar el acceso a la información sobre becas para sus estudiantes. El proceso para obtener detalles sobre postulaciones, requisitos y tipos de becas a menudo resultaba complejo, generando un alto volumen de consultas repetitivas al personal administrativo, para modernizar y hacer más eficiente este servicio, la UTPL pensó en implementar un asistente virtual como estrategia de comunicación. Este chatbot fue desarrollado utilizando la plataforma de inteligencia artificial IBM Watson, convirtiéndose en un recurso de gran valor para los estudiantes al brindar asistencia precisa las 24 horas del día. El objetivo principal del proyecto fue agrupar toda la información referente a las becas y responder de manera automática a las preguntas más frecuentes de los postulantes y beneficiarios. El desarrollo de este asistente virtual mejora exponencialmente la experiencia del estudiante, garantizando respuestas

inmediatas y accesibles en cualquier momento, además de contribuir a la eficiencia operativa de la universidad, al reducir la carga de trabajo del personal administrativo derivada de consultas recurrentes, el chatbot permite que los equipos de la UTPL puedan concentrarse en resolver casos más específicos y de mayor complejidad, mejorando la calidad de la atención [20].

4. Alcance del proyecto

4.1. Declaración del alcance

El presente proyecto de trabajo de grado tiene la intención de dar una respuesta innovadora mediante la tecnología para agilizar la atención de los estudiantes brindando atención de manera simple e inmediata evitando así tiempos de espera y desplazamientos a la oficina de Ingeniería de Sistemas de la Universidad del Valle Sede Tuluá. Permitiendo que los administrativos dispongan de ese tiempo para actividades importantes, disminuyendo las interrupciones de un flujo de trabajo y aumentando la productividad.

Actualmente los medios de atención e información son limitados, centrándose en los miembros de la oficina que son: profesores tiempo completo, auxiliares y monitores, teniendo en cuenta que estos últimos están solo unas pocas horas al día en la oficina. Los miembros de la oficina pueden atender a los alumnos presencialmente o por el correo electrónico del programa; En fechas como matrícula académica, los medios pueden sufrir colapsos dada la gran cantidad de correos o personas que se acercan a hacer consultas.

Para abordar esta problemática, se propuso el desarrollo de un asistente virtual utilizando tecnologías cognitivas. Este asistente virtual, diseñado como un chatbot, es capaz de entender y responder a las consultas de los estudiantes de manera automática y eficiente, mejorando los tiempos de respuesta y la satisfacción de los usuarios, al mismo tiempo que se ahorra esfuerzo humano. El asistente virtual permite a los estudiantes solucionar sus dudas sobre procesos institucionales sin necesidad de desplazarse, facilitando la visualización y transmisión de información de manera inmediata.

El desarrollo del proyecto tuvo una etapa donde se recolectó la información necesaria para el funcionamiento del chatbot, tanto la información que brinda la oficina sobre las preguntas frecuentes de fácil respuesta que interrumpen de manera más continua su flujo operativo, como también la literatura necesaria para construir un modelo de inteligencia artificial que se adapte a las necesidades del proyecto. Lo anterior abrió paso a la integración de todo en la aplicación web que será presentada al usuario, el cual tendrá la tarea de hacer pruebas de usabilidad con el fin de evaluar el funcionamiento de la herramienta.

Es mandatorio recalcar que para llevar a cabo la correcta realización del proyecto se contó con determinadas condiciones que se presentan a continuación:

4.1.1. Supuestos:

- Continuidad laboral del director del trabajo de grado.
- Normalidad académica durante el desarrollo del proyecto.
- Acceso a la información y datos requeridos para la implementación del proyecto.
- No ocurrencia de eventos imprevistos o emergencias que puedan interrumpir el desarrollo del proyecto.

4.1.2. Restricciones:

- Se contó con un tiempo de 8 meses calendario (2 periodos académicos) para realizar el proyecto.
- Los equipos utilizados en el proyecto fueron los que estaban a disposición del estudiante, y todos los costos, a excepción del salario del director, fueron asumidos por el estudiante.

4.2. Objetivos

4.2.1. Objetivo general

Desarrollar una aplicación web con un chatbot integrado que contribuya a mejorar el flujo de trabajo del personal administrativo en la oficina de Ingeniería de Sistemas de la Universidad del Valle Sede Tuluá reduciendo las interrupciones.

4.2.2. Objetivos específicos

1. Recopilar información de procesos administrativos, preguntas frecuentes y datos que solicitan los estudiantes que interrumpen el flujo de trabajo del personal administrativo.
2. Investigar el estado del arte para la selección de la técnica de PLN adecuada para el proyecto.
3. Implementar el modelo de PLN seleccionado adaptándolo para la aplicación.
4. Evaluar la aplicación mediante pruebas de software y de usabilidad del usuario.

4.2.3. Resultados obtenidos

Los objetivos específicos guían el desarrollo de un proyecto, al culminar cada uno de estos se concreta uno o varios entregables que soportan su cumplimiento, como se puede apreciar en la *Tabla 1* donde se relacionan cada uno de los objetivos específicos con sus respectivos entregables.

Tabla 1: Resultados obtenidos

No.	Objetivo específico	Entregable
1	Recopilar información de procesos administrativos, preguntas frecuentes y datos que solicitan los estudiantes que interrumpen el flujo de trabajo del personal administrativo.	▪ Documento con la información de procesos y preguntas frecuentes que pueden ser resueltas por el asistente.
2	Investigar el estado del arte para la selección de la técnica de PLN adecuada para el proyecto.	▪ Documento con la recolección de literatura y justificación de la selección de técnica de PLN.
3	Implementar el modelo de PLN seleccionado adaptándolo para la aplicación.	▪ Código fuente del modelo de PLN construido. ▪ Documento con explicación del modelo.
4	Desarrollar la aplicación web con la información recopilada y el chatbot de colaboración al usuario.	▪ Documento con la investigación y justificación de las tecnologías web usadas. ▪ Mockups de la interfaz de la aplicación web. ▪ Código fuente de la aplicación web con la información dispuesta para el usuario y el chatbot integrado. ▪ Artefactos como diagrama de base de datos y casos de uso.
5	Evaluar la aplicación mediante pruebas de software y de usabilidad del usuario.	▪ Documento con los resultados de las pruebas. ▪ Documento de análisis de los resultados.

Fuente: Elaboración propia

5. Metodologías

5.1. Metodología de investigación

Los Niveles de Madurez Tecnológica (TRL, por sus siglas en inglés) son una herramienta útil para definir el alcance de las actividades de Investigación, Desarrollo Tecnológico e Innovación (I+D+i) en proyectos y programas. Originados en la NASA en los años 70, los TRL han sido adaptados para diversos usos, incluyendo su aplicación por COLCIENCIAS en la organización de actividades del Sistema Nacional de Ciencia, Tecnología e Innovación de Colombia. Los TRL permiten identificar la correspondencia de las actividades de I+D+i con las diferentes etapas del desarrollo tecnológico. Este esquema facilita la interpretación del grado de madurez tecnológica de los resultados esperados en propuestas de diversas áreas [21]. Usando como guía el TRL se evaluará el nivel de madurez de la aplicación web que será usada por los estudiantes de Ingeniería de Sistemas de la Sede Tuluá, generando una buena gestión de las actividades de I+D+i. Se espera que, al finalizar el proyecto, la aplicación se encuentre entre los niveles de TRL-5 y TRL-6 que se pueden apreciar en la Figura 2, siendo capaz de ser ejecutada en un entorno cercano al real sin ningún problema.

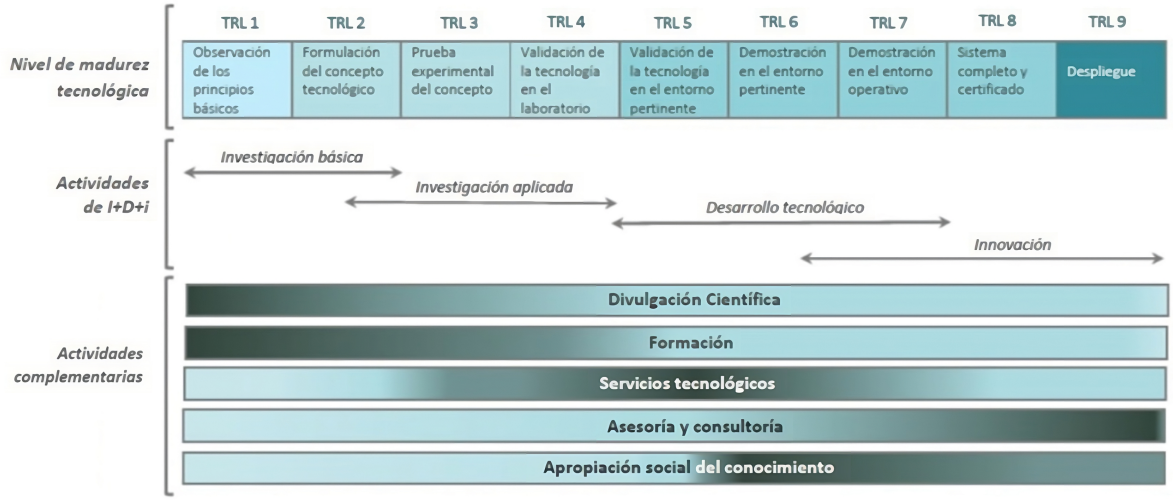

Figura 2: Niveles de maduración tecnológica
Fuente: Tomado de [21]

5.2. Metodología de desarrollo de software

Las metodologías ágiles han ganado prominencia en la comunidad de ingeniería del software, generando un intenso debate con respecto a las metodologías tradicionales. Surgidas del Manifiesto Ágil en febrero de 2001, estas metodologías fueron concebidas por 17 expertos de la industria del software que se reunieron en Utah, EE. UU. Su objetivo era establecer valores y principios que permitieran a los equipos desarrollar software de manera rápida y adaptativa frente a los cambios. Las metodologías tradicionales, llamadas de

otra forma como “metodologías pesadas”, se caracterizan por ser rígidas y centradas en la documentación exhaustiva en cada fase del desarrollo, a comparación de las metodologías ágiles que se diseñaron teniendo en mente plazos cortos, requisitos cambiantes usando nuevas tecnologías y haciendo énfasis en la flexibilidad y la colaboración continua Las metodologías ágiles ofrecen varias ventajas, como la capacidad de responder rápidamente a los cambios en los requisitos, lo que es muy importante en entornos de desarrollo donde los requerimientos pueden llegar a cambiar. Fomentan la colaboración constante entre los miembros del equipo y los clientes, lo que asegura que el producto final se apegue a las necesidades del usuario. Además, al centrarse en la entrega continua de software funcional, permiten una retroalimentación temprana y frecuente, mejorando la calidad del producto y aumentando la satisfacción del cliente [22]. Con motivo de elegir una metodología apropiada para el desarrollo del proyecto se hizo una comparativa de 3 metodologías ágiles que son: Scrumban, XP (Extreme Programming) y FDD (Feature-Driven Development), cabe aclarar que Scrum y Kanban son marcos de trabajo, pero en la industria del desarrollo se han utilizado de tal forma que es indistinto de una metodología, por lo que será evaluado como tal, la comparativa se presenta en la Tabla 2.

Tabla 2: Calificación de metodologías de desarrollo de software

Criterio	**Scrumban**	**XP**	**FDD**
Presencia en internet	5	5	2
Documentación	4	5	4
Certificación	3	4	2
Training	5	3	2
Comunidades	5	3	2
Presencia empresarial	4	4	3
Proyectos de software	4	4	2
Posibilidad de trabajo individual	4	2	3
Total	**34**	**32**	**20**

Fuente: Elaboración propia

Un criterio clave para calificar las metodologías es la posibilidad de realizar trabajo individual, ya que el proyecto fue llevado a cabo por una sola persona con el acompañamiento limitado del director. Scrumban, al combinar las mejores características de Scrum y Kanban, permitió trabajar sin roles definidos, lo que facilita el trabajo individual. Se llevaron a cabo sprints de una semana de duración, y al final de cada uno se realizó una reunión de retrospectiva con el director para presentar los avances logrados y discutir las dificultades encontradas durante el sprint. La flexibilidad de Scrumban permite que las historias de usuario no requieran estimación previa y se abordaran según se fue considerando su prioridad en el transcurso del proyecto.

5.3. Metodología de gestión de actividades

Al haberse elegido Scrumban como metodología de desarrollo de software, se usó el enfoque de Kanban para la gestión de actividades. Usando Kanban se hace gestión del trabajo en curso (WIP) que asegura un trabajo continuo y evita la sobrecarga de trabajo en el equipo. Este sistema produce solo la cantidad que el equipo puede manejar. Evitando excesos y sobrecargas. El Kanban es un sistema "just in time", lo que significa que evita la acumulación innecesaria de tareas y la inversión de tiempo en actividades menos prioritarias. Limitar el trabajo en progreso es clave, ya que gestionar demasiadas tareas a la vez disminuye la calidad del trabajo debido a la reducción de la capacidad de concentración de los desarrolladores. La aplicación habitual de Kanban se realiza mediante la gestión visual del flujo de trabajo con paneles que reflejan el estado del equipo en cada momento. Un tablero Kanban se divide en columnas que representan las diferentes etapas del proceso, como "En curso" y "Lista" como se muestra en la Figura 3. Cada tarea se representa con una tarjeta que se mueve a través de estas columnas según avanza en el proceso [23].

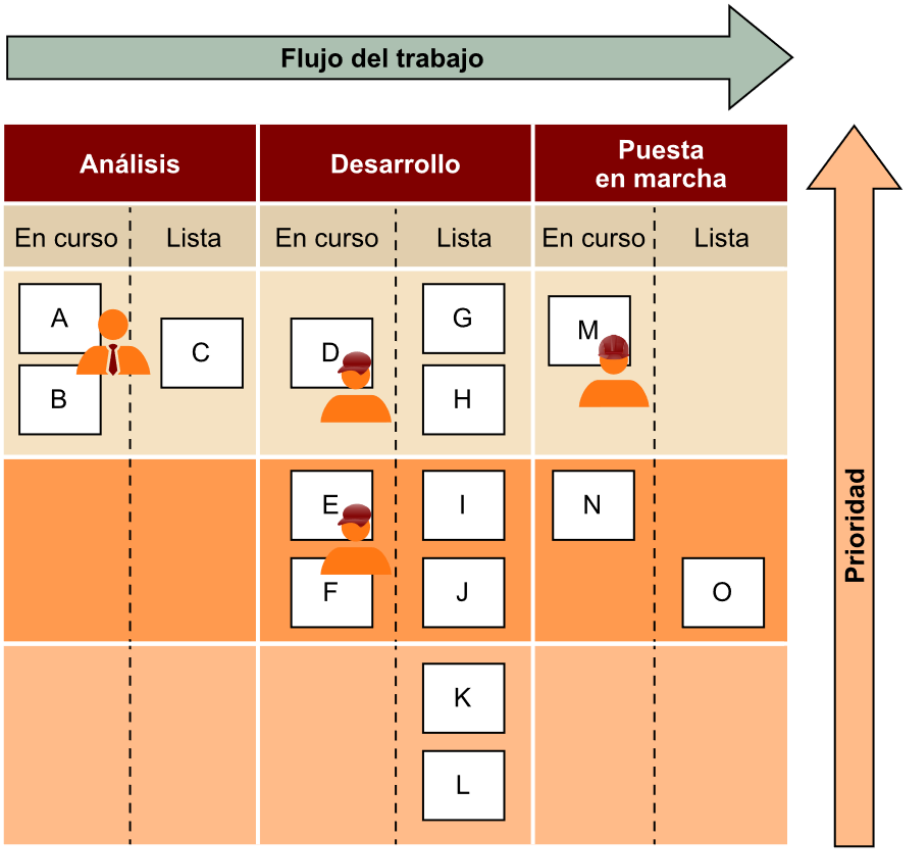

Figura 3: Tablero Kanban
Fuente: Tomado de [23]

6. Aplicación de Scrumban

Para el desarrollo de la aplicación, inicialmente se consideró la herramienta ClickUp por recomendación del director del trabajo de grado. Sin embargo, al explorar esta plataforma de gestión de proyectos, se identificó que, aunque ofrece una amplia variedad de funcionalidades, esta fortaleza se convierte en una desventaja para usuarios nuevos debido a su pronunciada curva de aprendizaje, lo que dificulta su uso. Por esto, durante la etapa de planificación del desarrollo de software, se decidió migrar a la plataforma Jira, la cual había sido utilizada anteriormente con éxito en otros proyectos. Jira es una herramienta sencilla pero robusta que permite gestionar proyectos de manera eficiente. Su interfaz intuitiva y su compatibilidad con metodologías ágiles facilitan la planificación de sprints, la creación y asignación de tareas, el movimiento de estas entre sprints y secciones del tablero Kanban, así como la visualización de fechas en el calendario para el seguimiento de actividades, entre otras funcionalidades. En la *Figura 4* y *Figura 5* se muestra la interfaz de la plataforma.

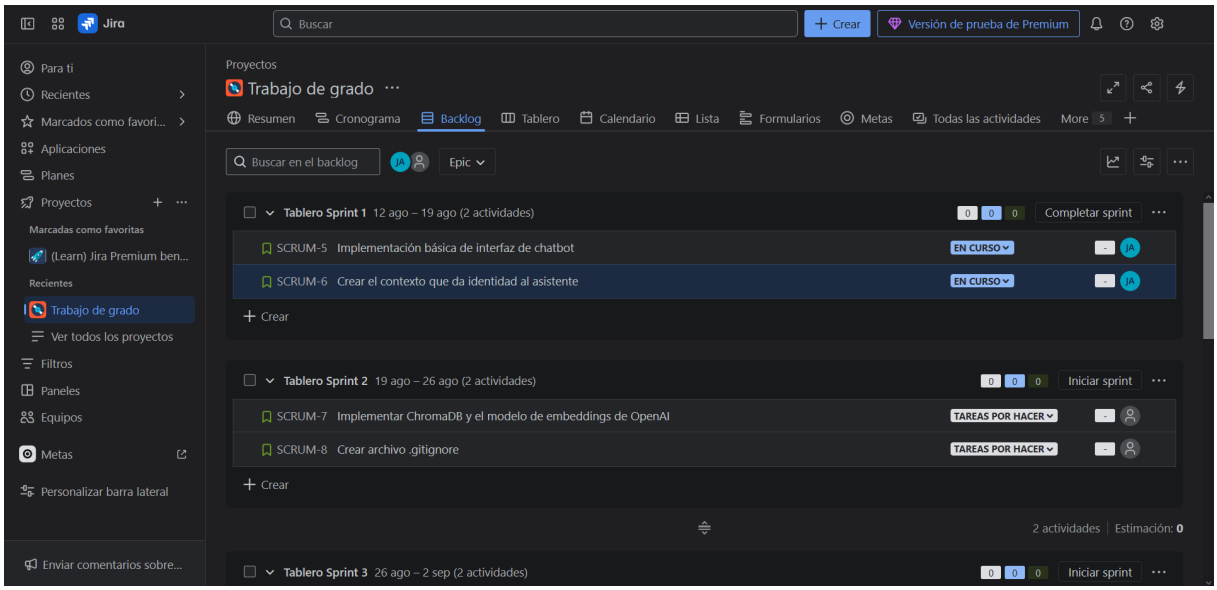

Figura 4: Sprints en Jira
Fuente: Tomado de [24]

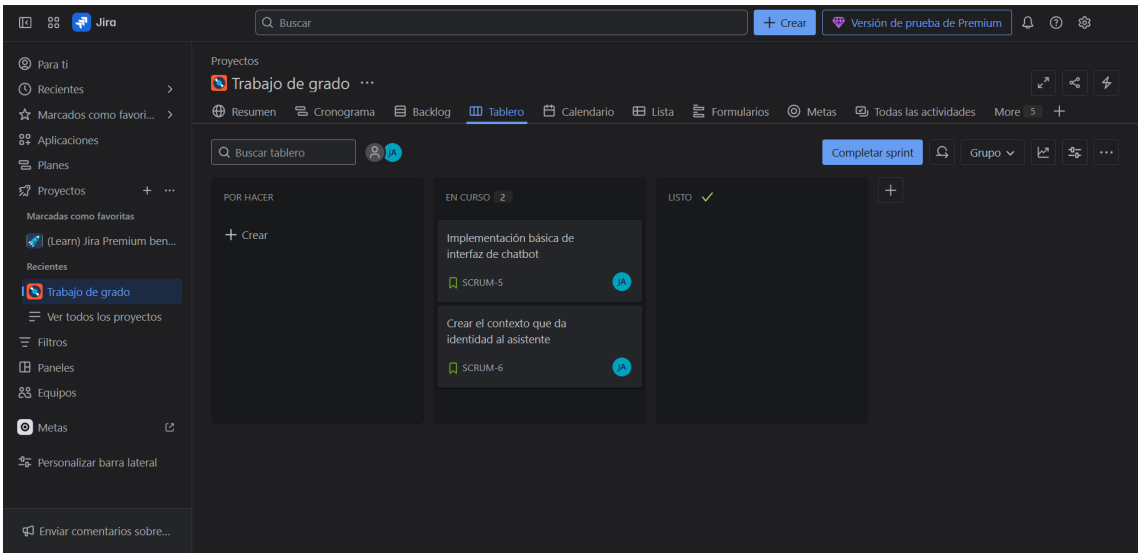

Figura 5: Tablero Kanban en Jira
Fuente: Tomado de [24]

El desarrollo del proyecto se llevó a cabo mediante historias de usuario, que son descripciones informales de una funcionalidad del software desde la perspectiva del usuario final o cliente . En este caso, dado que muchas tareas involucraban funcionalidades técnicas sin interacción directa del usuario, las historias se elaboraron desde el punto de vista del desarrollador. Las historias de usuario siguen comúnmente la estructura "persona + necesidad + propósito"[25]. De la *Tabla 3* a la *Tabla 10*, se presentan algunas de las historias de usuario del proyecto, añadiendo criterios de aceptación, los cuales definen las condiciones que deben cumplirse para considerar las historias de usuario completadas. Todas las historias de usuario se podrán ver en el anexo 1.

Tabla 3: HU1

ID	HU1
Tarea	Implementación básica de interfaz de *chatbot*
Tipo	*feature*
HU	Yo como usuario quiero interactuar con una interfaz similar a las usadas por los *chatbots* actuales como ChatGPT, Deepseek, etc.
Criterio aceptación	▪ La interfaz permite al usuario escribir un mensaje y enviarlo con un botón o con la tecla *Enter*. ▪ Se muestran los mensajes del usuario y las respuestas del *chatbot* en formato de conversación. ▪ La interfaz tiene un diseño visual limpio y sencillo, similar al de *chatbots* modernos.

Fuente: Elaboración propia

Tabla 4: HU2

ID	HU2
Tarea	Crear el contexto que da identidad al asistente
Tipo	*feature*
HU	Yo como desarrollador, quiero configurar un *prompt* base que guíe al *chatbot* para que se enfoque en temas de Ingeniería de Sistemas, para su comportamiento futuro.
Criterio aceptación	▪ El código del *chatbot* incluye un mensaje de sistema o *prompt* base explícito que instruye al modelo a enfocarse en temas de Ingeniería de Sistemas. ▪ El *prompt* base está escrito en un lenguaje claro y específico, e incluye indicaciones sobre rol, alcance temático y tono de respuesta.

Fuente: Elaboración propia

Tabla 5: HU3

ID	HU3
Tarea	Implementar *ChromaDB* y el modelo de *embeddings* de *OpenAI*
Tipo	*feature*
HU	Yo como desarrollador, quiero que el *chatbot* obtenga sus respuestas a partir de los resultados de búsqueda por similitud realizados en *ChromaDB*, utilizando fragmentos de documentos procesados como *embeddings* mediante el modelo de *OpenAI*, para garantizar que las respuestas estén con el contexto del contenido cargado.
Criterio aceptación	▪ Los documentos *pdfs* son procesados y divididos en fragmentos (*chunks*) para ser vectorizados. ▪ Cada fragmento es convertido en un *embedding* utilizando el modelo de *OpenAI* y almacenado correctamente en *ChromaDB*. ▪ Al recibir una pregunta del usuario, el sistema consulta *ChromaDB* y recupera los fragmentos usando búsqueda por similitud.

Fuente: Elaboración propia

Tabla 6: HU4

ID	HU4
Tarea	Crear archivo `.gitignore`
Tipo	*feature*
HU	Yo como desarrollador, quiero que la aplicación cuente con un archivo `.gitignore` configurado correctamente, para evitar que se suban al repositorio archivos temporales, de prueba o en constante cambio, para mantener un control limpio del código.
Criterio aceptación	▪ El archivo `.gitignore` está presente en el repositorio y contiene reglas que excluyen correctamente archivos y carpetas temporales, de prueba o en constante cambio (__pycache__/, `.env`, *.log, *.pvc, `db/`, etc.), evitando que se suban al control de versiones.

Fuente: Elaboración propia

Tabla 7: HU5

ID	HU5
Tarea	Mejorar el *prompt* de la variable de contexto del *chatbot*
Tipo	*refactor*
HU	Yo como desarrollador, quiero que el contexto proporcionado al *chatbot* esté claramente estructurado y definido, para asegurar que el modelo lo interprete correctamente y genere respuestas coherentes.
Criterio aceptación	▪ El mensaje de sistema del *chatbot* está redactado de forma clara, estructurada y alineada con el propósito del proyecto, y se valida que al ser incluido se generan respuestas dentro del dominio establecido.

Fuente: Elaboración propia

Tabla 8: HU6

ID	HU6
Tarea	Procesar el *prompt* del usuario para mejorar la búsqueda
Tipo	*feature*
HU	Yo como desarrollador, quiero que el *prompt* del usuario sea procesado mediante técnicas como la lematización o expandir la consulta con términos semejantes.
Criterio aceptación	▪ La consulta del usuario es lematizada correctamente, eliminando palabras vacías (*stopwords*), y generando una versión simplificada y generalizada del texto original. ▪ La consulta es ampliada con sinónimos y términos relacionados generados mediante un llamado a GPT. ▪ Las técnicas de lematización y expansión semántica son integradas correctamente en el flujo de procesamiento antes de ejecutar la búsqueda de información.

Fuente: Elaboración propia

Tabla 9: HU7

ID	HU7
Tarea	Implementar búsqueda híbrida con BM25
Tipo	*feature*
HU	Yo como desarrollador, quiero que el *chatbot* obtenga información a través de dos enfoques de búsqueda, para reducir posibles sesgos y mejorar la precisión en la recuperación de información.
Criterio aceptación	▪ El sistema implementa dos enfoques distintos de búsqueda: uno basado en similitud semántica mediante *embeddings* y otro basado en búsqueda tradicional (BM25). ▪ Ambos enfoques son ejecutados al momento de procesar una consulta del usuario, y sus resultados pueden combinarse o evaluarse para mejorar la comprensión de las respuestas.

Fuente: Elaboración propia

Tabla 10: HU8

ID	HU8
Tarea	Implementar el procesado de pdfs a texto plano
Tipo	*feature*
HU	Yo como desarrollador, quiero que los pdfs sean procesados a texto plano antes de generar los fragmentos que serán utilizados para crear la base de datos de *embeddings* para facilitar su lectura.
Criterio aceptación	▪ Los archivos pdf son convertidos exitosamente a texto plano sin pérdida de contenido. ▪ El texto plano conserva la estructura lógica de los párrafos, eliminando saltos de línea innecesarios o errores de segmentación. ▪ El texto plano resultante es utilizado como base para generar los fragmentos que luego serán convertidos en *embeddings* y almacenados en la base de datos.

Fuente: Elaboración propia

7. Información académica, administrativa y preguntas frecuentes

Con el fin de entender y reducir las interrupciones frecuentes del flujo de trabajo del personal administrativo, se estableció como primer objetivo recopilar sistemáticamente información relacionada a procesos, preguntas frecuentes y solicitudes que los estudiantes realizan de forma constante. Aunque muchas veces estas interrupciones son generadas por dudas legitimas de los estudiantes, terminan generando una tarea adicional para el personal, que tendrán que dejar momentáneamente sus labores. Por tal razón es fundamental identificar estos puntos clave para comprender en qué medida estas consultas pueden ser atendidas de manera automatizada. La información recopilada formará parte de la base de conocimiento del asistente, alimentado a través de documentos que describen los procesos que requieren los estudiantes.

Para llevar a cabo la recopilación de información, se diseñó una encuesta dirigida al personal administrativo de la oficina de Ingeniería de Sistemas, con el propósito de identificar las principales fuentes de interrupciones a su flujo de trabajo. Primeramente, se planteó el cuestionamiento sobre si el personal considera que se pueden ver interrumpidos en sus tareas por consultas redundantes de los estudiantes, obteniendo como resultado que un **87,5 %** de los encuestados marcaron que sí como se puede apreciar en la *Figura x*, además de que un **100 %** cree que un asistente de IA chatbot podría ser de utilidad en la tarea de brindar información y resolver preguntas frecuentes como muestra la *Figura x*.

Figura 6: Resultado de encuesta personal administrativo 1
Fuente: Elaboración propia

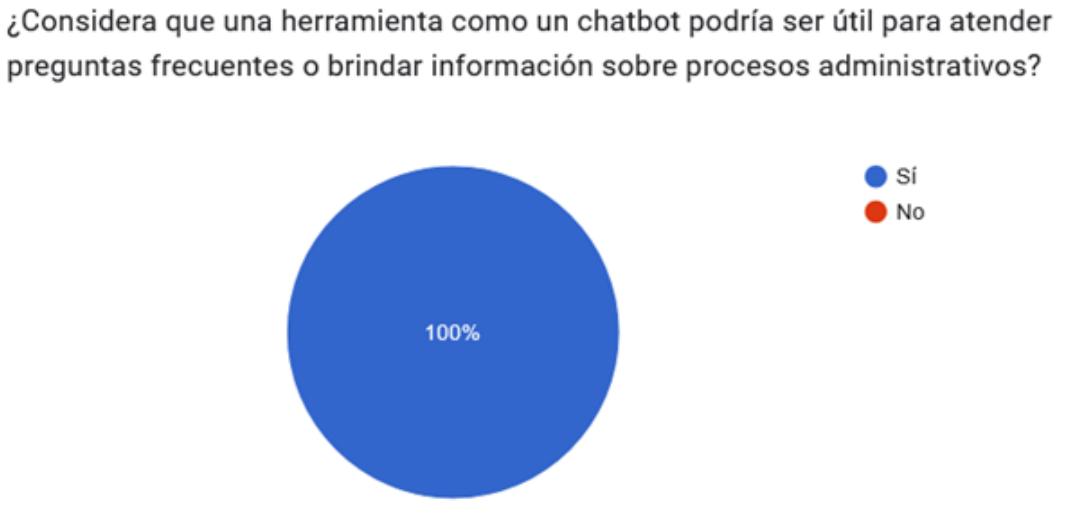

Figura 7: Resultado de encuesta personal administrativo 2
Fuente: Elaboración propia

Indagando con los integrantes de la oficina, se elaboró una lista con los temas de consulta más recurrentes por parte de los estudiantes. En dicha lista se incluyen los siguientes temas: **Bajos rendimientos, Traslados y transferencias, Equivalencia de asignaturas, Cambio de resolución (nueva malla), Matrícula de asignaturas, Matrícula de asignaturas de ciclo de Trabajo de Grado, Cupos, Prácticas profesionales, Grados, Entrega de trabajo de grado a biblioteca, Validación de idioma extranjero, Presentación de examen de proficiencia de idioma extranjero, Cursos de vacaciones, Adiciones y cancelaciones de asignaturas, Carta de presentación de estudiantes para empresa**.

Los resultados obtenidos indican que los temas más consultados por los estudiantes son **Bajos rendimientos académicos, Traslados y transferencias** y **Matrícula de asignaturas**, seleccionados por un **75 %** de los encuestados. Estos son seguidos por **Equivalencia de asignaturas** y **Grados**, con un **62,5 %** de las respuestas. En la *Figura X* se presenta el porcentaje de votos correspondiente a cada una de las opciones incluidas en la encuesta.

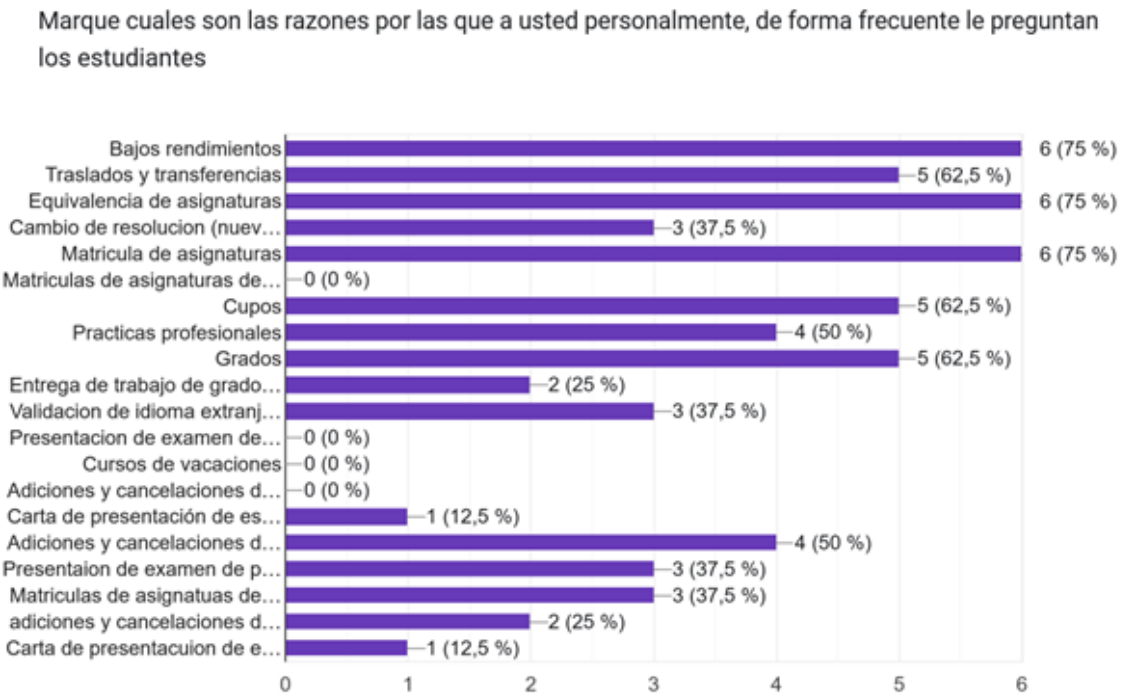

Figura 8: Resultado de encuesta personal administrativo 3
Fuente: Elaboración propia

Teniendo en cuenta los temas recopilados, se llevó a cabo una búsqueda y revisión de documentos oficiales emitidos por la Universidad del Valle, con el propósito de identificar información que diera respuesta principalmente a los temas más consultados por los estudiantes, no obstante, también se intentó incluir documentación relacionada con el resto de los temas, con el fin de construir una base de conocimiento lo más completa posible. En la *Tabla x* se muestra el resultado de los documentos seleccionados y la relevancia que tiene en relación con los temas identificados.

Tabla 11: Documentos normativos y orientadores relevantes para el programa de Ingeniería de Sistemas de la Universidad del Valle

Nombre	**Autor**	**Relevancia**
Resolución 048 29 de abril de 2010	Consejo Académico Univalle	Por el cual se define el currículo del Programa Académico de Ingeniería de Sistemas. Es la resolución por la cual se crea la antigua vigencia del currículo de Ingeniería de Sistemas.
PEP 048	Dirección del Programa Académico Ingeniería de Sistemas	Establece las bases para la formación integral de profesionales en este campo, enfocándose en el desarrollo de competencias técnicas, sociales y éticas. Además, busca promover la investigación, la innovación tecnológica y la proyección social, alineando el currículo con los desafíos y oportunidades del entorno global y local.
Resolución 047 16 de abril de 2020	Consejo Académico Univalle	Por el cual se define el currículo del Programa Académico de Ingeniería de Sistemas. Es la resolución por la cual se crea la nueva vigencia del currículo de Ingeniería de Sistemas.
PEP 047	Dirección del Programa Académico Ingeniería de Sistemas	Establece la identidad y orientación de un programa académico, definiendo sus objetivos formativos, estructura curricular y su relación con el entorno social, político y cultural, guiando el proceso educativo hacia el cumplimiento de los resultados de aprendizaje, investigación y proyección social.
Resolución 106 17 junio de 2021	Consejo Académico Univalle	Por la cual se corrige un error de forma en la Resolución No. 047 del 16 de abril de 2020 del Consejo Académico. Hace correcciones a los nombres de algunas asignaturas que están mal en la resolución 047.
Resolución 074 14 de mayo de 2015	Consejo Académico Univalle	Por la cual se aprueban las modalidades de trabajo de grado para la Facultad de Ingeniería.
Resolución 136 22 de diciembre 2017	Consejo Académico Univalle	Por la cual se reglamentan las condiciones para la creación y reforma de los programas de formación de pregrado de la Universidad del Valle. Esta resolución reglamenta cómo deben estar estructurados los nuevos programas de pregrado y cuyos programas se deban actualizar.

Continúa en la siguiente página

Nombre	Autor	Relevancia
Resolución 162 02 de septiembre 2021	Consejo Académico Univalle	Por la cual se reglamenta el examen de clasificación en idioma extranjero y sus respectivas homologaciones en los programas de pregrado. Mediante esta resolución se rige el examen mediante el cual los estudiantes se clasifican en un nivel de inglés según el Marco Común Europeo para validar los módulos de inglés requeridos por el programa.
Acuerdo 009 13 de noviembre de 1997	Consejo Superior Univalle	Por el cual se reglamentan las actividades académicas de los estudiantes regulares pertenecientes a los Programas Académicos de Pregrado. Acerca de: estudiantes y asignaturas, derechos y deberes del estudiante, representación estudiantil, proceso de evaluación, calificaciones, registro de matrícula, repetición de materias, bajos rendimientos, solicitudes y reclamos de los estudiantes, reintegros, traslados y transferencias, equivalencias, cursos vacacionales, trabajos de grado, estímulos académicos, grados, régimen disciplinario, certificaciones, reglamentos internos y otras disposiciones.
Acuerdo 006 28 de junio de 2023	Consejo Superior Univalle	Por medio del cual se modifica el CAPÍTULO VIII <DE LOS BAJOS RENDIMIENTOS ACADÉMICOS> del Acuerdo 009 de 1997 o Reglamento Estudiantil de Pregrado de la Universidad del Valle. Y que, condiciona para caer en bajo rendimiento que el promedio del semestre sea inferior a uno bajo rendimiento que solo aplica para estudiantes matriculados a partir del 2024-1 en adelante.
Plan de transición de la resolución 048 a la 047 – equivalencia de asignaturas	Dirección del Programa Académico Ingeniería de Sistemas	Mediante este documento se muestran las asignaturas de la antigua resolución que son equivalentes a las asignaturas de la nueva, con el fin de apoyar a estudiantes que se hagan al proceso de cambio al nuevo currículo de modo que puedan homologar algo de su progreso.

Fuente: Elaboración propia

La Universidad del Valle posee una normatividad extensa y sólida; sin embargo, esto dificulta la identificación de documentos específicos que aborden en profundidad los temas exactos identificados. Aunque algunos documentos como el Acuerdo 009 tratan múltiples temáticas, no lo hacen a profundidad, por ejemplo, la matrícula de trabajo de grado, el documento trata ciertos aspectos generales, pero no contiene requisitos específicos que maneja la oficina para este proceso. Debido a esto, algunos miembros de la oficina colaboraron en la redacción de un documento complementario que detalla cómo se llevan a cabo diversos procesos administrativos en el programa de Ingeniería de Sistemas Sede Tuluá. Este documento puede consultarse en el anexo 2.

8. Selección y Justificación de la Técnica de PLN

8.1. Investigación

En el entorno actual de la educación superior, la calidad del servicio y la velocidad de atención a los estudiantes son aspectos fundamentales para lograr su satisfacción, especialmente en un contexto marcado por la globalización y el avance de la tecnología. La necesidad de una comunicación más rápida y efectiva ha llevado a muchas instituciones a incorporar nuevas herramientas tecnológicas. Entre ellas, los chatbots basados en Inteligencia Artificial (IA) han ganado importancia como una buena opción para mejorar la atención estudiantil, su implementación permite automatizar la respuesta a consultas frecuentes, lo que no solo facilita el acceso a la información para los estudiantes, sino que también libera al personal administrativo para enfocarse en tareas más complejas, modernizando así los canales de atención [26].

Esta tendencia ha sido respaldada por múltiples experiencias en universidades donde los chatbots se han usado para resolver dudas comunes en distintos temas académicos y administrativos. Los estudios han mostrado altos niveles de satisfacción entre los usuarios, gracias a la facilidad de uso y la calidad de las respuestas ofrecidas, lo que evidencia su utilidad como canal de comunicación [27]. Por ejemplo, algunas instituciones han desarrollado asistentes virtuales para ayudar a los estudiantes con temas como matrículas y calificaciones, integrando estas funciones en aplicaciones móviles con interfaces tipo mensajería, estas soluciones han demostrado ser efectivas para atender necesidades concretas del sector educativo, mejorando la disponibilidad de información [28].

El uso de chatbots no se limita al ámbito educativo, en sectores como el legal, se han desarrollado asistentes virtuales para apoyar tareas complejas como la búsqueda de información jurídica, estos sistemas emplean tecnologías como el procesamiento del lenguaje natural (PLN) y algoritmos de aprendizaje automático, que permiten interpretar las preguntas de los usuarios y ofrecer respuestas útiles y personalizadas. Esto demuestra que los chatbots pueden adaptarse a contextos que exigen precisión y eficiencia, ofreciendo acceso rápido y automatizado a información especializada [29].

Durante la pandemia, la adopción de herramientas digitales en las universidades se incrementó exponencialmente, destacando aún más la importancia de los chatbots. En respuesta a las restricciones de la presencialidad, se desarrollaron sistemas inteligentes que facilitaron la comunicación remota con los estudiantes, aunque muchos de estos proyectos aún se quedaron en fase de prueba, mostraron su potencial para contribuir a la modernización de los medios de atención, ofreciendo respuestas rápidas y eficaces a las necesidades de los estudiantes [30].

Como se mencionó anteriormente, la utilidad de los chatbots se evidenció de gran manera durante la emergencia sanitaria global, cuando varias universidades adoptaron estas herramientas para mantener la atención a sus estudiantes a pesar de la suspensión de actividades presenciales. Los estudios realizados en este contexto muestran que los chatbots fueron fundamentales para garantizar la continuidad del servicio, consolidándose como una herramienta clave dentro de las estrategias de las instituciones [31].

En la búsqueda de una mejor experiencia para el usuario, se han propuesto sistemas de

búsqueda académica basados en chatbots que priorizan una respuesta rápida, una interfaz fácil de usar y una interacción intuitiva. Las pruebas realizadas con estudiantes indican una notable reducción en los tiempos de espera y una alta aceptación del asistente, lo que refuerza el valor de integrar servicios cognitivos en la nube para ofrecer una atención más eficiente y personalizada [32]. En otros sectores como el de la salud, se han utilizado chatbots en páginas web para responder preguntas frecuentes en hospitales, ayudando a mejorar la transparencia y la confianza del público, estos sistemas, basados también en PLN y aprendizaje automático, permiten ofrecer información precisa de manera inmediata, lo que confirma su utilidad en entornos donde la fiabilidad de la información es crítica [33].

Con los avances recientes en inteligencia artificial, especialmente en PLN, los Grandes Modelos de Lenguaje (LLM, por sus siglas en inglés) han potenciado la capacidad de los chatbots para generar respuestas coherentes y contextualizadas, sin embargo, estos modelos también presentan limitaciones, como generar respuestas coherentes, pero en realidad incorrectas (conocidas como "alucinaciones") o no poder acceder a información específica y actualizada. Para superar estos desafíos, ha surgido el enfoque de Generación Aumentada por Recuperación (RAG, por sus siglas en inglés), que combina las capacidades de los LLM con el acceso a bases de conocimiento externas, lo que permite generar respuestas basadas en información verificada, aumentando considerablemente su precisión y utilidad [19].

Un caso que ejemplifica el uso de LLMs en contextos administrativos es el desarrollo de chatbots para optimizar procesos en empresas, donde estas herramientas han demostrado ser útiles para automatizar tareas y reducir la carga operativa, lo que también puede reflejarse al entorno universitario [34].

RAG se presenta como una técnica avanzada que mejora el rendimiento de los LLM al integrarlos con mecanismos de recuperación de información. Diversos estudios han demostrado que esta combinación aumenta notablemente la precisión de las respuestas, al recuperar información relevante antes de generar la respuesta, se reduce la probabilidad de errores y se incrementa la calidad de las interacciones [35].

Un ejemplo de su aplicación es el uso de RAG junto con GPT-4 para analizar información altamente especializada, como notas clínicas no estructuradas. Este enfoque ha demostrado que, al limitar el análisis a los fragmentos más relevantes, se mejora tanto la precisión como la eficiencia del sistema, ofreciendo respuestas más confiables [36].

En el ámbito educativo, los chatbots basados en RAG han demostrado ser útiles como apoyo al aprendizaje. En un estudio sobre su uso en cursos en línea, los estudiantes valoraron positivamente la capacidad del chatbot para ofrecer explicaciones y ejemplos relacionados con el contenido del curso, esto demuestra que, al eliminar errores comunes y generar respuestas adaptadas al contexto, estos sistemas enriquecen significativamente la experiencia de aprendizaje [37].

La combinación de LLM y RAG ofrece una solución sólida para realizar búsquedas de información precisas y actualizadas, accediendo a datos específicos en tiempo real, esta capacidad es esencial para desarrollar chatbots confiables, capaces de responder con información relevante y verificada, lo que supera las limitaciones del conocimiento estático de los modelos entrenados previamente [38].

En este marco, el uso de un chatbot con tecnología RAG para responder preguntas

frecuentes y asistir en procesos administrativos en universidades representa una estrategia altamente efectiva. Este enfoque, similar a los utilizados para facilitar procesos de admisión, no solo mejora la precisión de las respuestas y reduce los errores, sino que también permite ofrecer atención personalizada y eficiente, transformando la experiencia de interacción del estudiante con los servicios institucionales [39].

RAG permite construir fácilmente una base de conocimiento especializada, que el chatbot puede consultar al momento de responder. Esta técnica combina la recuperación de información relevante con la capacidad del modelo de lenguaje para interpretar y seleccionar los datos más adecuados, generando respuestas claras, útiles y agradables para el usuario final. Además de mejorar la experiencia de los estudiantes al facilitarles el acceso a información administrativa confiable y bien presentada, esta tecnología también representa un beneficio para el personal administrativo, al reducir significativamente el volumen de consultas repetitivas y permitirles enfocarse en actividades de mayor complejidad e impacto.

8.2. Elección de LLM

Al optar por utilizar la técnica de Generación Aumentada por Recuperación (RAG) para el desarrollo del chatbot se implica no solo la capacidad del sistema para buscar y recuperar información relevante desde una base de conocimiento específica, sino también la necesidad de un modelo de lenguaje avanzado que pueda redactar respuestas claras y útiles para los usuarios a partir de la información obtenida.

Durante las primeras etapas del proyecto, se usaron algunos modelos razonadores como o1-mini, o3-mini y deepseek-r1, dado a su costo moderado, se miraban como opciones viables, sin embargo, estos modelos son diseñados para tareas complejas que requieren que estos usen largos periodos de razonamiento para encontrar una solución adecuada al problema, provocando que se tuvieran tiempos de respuesta prolongados y una experiencia poco fluida, por lo que rápidamente se descartaron estas opciones que se alejaban del foco de ofrecer respuestas ágiles y eficientes a las consultas.

Posteriormente, se probaron los modelos GPT-4o y GPT-4o-mini, con los que ya se contaba con cierta experiencia, siendo modelos usados en un sin número de implementaciones y que han demostrado ser estables, precisos y de fácil integración, destacando GPT-4o-mini por su buena relación costo-rendimiento, sin embargo, durante el proceso de desarrollo OpenAI lanzó al mercado su nueva familia de modelos GPT-4.1, que incluye las versiones GPT-4.1, GPT-4.1-mini y GPT-4.1-nano. De estos, GPT-4.1-mini resultó ser el más adecuado para el proyecto, ya que tiene características clave como lo son su gran ventana de contexto de más de un millón de tokens a comparación de otros modelos que no superan los 200,000 como se ve en la *Tabla 12* [40] [41], ideal para trabajar con un número grande de fragmentos de la base de conocimiento, también destaca su costo, aproximadamente un 84 % más barato por millón de tokens en comparación con GPT-4o, además, al ser un modelo pequeño, su velocidad es muy alta, pero mostrando una potencia casi equiparable a la de GPT-4o, como se muestra en la *Figura 9* [42]. Estas características lo hacen ideal para el sistema RAG donde se tienen que buscar y procesar de manera rápida los fragmentos de información para garantizar una buena experiencia de uso.

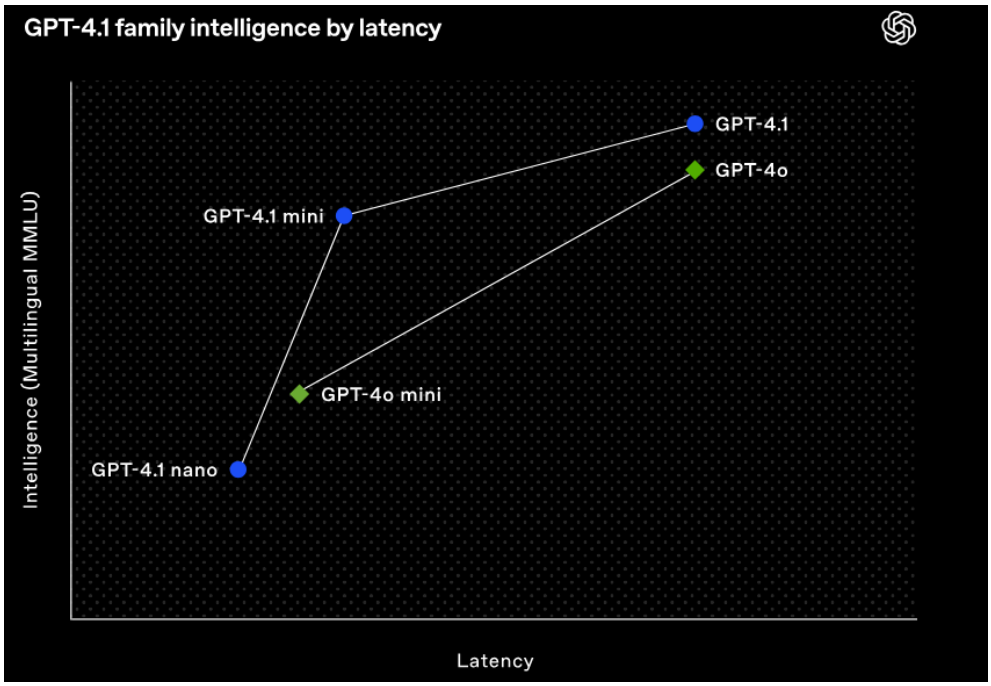

Figura 9: Familia GPT-4.1 vs GPT-4o
Fuente: Tomado de [42]

Tabla 12: Comparación de precios y capacidades entre modelos de OpenAI

Modelo	**Input**	**Output**	**Ventana de contexto**
gpt-4.1	$2.00	$8.00	1.047.576
gpt-4.1-mini	$0.40	$1.60	1.047.576
gpt-4.1-nano	$0.10	$0.40	1.047.576
gpt-4o	$2.50	$10.00	128.000
gpt-4o-mini	$0.15	$0.60	128.000
o1-mini	$1.10	$4.40	128.000
o3-mini	$1.10	$4.40	200.000

Fuente: Tomado de [40][41]

A pesar de que GPT-4.1-mini está diseñado para operar con menor potencia de procesamiento, ha demostrado desempeños elevados, logrando estar entre los mejores modelos de su misma clase, como se muestra en la *Figura 10*, donde se muestran sus resultados en 7 distintos test [43]. Esto lo convierte en una alternativa equilibrada que se adapta bien a las necesidades del proyecto, combinando eficiencia, calidad de respuesta y viabilidad económica.

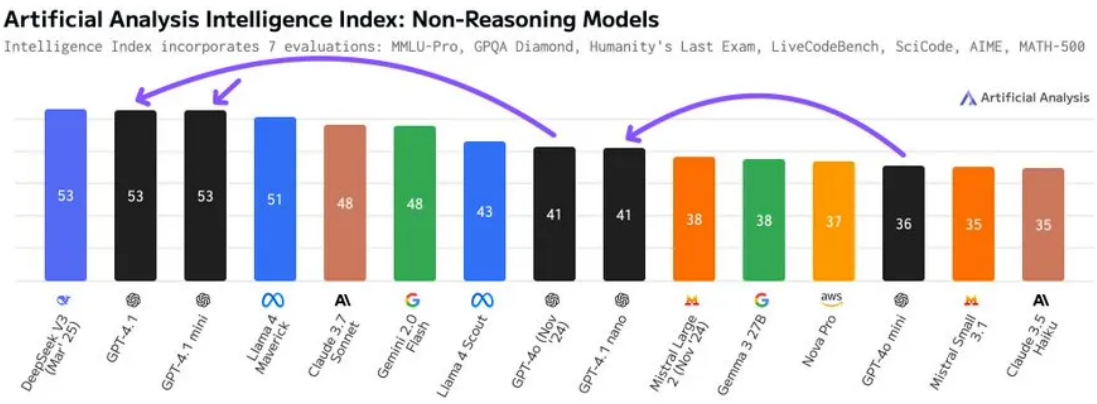

Figura 10: Familia GPT-4.1 vs otros LLMs
Fuente: Tomado de [43]

9. Implementación del modelo

Para el desarrollo del sistema se consideró inicialmente el uso del SDK oficial de OpenAI, dado que se trata de una herramienta nativa para interactuar con sus modelos de lenguaje, sin embargo, se optó por utilizar LangChain, debido a que ofrece ventajas para construir sistemas basados en RAG. Aunque el SDK de OpenAI permite una integración directa con los modelos de generación de texto y embeddings, LangChain ofrece una gama amplia de herramientas, diseñadas específicamente para facilitar este tipo de desarrollos. Esta librería no solo permite el uso estable y eficiente de los modelos de OpenAI a través de su API, sino que también simplifica tareas clave en la construcción del pipeline RAG [44].

LangChain proporciona mecanismos para dividir textos en fragmentos (chunks) de manera eficiente, lo que resulta fundamental para la indexación y recuperación precisa de la información, asimismo, se integra de forma nativa con bases de datos vectoriales como ChromaDB, facilitando tanto el almacenamiento como la búsqueda de los embeddings creados [45].

Además, LangChain tiene funcionalidades que hacen que la implementación del streaming de tokens sea mucho más sencilla [46], permitiendo mostrar la respuesta generada por el modelo de forma progresiva, a medida que este responde, esta funcionalidad mejora notablemente la experiencia del usuario al reducir la percepción de espera.

Después de investigar sobre los RAG, se identificó una variante especialmente interesante por su eficacia, los RAG Híbridos. Esta propuesta combina diferentes formas de recuperar información, lo que permite mejorar considerablemente la selección de los fragmentos de texto más relevantes. El propósito de integrar varios métodos es reducir posibles limitaciones asociadas a depender de una sola estrategia [47].

Desde un principio, los embeddings se consideraron una herramienta indispensable, su capacidad para comprender el contexto y el significado de un texto los hace muy útiles para identificar contenido realmente relevante. Al convertir el texto en una representación numérica, los embeddings permiten realizar búsquedas más precisas, ya que no se limitan a buscar palabras exactas, sino que intentan entender la idea detrás de lo que se consulta.

Para complementar la búsqueda por embeddings, se exploraron otras opciones como segundo método de recuperación, y uno de los algoritmos que con más frecuencia se menciona es BM25, este algoritmo mejora la técnica TF-IDF, y ha demostrado ser especialmente útil en tareas de búsqueda de información [48].

BM25 se basa en la frecuencia con la que aparecen los términos dentro de un documento y en el conjunto general de documentos, pero también tiene en cuenta factores como la longitud del texto, lo que ayuda a mantener un equilibrio justo en la evaluación de la relevancia, características que hacen que este algoritmo dé grandes resultados en proyectos de la misma naturaleza [49].

La combinación de embeddings con BM25 permite construir una estrategia de recuperación más completa, que equilibra tanto el significado general del contenido como la importancia de las palabras que lo componen. .

El funcionamiento del modelo RAG inicia cuando el usuario envía un prompt. Este prompt se procesa mediante el modelo de embedding, y su representación vectorial se

utiliza para realizar una búsqueda por similitud en ChromaDB, que contiene vectores previamente almacenados. Paralelamente, el mismo prompt se emplea para ejecutar el algoritmo BM25, el cual realiza una búsqueda en un corpus textual guardado en un archivo .pkl, previamente procesado a través del pipeline del sistema.

Una vez se obtienen los resultados de ambos métodos de, estos se combinan, eliminando cualquier resultado duplicado. Los fragmentos seleccionados se integran con el prompt original del usuario y el contexto que define la identidad del chatbot , y se envían al LLM. Finalmente, el LLM genera una respuesta, que es presentada al usuario. La *Figura 11* muestra el flujo completo del funcionamiento del modelo RAG y el repositorio donde se encuentra el código de la aplicación se puede encontrar en el anexo 3.

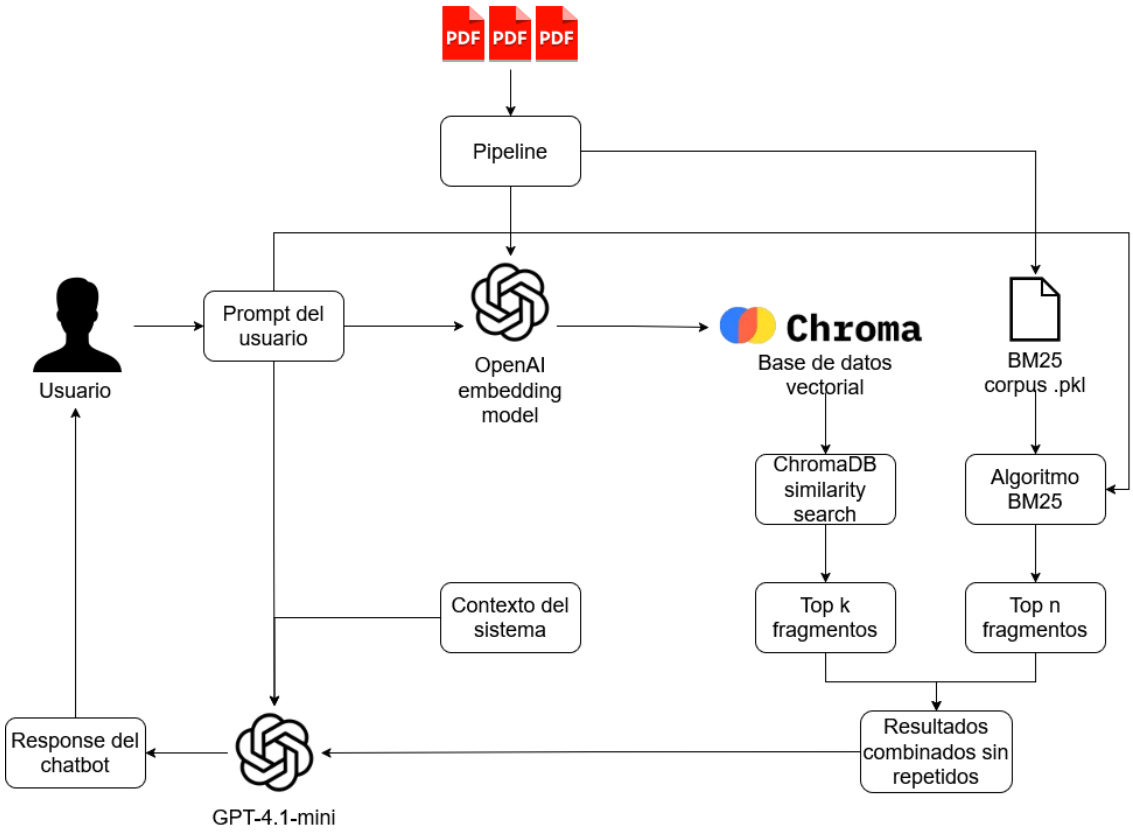

Figura 11: Flujo del funcionamiento del modelo RAG
Fuente: Elaboración propia

10. Desarrollo de la aplicación web

10.1. Tecnologías de desarrollo

Para el desarrollo del chatbot basado en GPT-4.1-mini, se eligió Streamlit como framework para construir la aplicación web. Esta elección se fundamentó en consideraciones tanto técnicas como prácticas, pensando en los objetivos y limitaciones del proyecto.

Streamlit es una herramienta pensada especialmente para el desarrollo de aplicaciones de ciencia de datos y aprendizaje automático, lo que lo convierte en una opción adecuada cuando se trabaja con modelos de inteligencia artificial. A diferencia de frameworks más generales como React o Angular, Streamlit ofrece componentes diseñados específicamente para visualizar modelos, mostrar métricas de rendimiento y presentar resultados de forma clara e interactiva.

Uno de sus principales beneficios es su compatibilidad directa con las bibliotecas más utilizadas del ecosistema científico de Python, como pandas para el manejo de datos, matplotlib y Plotly para la creación de gráficos, y scikit-learn para el cálculo de métricas.

En el caso particular de los sistemas de chatbot e interfaces tipo chat, es fundamental gestionar adecuadamente el flujo del dialogo, el estado de la sesión y la forma en que se presentan los mensajes. Streamlit incluye componentes nativos como st.chat_message() y st.chat_input(), que fueron diseñados precisamente para ese propósito, simplificando así el desarrollo de una experiencia conversacional funcional desde el inicio.

Otro aspecto clave en sistemas que hacen uso de modelos de lenguaje es el tiempo de respuesta, ya que la generación de texto puede tomar varios segundos. En este sentido, Streamlit ofrece la función st.write_stream(), que permite mostrar la respuesta del modelo de manera progresiva, a medida que se genera. Esta característica mejora considerablemente la experiencia del usuario, al reducir la sensación de espera y hacer más fluida la interacción [50].

Una de las ventajas más importantes de Streamlit es que maneja automáticamente la comunicación en tiempo real entre el servidor y la interfaz, gracias al uso interno del protocolo WebSocket, este protocolo permite establecer una conexión continua y bidireccional entre la aplicación web y el servidor, lo que significa que los datos pueden enviarse y recibirse al mismo tiempo, sin necesidad de hacer múltiples solicitudes como ocurre en los modelos tradicionales. Esto es especialmente útil en aplicaciones que requieren mostrar información a medida que se genera [51], como es el caso del streaming de respuestas desde un modelo de lenguaje. Al estar integrado en Streamlit, el uso de WebSocket no requiere configuración adicional ni conocimientos técnicos avanzados, lo que simplifica significativamente el desarrollo y mejora la experiencia del usuario final.

Al desarrollar este proyecto los recursos de tiempo y personal fueron limitados, por lo que contar con herramientas que optimicen el proceso de desarrollo fue clave. Streamlit permite construir aplicaciones web completas utilizando únicamente Python, generando aplicaciones completamente responsive, quitando la necesidad de aprender tecnologías frontend como HTML, CSS o JavaScript, esta ventaja resulta especialmente útil, puesto que la atención se debió centrar en diseñar, probar y validar la funcionalidad del chatbot, más que en desarrollar interfaces complejas, gracias a esta simplicidad, es posible construir

prototipos funcionales en poco tiempo.

Para respaldar la elección, se llevó a cabo una comparación con React y Angular en una escala de 1 a 5, mientras más alta la nota es mejor, más fácil o más útil, como se aprecia en la *Tabla 13*, las tecnologías más usadas para frontend, esta comparativa tuvo en cuenta criterios relevantes para desarrollar los objetivos específicos del proyecto, concluyendo que Streamlit representaba una buena alternativa para este caso particular.

Tabla 13: Comparación entre frameworks para el desarrollo del chatbot

Criterio	**Streamlit**	**React**	**Angular**
Tiempo de desarrollo	5	2	2
Curva de aprendizaje	4	2	1
Integración con Python	5	1	1
Componentes para chatbot	5	3	3
Diseño responsive automático	5	2	2
Total	**24**	**10**	**9**

Fuente: Elaboración propia

10.2. Interfaz de la aplicación

Para el diseño de la interfaz de la aplicación, se buscó que esta resultara familiar y fácil de usar, inspirándose en las interfaces comunes de los chatbots de inteligencia artificial, para ofrecer una experiencia de navegación intuitiva, reduciendo al mínimo la curva de aprendizaje de los usuarios. La idea consistió en una vista principal basada en una interfaz conversacional tipo chat, en la que el usuario pueda interactuar directamente con el modelo de IA. Además, una barra lateral (sidebar), que permite acceder a otras vistas donde se presentan resúmenes de documentos oficiales de la Universidad, acompañados de su respectivo PDF embebido para consulta directa. Para representar visualmente esta idea, se utilizaron mockups elaborados en la plataforma Figma. Estas representaciones gráficas sirvieron como guía del diseño final y pueden observarse desde la *Figura 12* hasta la *Figura 17* y el codigo del resultado final se encuentra en el anexo 3.

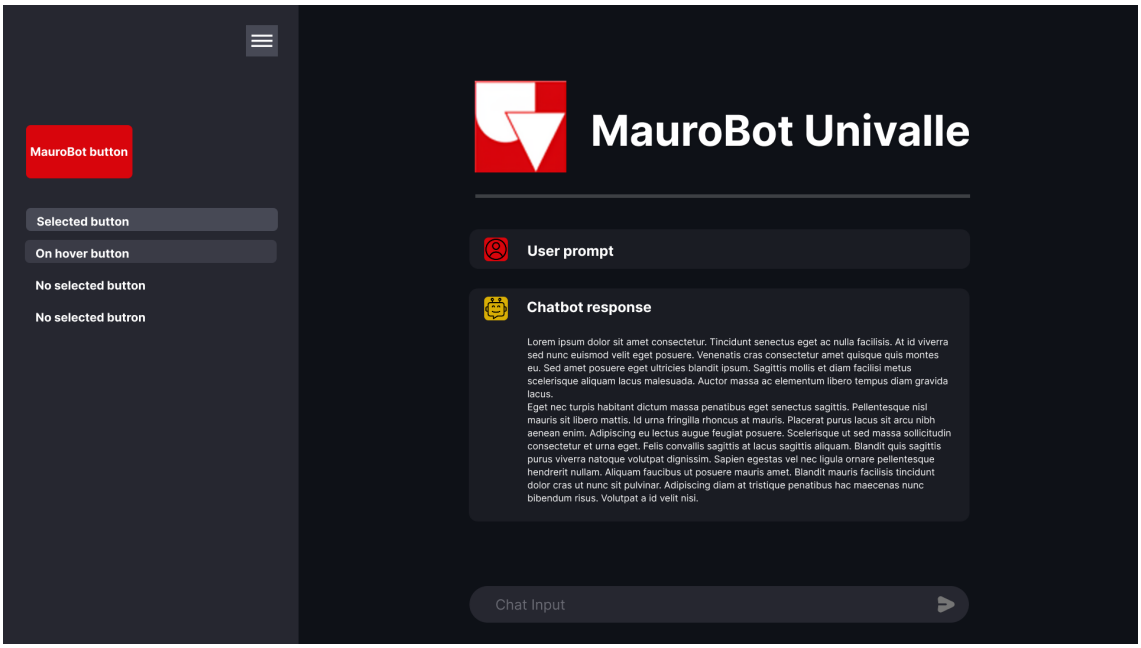

Figura 12: Mockup vista principal chatbot con sidebar desplegada
Fuente: Elaboración propia

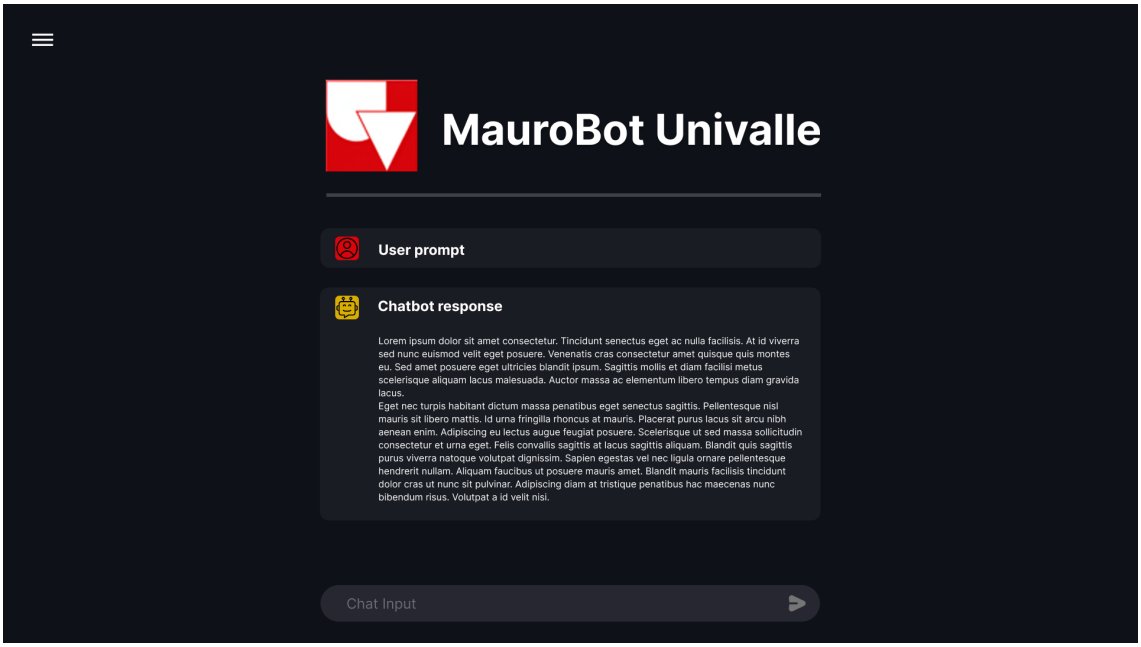

Figura 13: Mockup vista principal chatbot con sidebar sin desplegar
Fuente: Elaboración propia

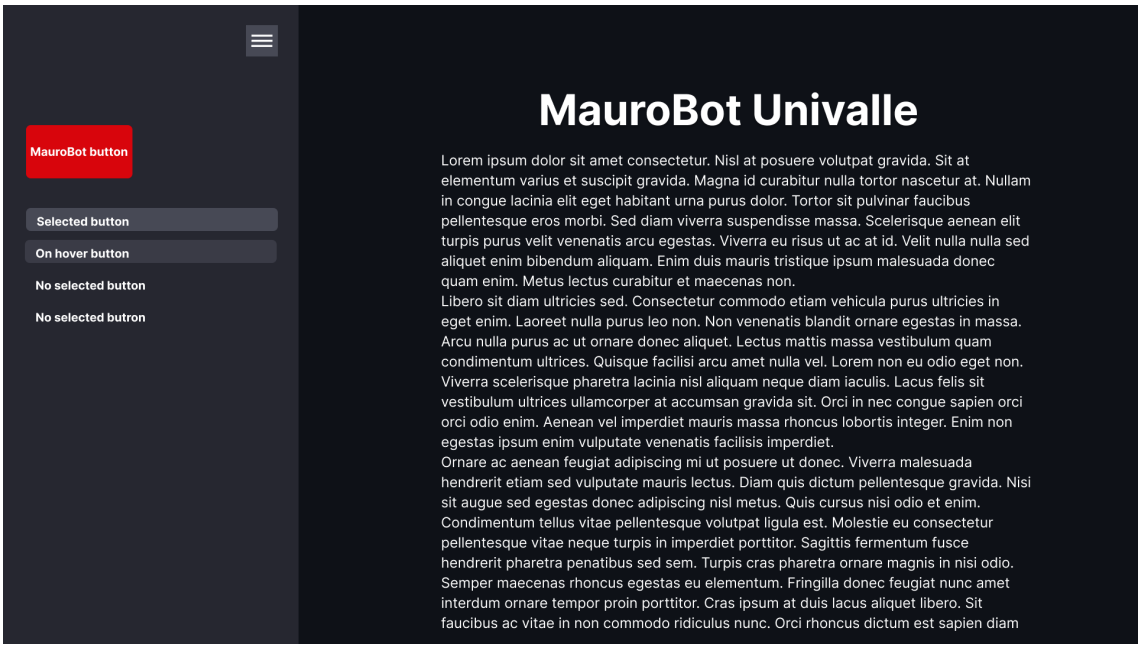

Figura 14: Mockup otras vistas con sidebar desplegada
Fuente: Elaboración propia

Figura 15: Mockup otras vistas con sidebar sin desplegar
Fuente: Elaboración propia

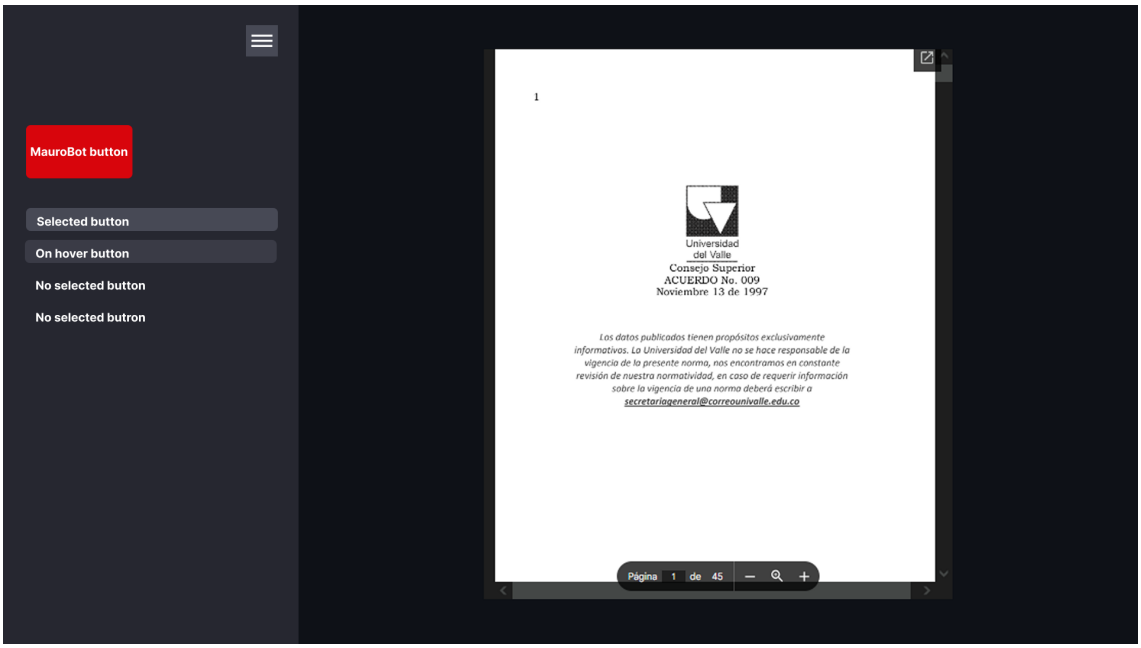

Figura 16: Mockup PDFs embebidos con sidebar desplegada
Fuente: Elaboración propia

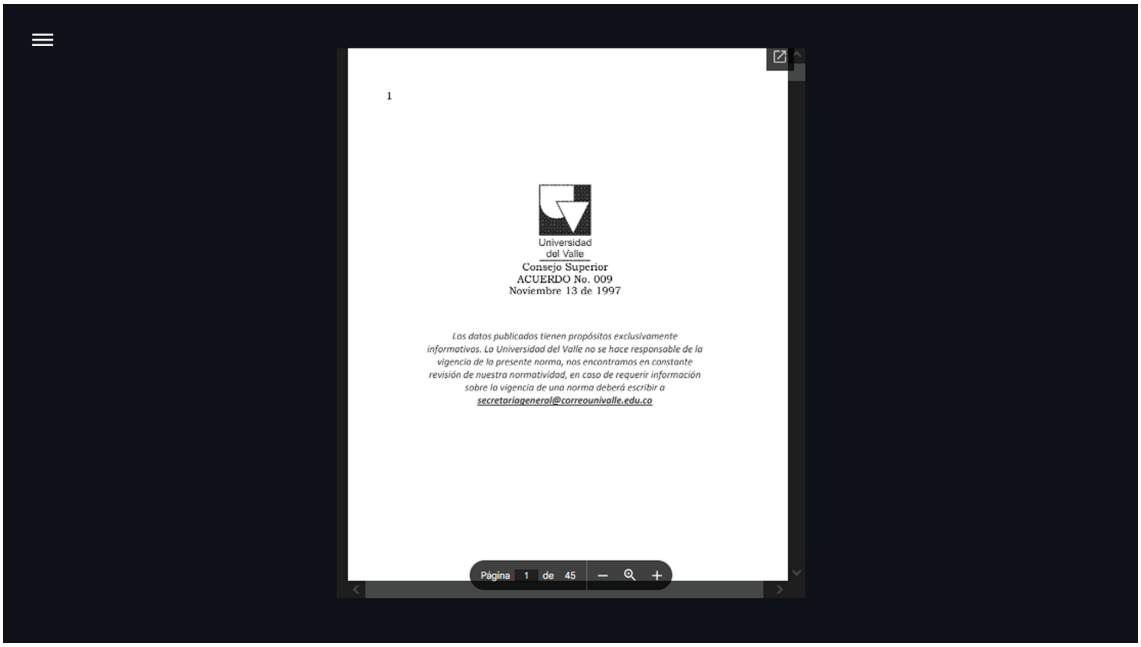

Figura 17: Mockup PDFs embebidos con sidebar sin desplegar
Fuente: Elaboración propia

10.3. Casos de uso

A diferencia de sistemas complejos que requieren múltiples roles o funcionalidades, este proyecto se centró en una propuesta sencilla pero poderosa, un asistente conversacional que responde preguntas sobre dudas y procesos de la Universidad, junto con el acceso a documentos oficiales y sus respectivas síntesis, por esta razón, el número de casos de uso es reducido, pero todos cumplen bien su función y están alineados con los objetivos del sistema.

Esta decisión responde a la naturaleza del proyecto, que no buscó desarrollar una aplicación con muchas funciones, sino optimizar en particular la interacción con el chatbot. La idea principal fue lograr que esta funcionalidad fuera eficaz, rápida y precisa, garantizando respuestas útiles a partir de las consultas de los usuarios, con una experiencia fluida.

Los diagramas de casos de uso permitieron representar gráficamente las acciones que puede realizar el usuario, como enviar preguntas al chatbot y acceder a las vistas con resúmenes de los documentos. Estos casos de uso ayudaron a prever las necesidades del usuario como consultar rápidamente la información sin tener que navegar por múltiples menús o interfaces complejas. Los dos casos de uso que se plantearon fueron:

UC01 – Consultar información al chatbot

Actor principal: Usuario
Descripción: El usuario ingresa una pregunta en la interfaz principal y recibe una respuesta generada por el modelo de IA.
Precondición: El usuario ha accedido al sistema.
Flujo principal:

1. El usuario escribe una pregunta en el campo de entrada.
2. El sistema envía la consulta al modelo de lenguaje (LLM).
3. El modelo busca información relevante (usando el motor RAG) y genera una respuesta.
4. El sistema muestra la respuesta al usuario en la interfaz de chat.

Postcondición: El usuario obtiene una respuesta relacionada con la documentación institucional.

UC02 – Navegar entre vistas de documentos

Actor principal: Usuario
Descripción: El usuario selecciona una opción en la barra lateral para consultar el resumen y el documento PDF asociado.
Precondición: El usuario ha accedido al sistema.
Flujo principal:

1. El usuario visualiza la barra lateral.

2. Selecciona uno de los documentos disponibles.
3. El sistema redirige a una vista que contiene:
    - Un resumen del documento.
    - El documento embebido en PDF para su lectura.

Postcondición: El usuario puede leer tanto el resumen como el documento oficial sin necesidad de salir de la aplicación.

En la *Figura 18* se ve el diagrama de casos de uso que muestra la interacción del usuario con el sistemas y los casos de uso posibles.

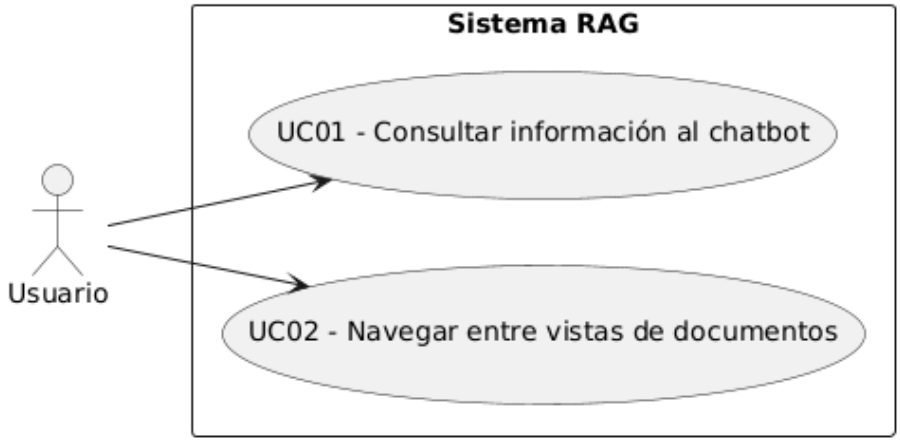

Figura 18: Diagrama de casos de uso
Fuente: Elaboración propia

10.4. Estructura de base de datos

La naturaleza del proyecto exige el uso de tecnologías especializadas. ChromaDB, por ejemplo, es una base de datos diseñada específicamente para almacenar y recuperar incrustaciones (embeddings) de manera eficiente, lo que la hace especialmente útil en proyectos de PLN y Machine Learning [52]. Aunque internamente utiliza SQLite, ChromaDB no funciona como una base de datos relacional tradicional, en lugar de manejar tablas y relaciones convencionales, está orientada a documentos y vectores, aprovechando la flexibilidad, velocidad y ligereza de SQLite para optimizar el manejo de embeddings.

Además de ChromaDB, el proyecto utiliza un archivo binario con extensión .pkl, que permite persistir estructuras de datos de Python, este archivo contiene un diccionario con listas, que contienen los chunks originales y su versión lematizada, los cuales son utilizados por el algoritmo BM25 para realizar búsquedas por coincidencia de palabras.

La *Figura 19* muestra cómo se estructuran y almacenan los datos tanto en ChromaDB como en el archivo binario, permitiendo una visión general del sistema de recuperación híbrido implementado.

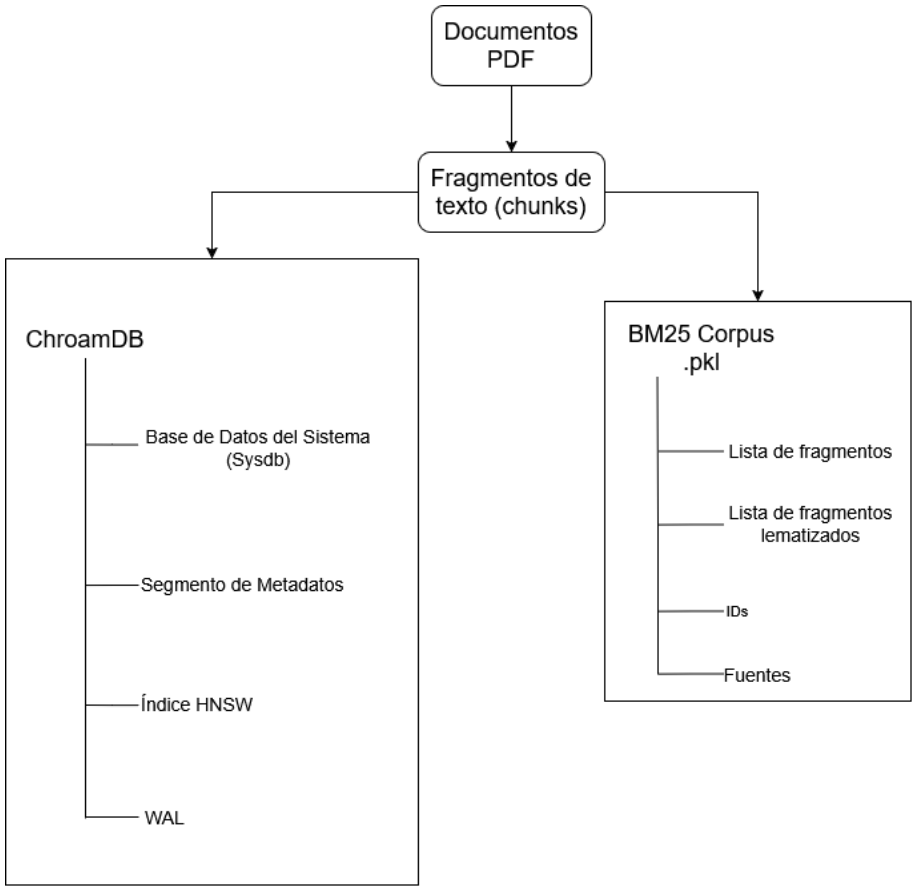

Figura 19: Diagrama de base de datos
Fuente: Elaboración propia

En la Figura anterior se muestra la estructura de la base de datos de Chroma de manera simplificada, ya que consiste de una gran variedad de archivos que convierten SQLite en una base para embeddings. El proyecto de software libre Chroma Cookbook muestra de manera detallada las partes que componen su estructura [53], como se ve a continuación:

Base de Datos del Sistema (Sysdb)

- `tenants` y `databases`: Organizan lógicamente los datos, permitiendo la separación de proyectos o entornos.
- `collections`: Agrupan incrustaciones, documentos y metadatos relacionados. Son como "tablas" en una base de datos tradicional.
- `collection_metadata`: Almacena la configuración y los parámetros de cada colección (por ejemplo, función de incrustación, métrica de distancia HNSW).
- `segments` y `segment_metadata`: Gestionan las estructuras de indexación subyacentes para cada colección, incluyendo los parámetros del índice HNSW.

- `migrations`: Registra las actualizaciones del esquema de la base de datos.
- `maintenance_log`: (Uso interno) Registra operaciones de mantenimiento.

Segmento de Metadatos (Donde residen los datos de las colecciones)

- `embeddings`: Contiene las incrustaciones vectoriales numéricas de tus documentos o fragmentos de texto. Es el corazón de la búsqueda semántica.
- `embedding_metadata`: Almacena metadatos adicionales (pares clave-valor) para *cada incrustación individual*, permitiendo un filtrado preciso.
- `embedding_fulltext_search` (Tabla Virtual FTS5): Permite la búsqueda de texto completo basada en palabras clave sobre el contenido de los documentos.

Índice HNSW (Archivos binarios externos)

- Archivos como `header.bin`, `data_level0.bin`, etc.: Almacenan la estructura de grafo jerárquico que permite búsquedas rápidas de similitud vectorial (Vecinos Más Cercanos Aproximados).

Write-Ahead Log (WAL)

- `embeddings_queue`: Registra todas las operaciones de ingesta de datos (añadir, actualizar, eliminar) para asegurar la durabilidad y consistencia de los datos.
- `embeddings_queue_config`: Configuración para la cola de incrustaciones, como la purga automática.
- `max_seq_id`: Rastrea el ID de secuencia máximo para la recuperación del WAL.

11. Pruebas

11.1. Pruebas de software

Para las pruebas de software no se emplearon pruebas de carga tradicionales, ya que la tecnología de WebSockets gestionados por Streamlit, no expone endpoints REST convencionales sobre los cuales puedan realizarse peticiones automatizadas para pruebas, a diferencia de una API tradicional basada en HTTP request, donde herramientas como Postman pueden generar tráfico simulado con facilidad, el modelo de comunicación de Streamlit se basa en un canal WebSocket persistente y administrado internamente.

Aunque existen herramientas como Locust que permiten realizar pruebas de carga sobre conexiones WebSocket, se presentan varios inconvenientes, en primer lugar, es necesario identificar el formato de los mensajes y los eventos que deben ser enviados a través del canal para activar los flujos correctos dentro de la aplicación, lo implica comprender cómo se estructura la información y qué eventos deben ser disparados para simular interacciones reales de los usuarios. En el caso de Streamlit, esta tarea se complica debido a que la biblioteca orquesta automáticamente los eventos, genera nombres internos automáticamente y emplea identificadores genéricos para estos eventos, implicando tener que rastrear con exactitud los puntos del flujo de datos que deben ser usados y hacer una prueba de carga grandes conocimientos de Web Scraping y un análisis detallado del tráfico WebSocket en tiempo real. Este nivel de complejidad vuelve difícil y poco práctico el diseño de una prueba de carga sin una comprensión profunda del funcionamiento interno del framework.

Por esta razón, se optó por realizar pruebas de tipo End-to-End o E2E, las cuales permiten simular la interacción completa del usuario con el sistema, evaluando así el funcionamiento real de la aplicación desde la perspectiva del usuario final [54].

Por este motivo, se desarrollaron scripts en JavaScript utilizando Node.js y la librería Puppeteer, que permite automatizar acciones en instancias de Chrome o Firefox, el codigo de estas pruebas se puede encontrar en el anexo 4. Estas pruebas simulan el uso del chatbot, midiendo el tiempo de respuesta entre el envío del prompt y la recepción completa de la response del chatbot. Para mayor realismo, se creó una lista de preguntas de diferentes tamaños, representando consultas de distintos usuarios.

Las pruebas se organizaron en dos categorías:

- **Prueba por lotes**: Se simula una cantidad específica de usuarios que acceden al sistema en grupos definidos, evaluando cómo se comporta la aplicación al procesar varios usuarios simultáneamente.
- **Prueba de rendimiento tipo throughput**: Se mide la capacidad de la aplicación para atender usuarios de forma continua, lanzando nuevos usuarios *cada ciertos segundos* y registrando cuántas interacciones puede gestionar por minuto sin degradar el rendimiento.

Cabe recalcar que al usar el API de OpenAI las pruebas dependen mucho de la conexión

de internet del servidor donde corre la aplicación, lo que puede generar variaciones en los datos, viendo tiempos más elevados de lo normal.

11.1.1. Prueba por lotes

En las mediciones de tiempo de respuesta del chatbot, se crearon 4 clases para agrupar los tiempos y que fuera más fácil de analizar los resultados, [0, 3], (3, 7], (7, 10] y (10, $+\infty$), se deja como último los tiempos de 10 segundos o más puesto que se consideró que desde ese punto ya se considera una respuesta muy lenta.

Para iniciar, se simularon 10 usuarios en lotes de 1, osea, simulando un usuario cada vez que otro termina, teniendo un tiempo mínimo de **3,76 segundos**, un tiempo máximo de **9,29 segundos** y un promedio de **5,81 segundos**, en la *Figura 20* se muestra la cantidad de tiempos de cada clase.

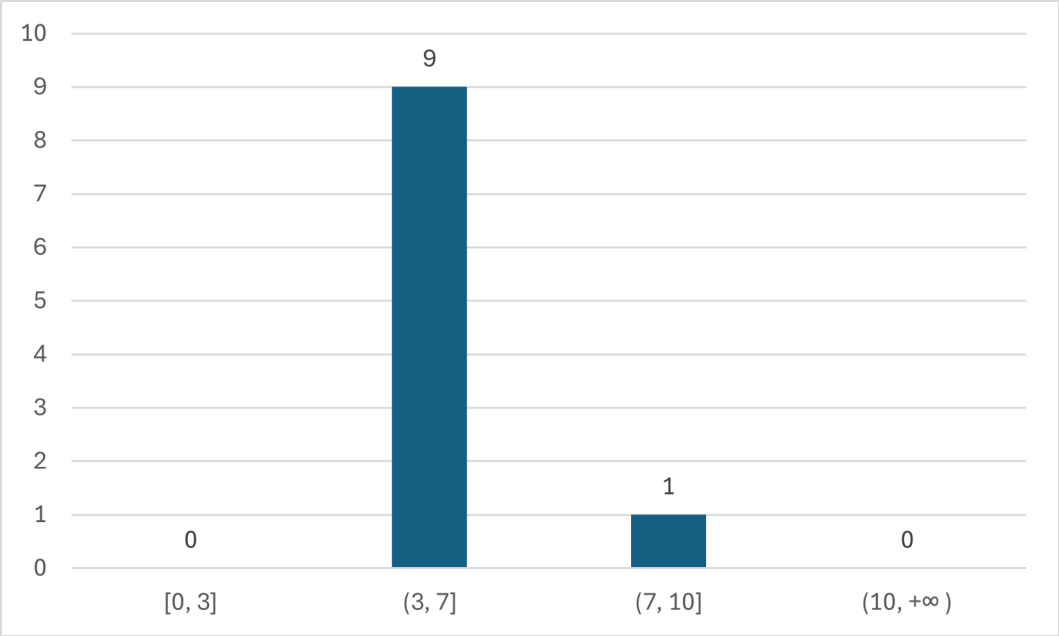

Figura 20: Pruebas por lotes 1
Fuente: Elaboración propia

Después, se simularon 20 usuarios en lotes de 2, teniendo un tiempo mínimo de **3,22 segundos**, un tiempo máximo de **18,90 segundos** y un promedio de **7,68 segundos**, en la *Figura 21* se muestra la cantidad de tiempos de cada clase.

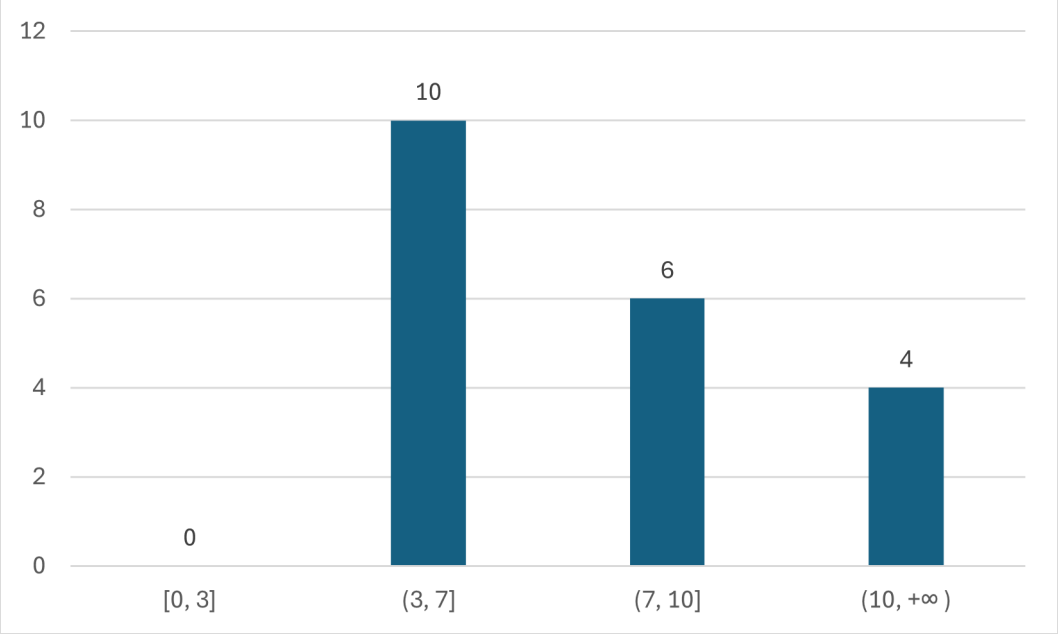

Figura 21: Pruebas por lotes 2
Fuente: Elaboración propia

Con 50 usuarios en lotes de 5, se obtuvo un tiempo mínimo de **3,46 segundos**, un tiempo máximo de **15,05 segundos** y un promedio de **8,62 segundos**, en la *Figura 22* se muestra la cantidad de tiempos de cada clase.

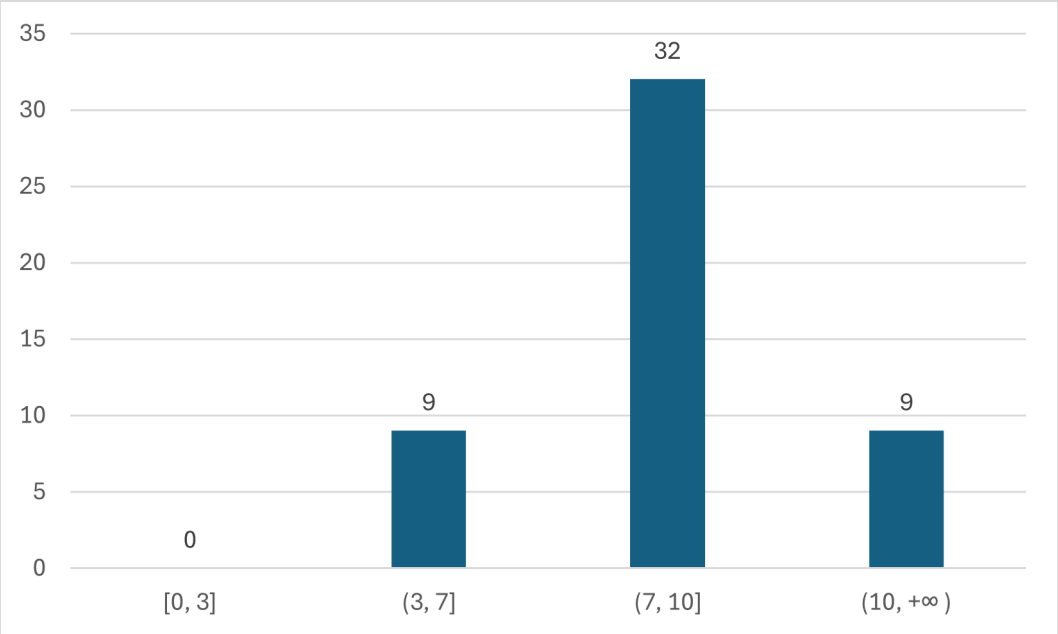

Figura 22: Pruebas por lotes 3
Fuente: Elaboración propia

Con 100 usuarios en lotes de 10, se obtuvo un tiempo mínimo de **5,26 segundos**, un tiempo máximo de **19,78 segundos** y un promedio de **12,54 segundos**, en la *Figura 23* se muestra la cantidad de tiempos de cada clase.

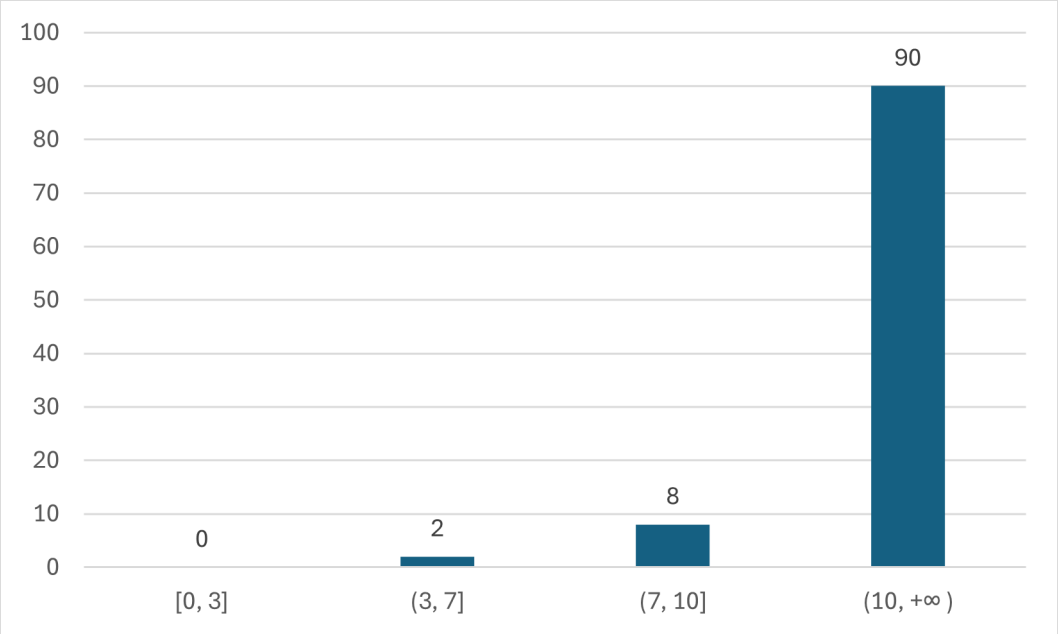

Figura 23: Pruebas por lotes 4
Fuente: Elaboración propia

Por último, con 200 usuarios en lotes de 20, se llegó al punto donde las pruebas generaban errores, puesto que se estaban esperando tiempos demasiado prolongados, fallando en el **96 %** de los casos, teniendo un tiempo mínimo de las instancias que no fallaron de **21,09 segundos**, un tiempo máximo de **31,23 segundos** y un promedio de **25,82 segundos**. En la *Figura 24* se muestra la clasificación de las instancias que no fallaron, y la *Figura 25* ilustra la cantidad de instancias que no fallaron y las que sí.

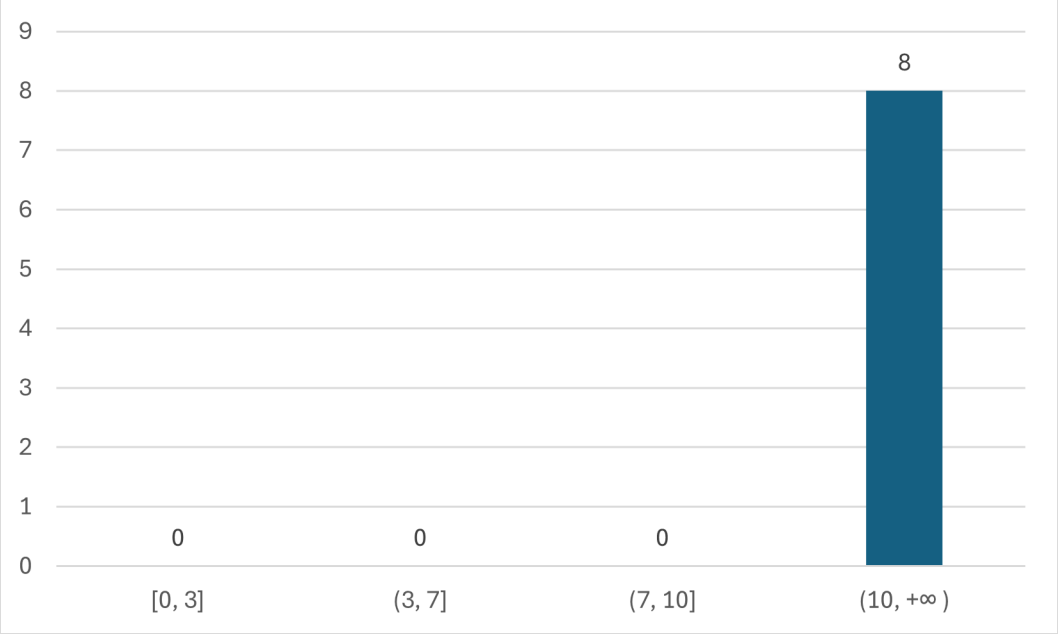

Figura 24: Pruebas por lotes 5
Fuente: Elaboración propia

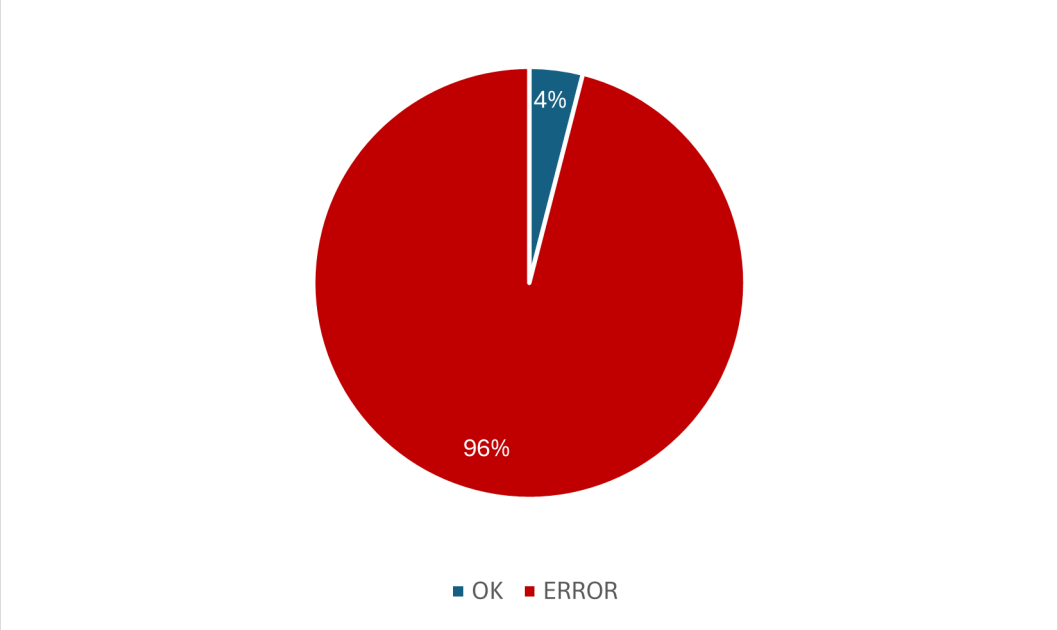

Figura 25: Pruebas por lotes 5
Fuente: Elaboración propia

Los resultados muestran una tendencia clara, a medida que se incrementa la cantidad de usuarios que interactúan simultáneamente con la aplicación, el tiempo promedio de respuesta también aumenta. Esto no se debe a un límite en la cantidad de tokens procesados por minuto, sino a las restricciones en la cantidad de solicitudes que se pueden

realizar. La API utilizada tiene límites para cuentas personales, que son más bajos que los ofrecidos a organizaciones, esto significa que, aunque no se están agotando los recursos de tokens, sí se alcanzan los límites de frecuencia de uso. Cuando se llega a ese límite, las solicitudes se van dejando en cola y, a medida que se liberan los recursos, se atienden las peticiones de la cola, lo que contribuye al incremento de los tiempos de respuesta. Aun así, estos resultados demuestran que el sistema puede manejar una cantidad considerable de usuarios antes de llegar a su límite de capacidad. En la *Tabla 14* se muestran los límites del modelo GPT-4.1-mini directamente de la página de OpenAI [41].

Tabla 14: Límites por nivel (Tier) del API de OpenAI

TIER	RPM	RPD	TPM	BATCH QUEUE LIMIT
Free	3	200	40,000	–
Tier 1	500	10,000	200,000	2,000,000
Tier 2	5,000	–	2,000,000	20,000,000
Tier 3	5,000	–	4,000,000	40,000,000
Tier 4	10,000	–	10,000,000	1,000,000,000
Tier 5	30,000	–	150,000,000	15,000,000,000

Fuente: Tomado de [41]

11.1.2. Prueba de throughput

Para mostrar de otra manera las capacidades tanto de la aplicación como del API de OpenAI, se diseñó una prueba enfocada en medir cuántos usuarios por minuto puede atender el sistema y cómo varían los tiempos de respuesta a medida que aumenta la cantidad de usuarios, esta prueba busca simular un uso más realista de la aplicación, mientras algunos usuarios apenas envían su pregunta al chatbot, otros ya están recibiendo una respuesta. Así se representa un escenario más cercano al uso cotidiano, en el que las interacciones no ocurren todas al mismo tiempo, sino de forma continua y escalonada.

Todos los casos de esta prueba se realizaron con 50 usuarios, en parte para optimizar el tiempo de ejecución, además, dado que en este tipo de prueba no se incrementan los lotes de usuarios, no es necesario aumentar su cantidad para obtener datos suficientes y representativos.

Para iniciar, se realizó la prueba lanzando un usuario cada 15 segundos, obteniendo como resultado de todas las ejecuciones un tiempo mínimo de **3,81 segundos**, un tiempo máximo de **16,20 segundos** y un tiempo promedio de **7,26 segundos**. En la *Figura 26* se ve la distribución de los datos en las clases.

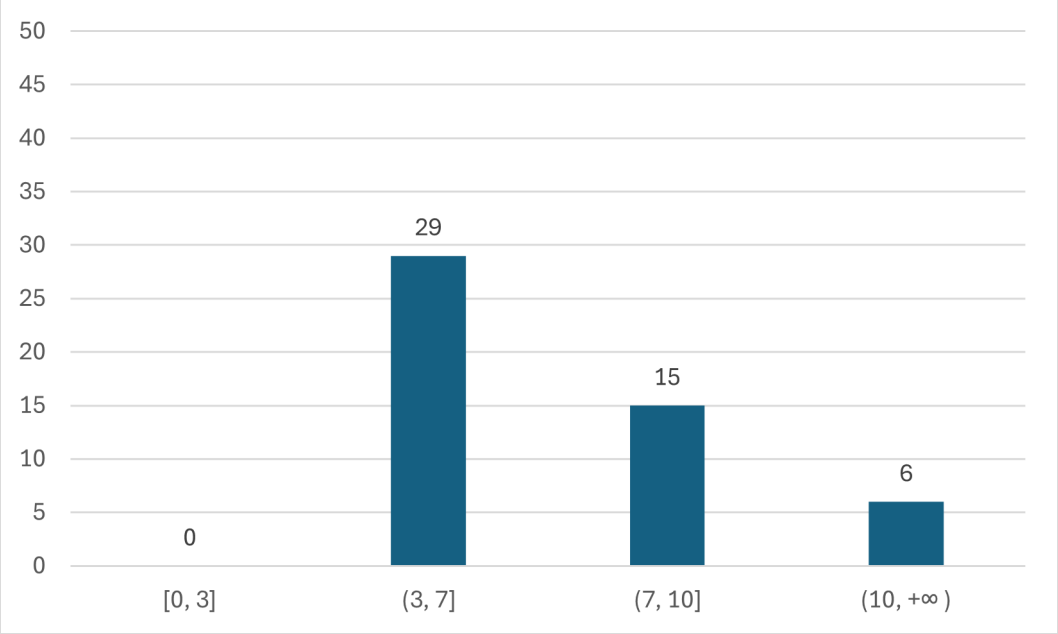

Figura 26: Pruebas throughput 1
Fuente: Elaboración propia

Siguiendo con la prueba, se bajó la cantidad a un usuario cada 10 segundos, dando como resultado un tiempo mínimo de **3,1 segundos**, como tiempo máximo **12,37 segundos** y un tiempo promedio de **6,77 segundos**. En la *Figura 27* se ve la distribución de los datos en las clases.

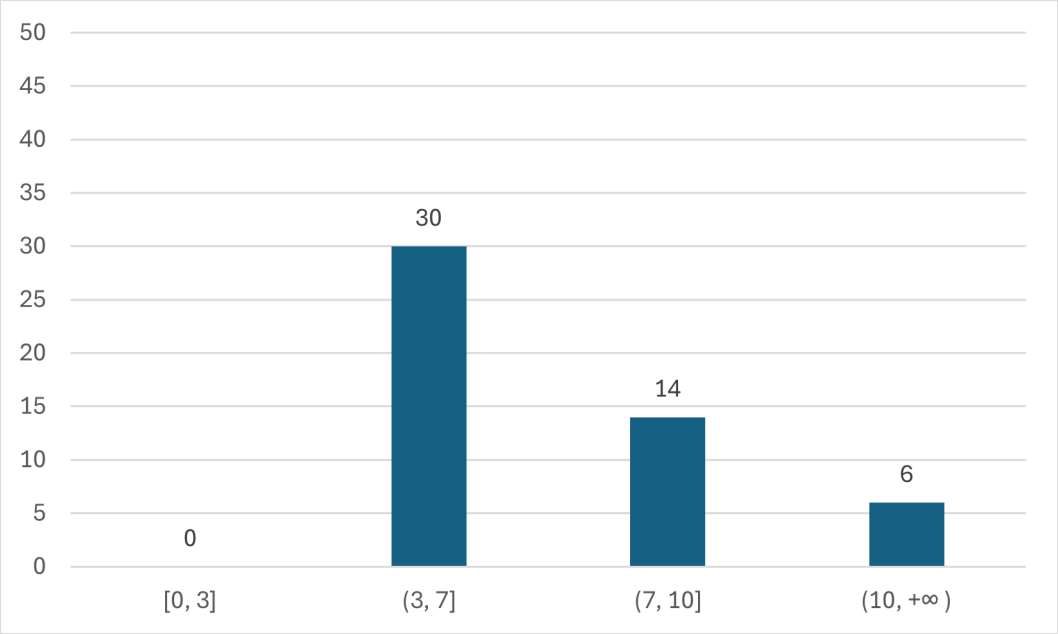

Figura 27: Pruebas throughput 2
Fuente: Elaboración propia

De la misma manera se bajó la cantidad a un usuario cada 5 segundos, obteniendo un tiempo mínimo de **3,72 segundos**, un tiempo máximo de **12,75 segundos** y un tiempo promedio de **7,96 segundos**. En la *Figura 28* se ve la distribución de los datos en las clases.

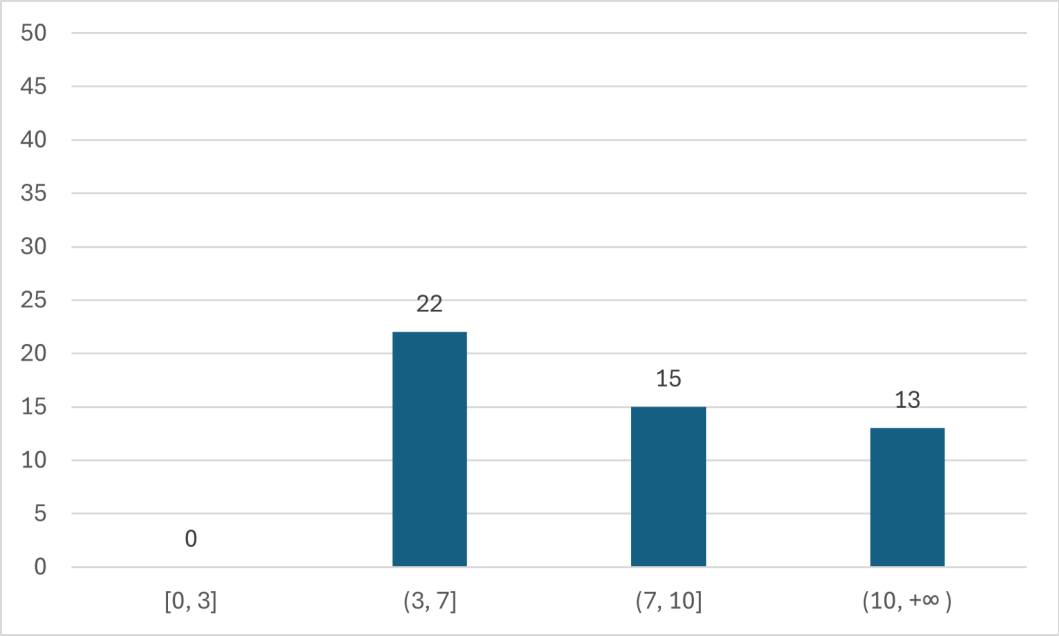

Figura 28: Pruebas throughput 3
Fuente: Elaboración propia

Luego se redujo la cantidad a un usuario cada 2 segundos, dejando como resultado un tiempo mínimo de **2,9 segundos**, un tiempo máximo de **12,29 segundos** y un tiempo promedio de **6,78 segundos**. En la *Figura 29* se ve la distribución de los datos en las clases.

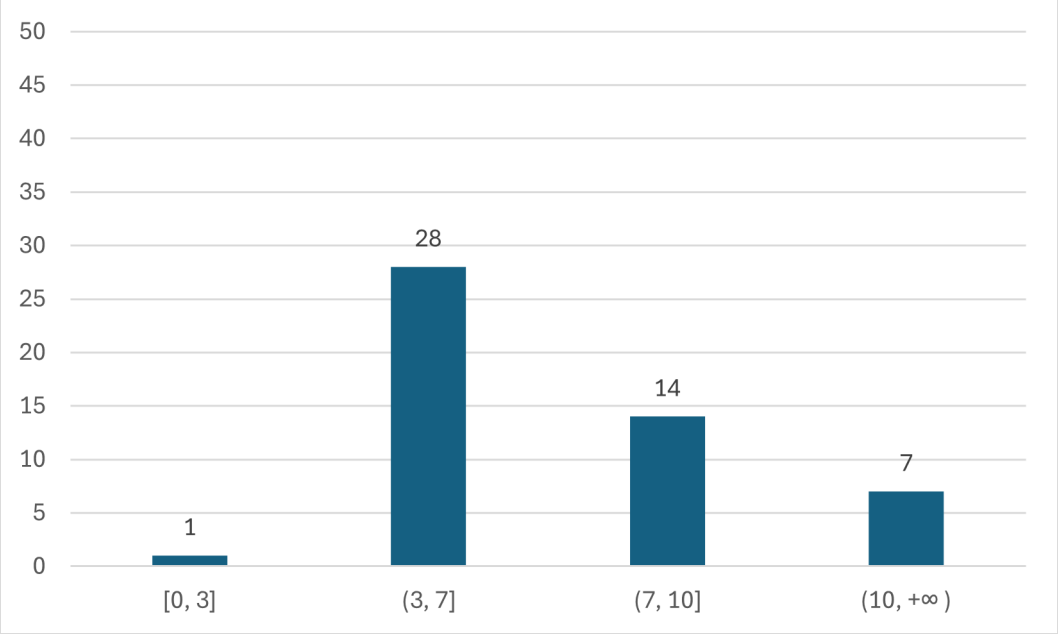

Figura 29: Pruebas throughput 4
Fuente: Elaboración propia

Después se redujo la cantidad a un usuario cada 1 segundos, dejando como resultado un tiempo mínimo de **3,09 segundos**, un tiempo máximo de **12,41 segundos** y un tiempo promedio de **6,32 segundos**. En la *Figura 30* se ve la distribución de los datos en las clases.

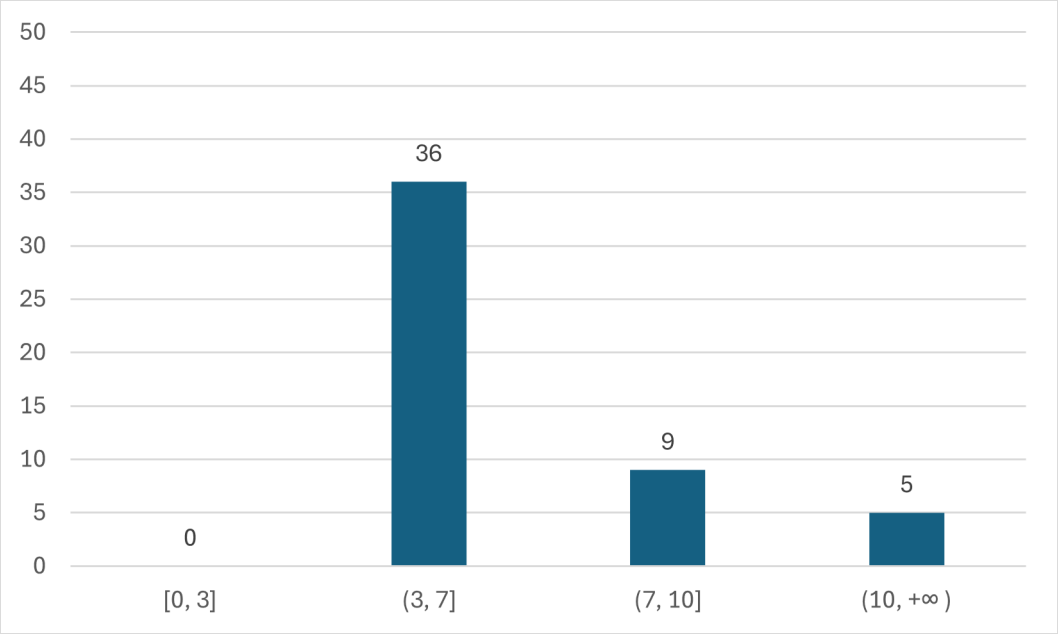

Figura 30: Pruebas throughput 5
Fuente: Elaboración propia

Por último, se redujo la cantidad a un usuario cada 500 milisegundos, o dos usuarios por segundo, dejando como resultado un tiempo mínimo de **2,66 segundos**, un tiempo máximo de **12,18 segundos** y un tiempo promedio de **6,31 segundos**. En la *Figura 31* se ve la distribución de los datos en las clases.

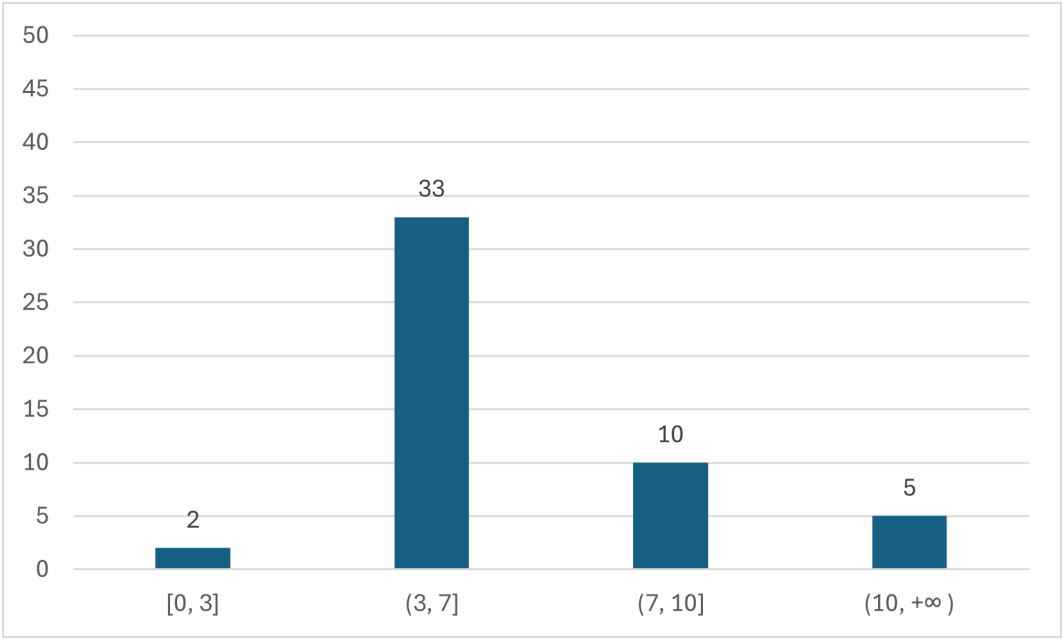

Figura 31: Pruebas throughput 6
Fuente: Elaboración propia

Estas pruebas resultan esenciales para demostrar la robustez y capacidad de escalado de la ampliación, evidenciando que el sistema es capaz de atender un gran número de usuarios de manera eficiente y progresiva, sin que todas las interacciones ocurran de manera simultánea, lo que es muy importante en el entorno académico, donde la demanda de atención varia de forma considerable en periodos como matriculas y adiciones y cancelaciones.

Los resultados de las pruebas fueron muy relevantes, al simular la interacción de dos usuarios por segundo, el chatbot demostró una capacidad teórica de gestionar hasta 120 usuarios por minuto sin presentar errores, lo cual es una cifra de atención por encima de lo que podría alcanzar el personal humano, para quienes seria imposible llegar a este número.

Al automatizar esta tarea, el sistemas no solamente incrementa la eficiencia en la atención de los estudiantes, sino que también libera carga al personal administrativo, dándoles la posibilidad de usar este tiempo en tareas mas complejas, urgentes o que generan mayor valor, siendo esta capacidad de escalar en gran medida la atención sin comprometer calidad de servicio, la mayor virtud de la aplicación, con el valor añadido de estar disponible 24 horas al día, optimizando el uso del recurso humano.

11.2. Comparación con un sistema similar

Aunque al investigar se pueden encontrar varios trabajos que hablan sobre chatbots de inteligencia artificial para la asistencia en algún área en instituciones de educación superior, un gran número de estos son propuestas que no han llegado a la etapa de desarrollo, y de los trabajos que sí tuvieron un desarrollo, son chatbots que no están desplegados para su uso o son de uso exclusivo como el chatbot de la Universidad Técnica Particular de la Loja [20], por lo que generar comparaciones entre el asistente de este proyecto y otros se vuelve una tarea compleja.

Dadas estas complicaciones se investigaron chatbots tipo RAG que estuvieran disponibles para uso público, encontrando que actualmente no existen prácticamente chatbots que usen esta tecnología o que abiertamente especifiquen que fueron hechos de esta manera, se puede suponer que los chatbots que permiten interactuar con archivos como PDFs usan en su mayoría tecnologías RAG, pero sería imprudente realizar esta afirmación.

Debido a la complicación de conseguir asistentes creados con el mismo fin, se busca comparar con chatbots creados no precisamente con el mismo fin, pero sí con técnicas de recuperación de información. Grandes proveedores cuentan con servicios para crear, almacenar y levantar chatbots hechos con RAG, así como IBM Watson X, Amazon Bedrock o los servicios de IA de Azure, pero estos son servicios que brindan las herramientas para crear los asistentes, mas no tienen chatbots creados con los que se pueda interactuar. Sin embargo, MongoDB cuenta con una herramienta llamada Chatbot Demo Builder [55], que permite crear chatbots RAG en segundos y ajustar algunos parámetros como el tamaño de los chunks y el solapamiento entre estos.

Para comparar el tiempo de respuesta de esta herramienta, al igual que con el asistente de este proyecto, se probó con diferentes preguntas con longitudes diferentes para simular el uso real de un usuario. Se debe mencionar que en la prueba se usó un solo archivo, que fue el Acuerdo 009, ya que la herramienta limita a un solo documento, y este es el más extenso de ellos. Se realizaron 20 consultas al asistente, dando como resultado un tiempo mínimo de **4.02 segundos**, un tiempo máximo de **11 segundos** y un tiempo promedio de **6.56 segundos**, en la *Figura 32* se muestra cómo se distribuyeron los tiempos en las clases.

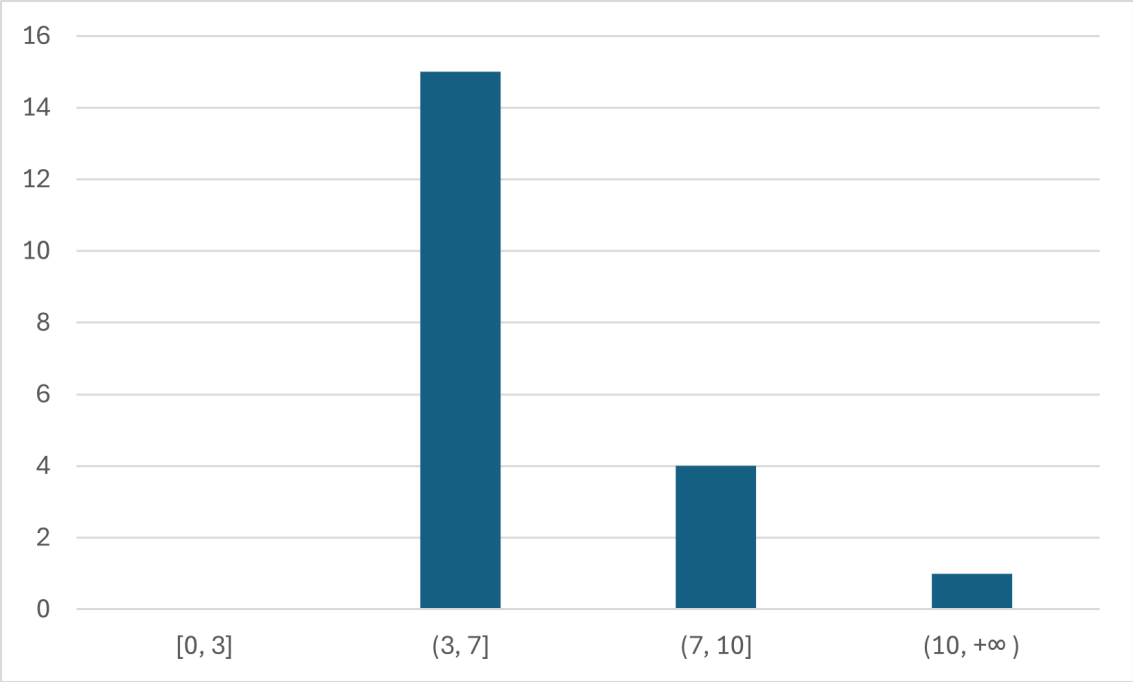

Figura 32: Resultados tiempo de respuesta MongoDB Demo Builder
Fuente: Elaboración propia

Teniendo en cuenta todos los resultados obtenidos en la prueba de throughput, tenemos que la herramienta del proyecto tuvo un tiempo mínimo de **2.66 segundos**, un tiempo máximo de **16.2** segundos y un tiempo promedio de 6.9 segundos. Se puede notar claramente que el chatbot del proyecto puede llegar a dar respuesta en un menor tiempo que la herramienta de MongoDB, pero así mismo, en las preguntas más elaboradas, puede tener tiempos de respuesta de hasta **5 segundos** más que Chatbot Demo Builder, pero en general, la experiencia del usuario debe ser relativamente similar, pues el tiempo promedio de las dos herramientas solo difiere en **0.34 segundos**, lo que demuestra que el chatbot para la oficina de Ingeniería de Sistemas, en cuanto a tiempos de respuestas para la experiencia del usuario es competente y está a la altura de soluciones similares.

Claramente, el chatbot de la Universidad tiene oportunidades de mejora que se pueden trabajar para mejorar tanto el rendimiento como la experiencia del usuario. Se cree que una posible forma de dar una mejora sustancial al rendimiento es implementar concurrencia en la aplicación, puesto que al ser un chatbot híbrido se realizan dos procesos de búsqueda, primero la búsqueda por embeddings y luego por el algoritmo BM25, que se hace secuencialmente, haciendo que se ejecuten al mismo tiempo en hilos diferentes se podría conseguir un aumento de velocidad en la respuesta de la aplicación.

También se debe reconocer que el proyecto tiene puntos fuertes y débiles con respecto a lo que ofrecen empresas para estas soluciones, entre sus puntos fuertes está la flexibilidad, puesto que se puede moldear el proyecto de muchas maneras casi sin limitaciones, ajustando y cambiando tanto parámetros como tecnologías, se puede cambiar de LLM, proveedor, modelo de embeddings, e incluso de métodos de recuperación, además de prácticamente ninguna otra solución permite crear chatbots RAG híbridos, pero, se tiene como contra

que se deben tener conocimientos para realizar estos cambios, mientras que los proveedores facilitan la creación y puesta en marcha del chatbot sin necesidad de crear código.

11.3. Pruebas de usabilidad

Las pruebas de usabilidad se dividieron en dos enfoques complementarios: por un lado, se aplicó una encuesta con escala de Likert a estudiantes, quienes representan a los usuarios finales del sistema; por otro, se realizó una evaluación experta con ingenieros en sistemas, basada en las heurísticas de Jakob Nielsen. Esta división permitió obtener una perspectiva equilibrada entre la experiencia real de uso y el análisis técnico especializado, asegurando una evaluación más completa.

11.3.1. Prueba con estudiantes

La encuesta de usabilidad para los estudiantes se dividió en dos secciones, en la primera sección donde se pregunta de la interacción de los estudiantes con herramientas de IA y la segunda donde se pregunta acerca de la aplicación en cuestión.

En la primera sección se deseaba conocer en qué medida los estudiantes usan en su día a día herramientas de chatbot o asistentes virtuales, el **24,5 %** de los estudiantes interactúan con estas herramientas frecuentemente y el **41,5 %** lo hace muy frecuentemente, como se ve en la *Figura 33* , lo que significa que tienen experiencia usando herramientas similares. Además, el **92,5 %** indica no conocer una herramienta como un chatbot que brinde información acerca del programa o la universidad, como se aprecia en la *Figura 34* .

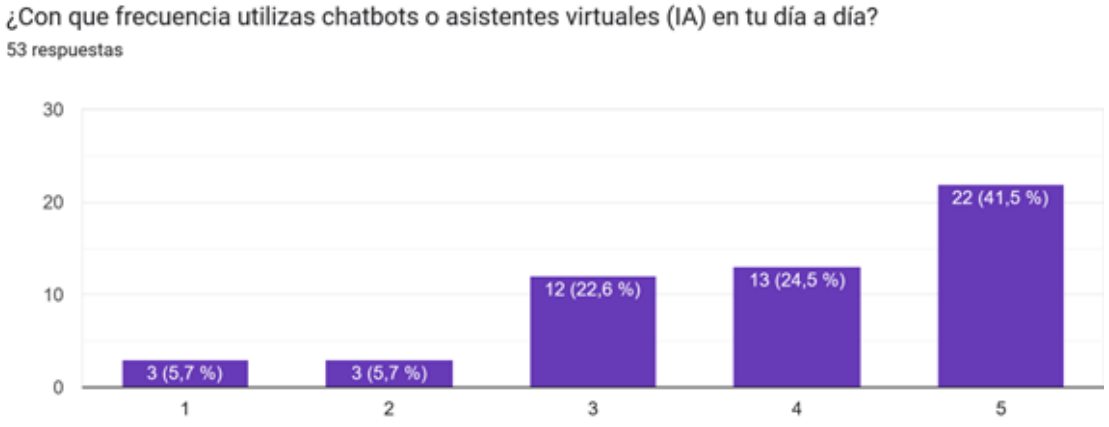

Figura 33: Resultados prueba usabilidad 1
Fuente: Elaboración propia

Figura 34: Resultados prueba usabilidad 2
Fuente: Elaboración propia

En la segunda sección, se proporcionaron a los estudiantes una serie de instrucciones para que pudieran explorar de forma guiada las principales funciones de la aplicación. Se les pidió realizar una consulta general al chatbot, hacer preguntas relacionadas con secciones específicas de un documento, formular una pregunta sobre un tema que no estuviera dentro del alcance del sistema y, finalmente, recorrer la interfaz y explorar las diferentes vistas disponibles en la aplicación.

Al ver las respuestas, se encuentra que el **43,4 %** de los estudiantes encuentra fácil obtener información a través del chatbot y **45,3 %** lo encuentra muy fácil, lo que demuestra que es una fuente de información eficaz e intuitiva, esto se puede apreciar en la *Figura 35* .

Figura 35: Resultados prueba usabilidad 3
Fuente: Elaboración propia

También el **43,4 %** de los encuestados consideran que la fuente de la información de donde el chatbot se basa, y un **47,2 %** piensa que es muy clara, significando que el asistente presenta de manera precisa los documentos de origen de los datos. Esto se puede ver en la *Figura 36* .

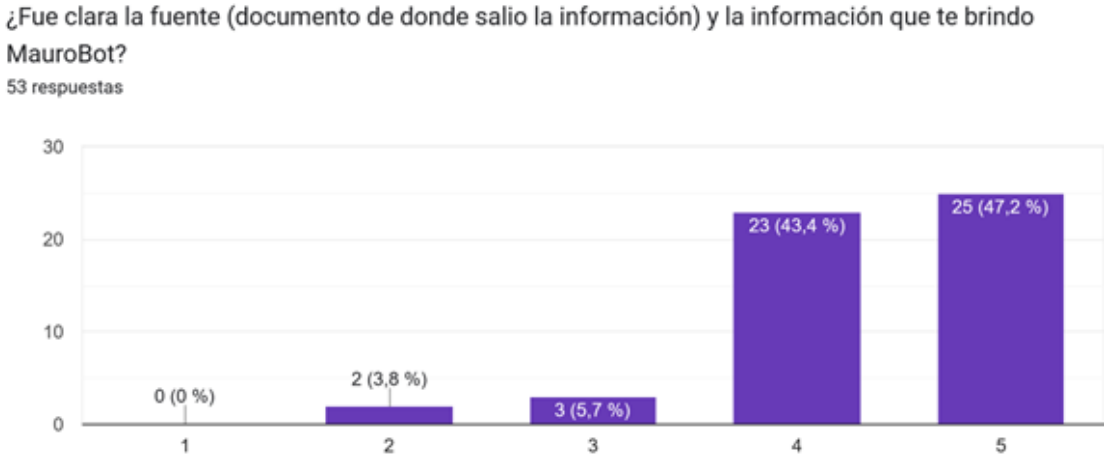

Figura 36: Resultados prueba usabilidad 4
Fuente: Elaboración propia

Del mismo modo, se consultó a los estudiantes sobre un aspecto clave del chatbot, su capacidad para mantenerse dentro del dominio de temas definido. Aunque el modelo de lenguaje ha sido entrenado con una amplia variedad de contenidos y es capaz de hablar

sobre muchos temas, en este caso su función debe centrarse exclusivamente en la Universidad del Valle y el Programa de Ingeniería de Sistemas. Según los resultados, en el **86,8 %** de las interacciones el chatbot evitó referirse a otros temas, mientras que en un **13,2 %** proporcionó información fuera del ámbito académico, como se ve en la *Figura 37* . Estos datos muestran que en la mayoría de los casos el asistente logra mantenerse dentro de los límites establecidos, sin embargo, delimitar el comportamiento de un modelo de este tipo es un desafío, ya que ciertas formas de realizar prompts pueden hacer que responda sobre otros temas, por esta razón sería útil investigar mecanismos que refuercen aún más esa restricción.

Figura 37: Resultados prueba usabilidad 5
Fuente: Elaboración propia

Al centrarse en la interfaz de la aplicación, los resultados de la encuesta reflejan una percepción muy positiva por parte de los estudiantes, el **30,2 %** de los encuestados consideró que la interfaz es intuitiva, y un **50,9 %** opinó que muy intuitiva, como se muestra en la *Figura 38* . En cuanto a la facilidad de navegación, un **22,6 %** indicó que es fácil moverse por la aplicación, mientras que un **66 %** la calificó como muy fácil de utilizar, según la *Figura 39* . Estos resultados confirman que el diseño inspirado en las interfaces de los chatbots de inteligencia artificial actuales logra su objetivo de ofrecer una experiencia familiar, clara y cómoda, sin necesitar que los usuarios aprendan a usar la herramienta desde cero.

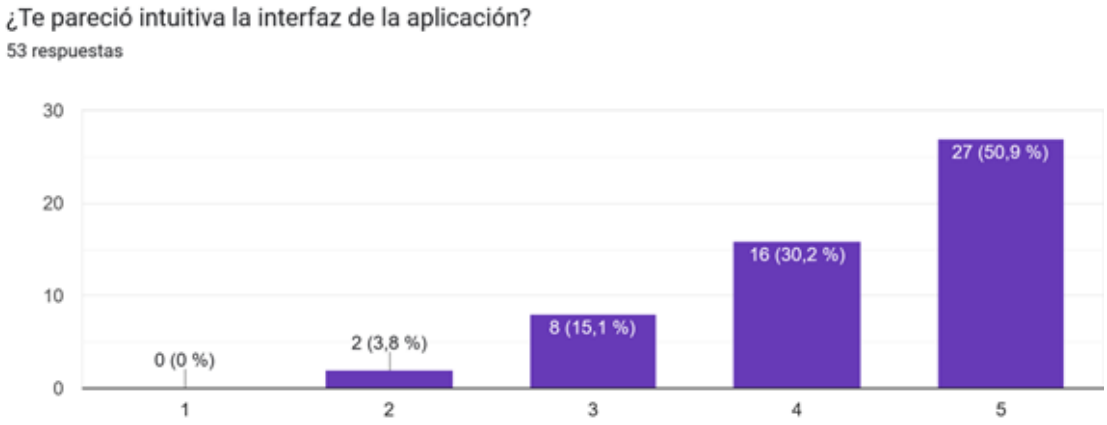

Figura 38: Resultados prueba usabilidad 6
Fuente: Elaboración propia

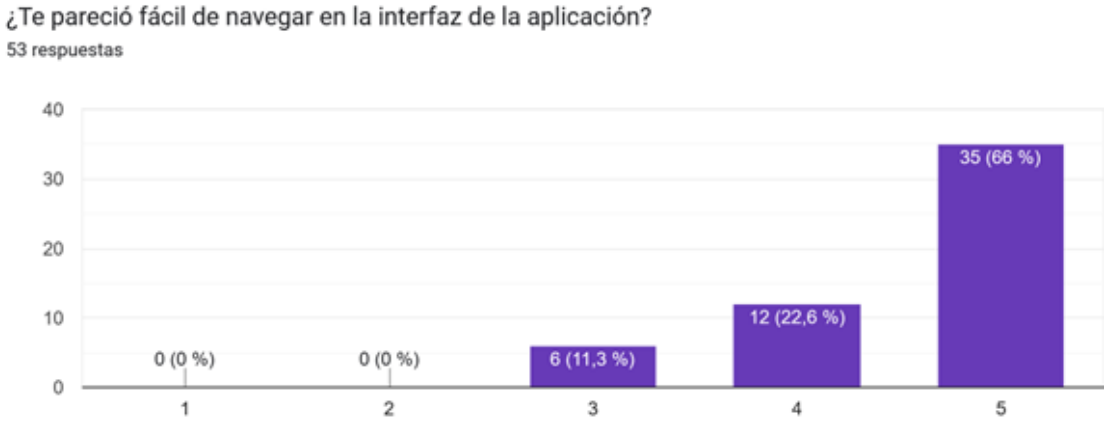

Figura 39: Resultados prueba usabilidad 7
Fuente: Elaboración propia

11.3.2. Evaluación experta basada en las heurísticas de Nielsen

Las heurísticas de Jakob Nielsen son diez principios generales de usabilidad que orientan el diseño de interfaces eficaces, no se trata de reglas estrictas, sino de recomendaciones ampliamente aceptadas que ayudan a identificar posibles mejoras en la experiencia del usuario, estas diez heurísticas son [56]:

1. Visibilidad del estado del sistema

2. Correspondencia entre el sistema y el mundo real
3. Control y libertad del usuario
4. Consistencia y estándares
5. Prevención de errores
6. Reconocimiento en lugar de recuerdo
7. Flexibilidad y eficiencia de uso
8. Diseño estético y minimalista
9. Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores
10. Ayuda y documentación

En cuanto a la visibilidad del estado del sistema, los resultados fueron muy positivos, el **88,9 %** de los expertos indicó estar completamente de acuerdo con que este principio se cumple en la aplicación, como se ve en la *Figura 40* , este resultado refleja que el sistema comunica adecuadamente su estado, ya que el chatbot informa de forma clara cuando está procesando la consulta del usuario y buscando la información necesaria para generar la respuesta.

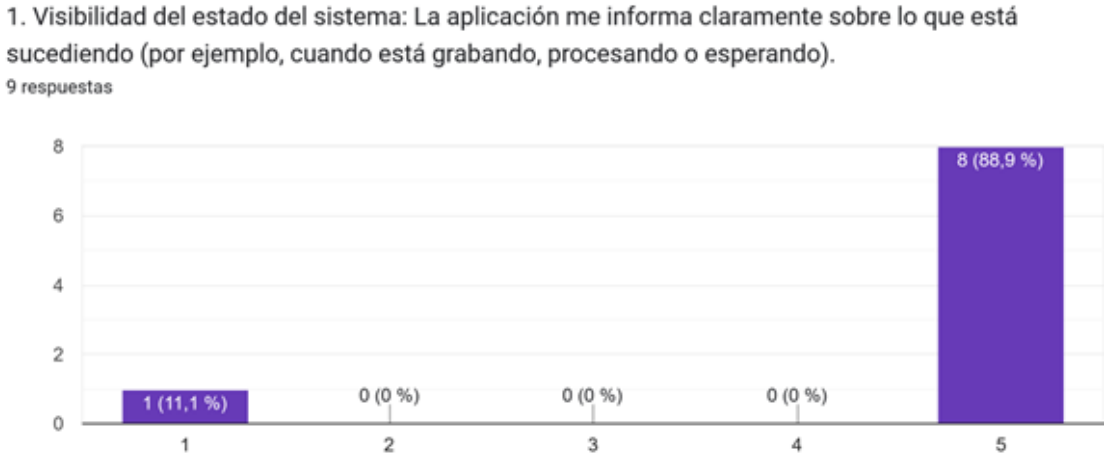

Figura 40: Resultados prueba Jakob Nielsen Visibilidad del estado del sistema
Fuente: Elaboración propia

En lo relacionado con la correspondencia entre el sistema y el mundo real, los resultados reflejan una valoración positiva por parte de los expertos, el **55,6 %** indicó estar

completamente de acuerdo con que el lenguaje y los íconos utilizados son fáciles de entender y les resultan familiares, mientras que un **33,3 %** manifestó estar de acuerdo. Solo un 11,1 % mostró una postura neutral y no se presentaron respuestas en desacuerdo. Estos resultados indican que la aplicación logra comunicar sus funciones de manera clara, utilizando términos y elementos visuales que los usuarios reconocen con facilidad, lo cual contribuye a una experiencia intuitiva.

Figura 41: Resultados prueba Jakob Nielsen Correspondencia entre el sistema y el mundo real

Fuente: Elaboración propia

En cuanto a la heurística de control y libertad del usuario, los resultados muestran una percepción mayormente positiva, aunque con algunas observaciones, el **44,4 %** de los encuestados dijo estar completamente de acuerdo en que pueden deshacer o cancelar acciones con facilidad si cometen un error, mientras que un **33,3 %** expresó estar de acuerdo, sin embargo, un **22,2 %** manifestó desacuerdo en distintos niveles. Estos números sugieren que, si bien la mayoría considera que la aplicación brinda cierto nivel de control, aún hay oportunidades de mejora para proporcionar opciones más claras que permitan corregir acciones, como por ejemplo, añadir un botón para detener la response del chatbot si el usuario se equivocó en su prompt.

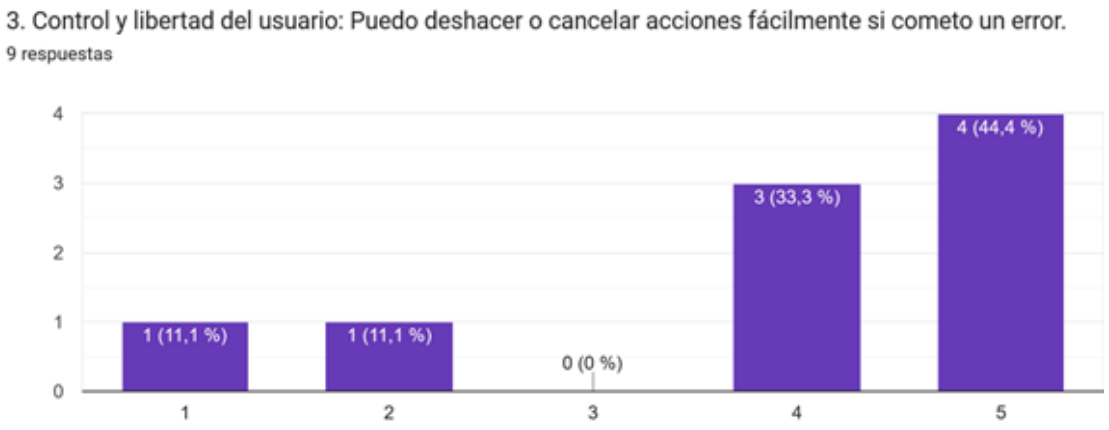

Figura 42: Resultados prueba Jakob Nielsen Control y libertad del usuario
Fuente: Elaboración propia

Respecto a la heurística de consistencia y estándares, los resultados fueron igualmente muy positivos, un **66,7 %** de los encuestados indicó estar completamente de acuerdo con que los elementos de la interfaz como botones, menús y mensajes, son consistentes a lo largo de toda la aplicación, mientras que un **22,2 %** manifestó estar de acuerdo. Esta valoración indica que el diseño tiene una coherencia visual y funcional, lo cual contribuye a la experiencia del usuario, reforzando la decisión de hacer la interfaz inspirada en los chatbots actuales, lo que aumenta la familiaridad y reduciendo la curva de aprendizaje.

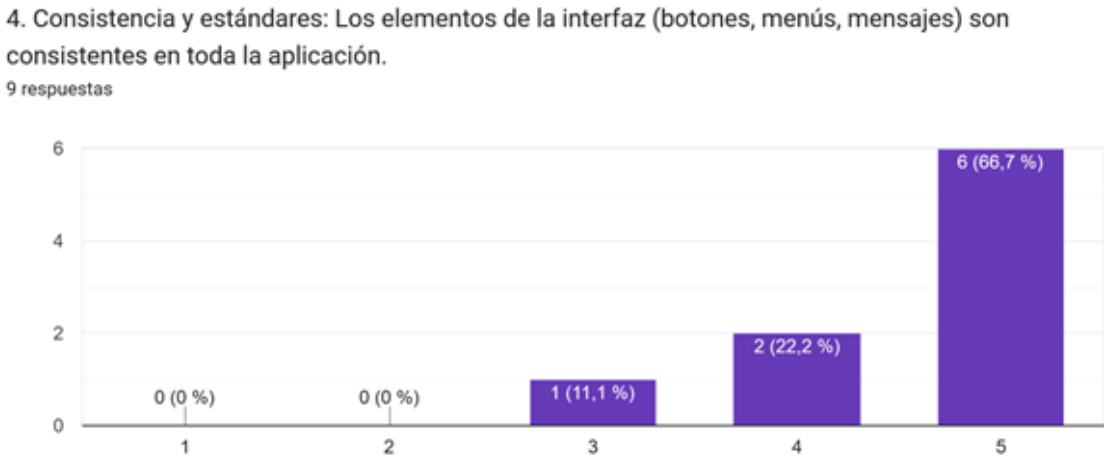

Figura 43: Resultados prueba Jakob Nielsen Consistencia y estándares
Fuente: Elaboración propia

En cuanto a la prevención de errores, los resultados muestran variaciones en las respuestas. El **33,3 %** de los evaluadores indicó estar totalmente de acuerdo con que la aplicación ayuda a evitar errores y advierte antes de ejecutar acciones importantes, mientras que otro **33,3 %** expresó una valoración neutral. Además, un **22,2 %** manifestó estar de acuerdo y un **11,1 %** mostró desacuerdo. Esta variabilidad se debe a que el chatbot no dispone de muchas acciones que puedan ser sensibles o de importancia, más allá de la interacción con al asistente, por lo que no hay que advertir de al usuario de antes de realizar una acción.

Figura 44: Resultados prueba Jakob Nielsen Prevención de errores
Fuente: Elaboración propia

En cuanto a la heurística de reconocer antes que recordar, los resultados fueron positivos, el **66,7 %** de los evaluadores indicó estar totalmente de acuerdo en que la información y las opciones importantes están siempre visibles o son fácilmente accesibles, mientras que el **33,3 %** expresó estar de acuerdo. Ningún evaluador dio una calificación neutral o negativa, lo que indica que la aplicación presenta de forma efectiva los elementos clave para la interacción, sin hacer que el usuario tenga que recordar pasos anteriores o extensos, lo cual mejora la fluidez en el uso y reduce al mínimo la curva de aprendizaje.

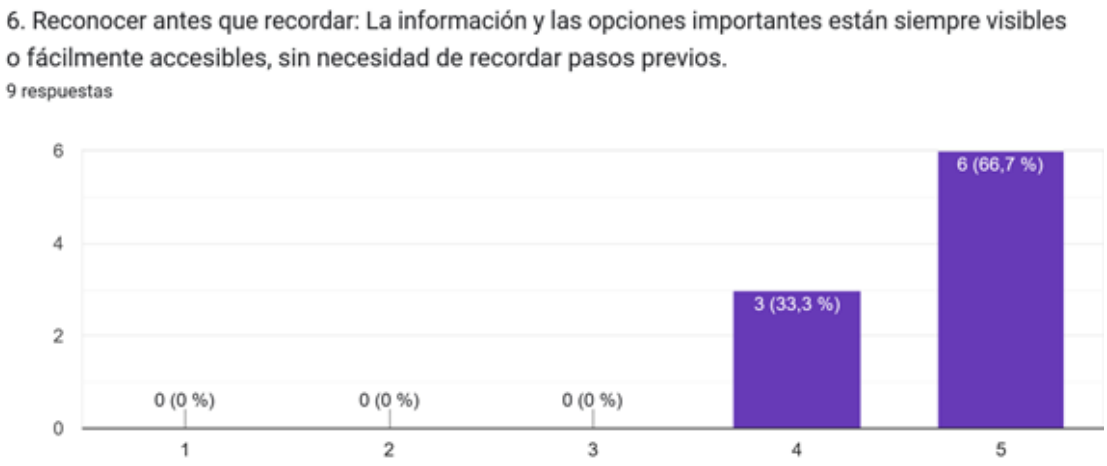

Figura 45: Resultados prueba Jakob Nielsen Reconocimiento en lugar de recuerdo
Fuente: Elaboración propia

Respecto a la heurística de flexibilidad y eficiencia de uso, los resultados muestran una valoración muy positiva, ya que el **66,7 %** indicó estar totalmente de acuerdo y el **33,3 %** de acuerdo en que la aplicación permite realizar tareas de manera eficiente tanto para usuarios nuevos como experimentados. Como se mencionó anteriormente, el hecho de tomar como inspiración el diseño de los chatbots actuales también contribuye a que la aplicación también destaque en esta heurística, ya que ofrece una interfaz sencilla y descriptiva para quienes la usan por primera vez, y al mismo tiempo resulta familiar para quienes ya interactúan con este tipo de herramientas.

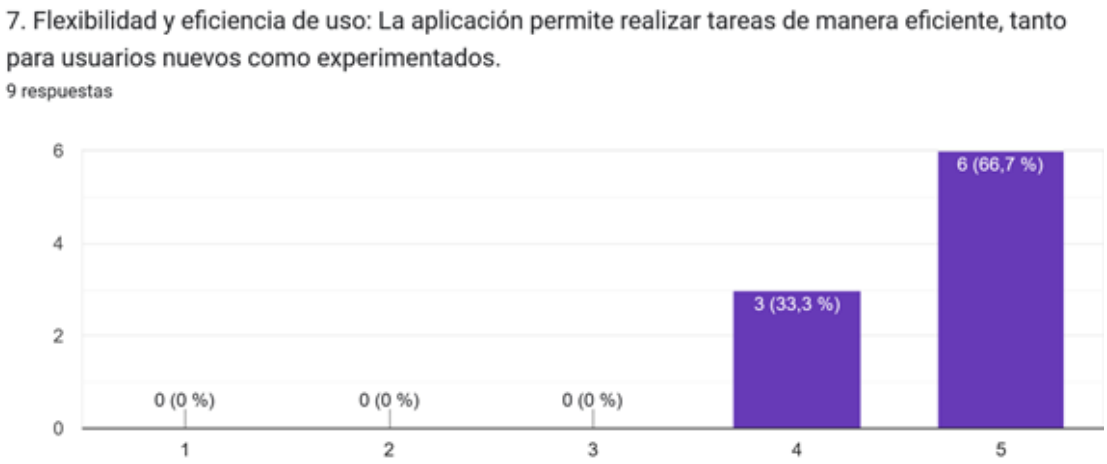

Figura 46: Resultados prueba Jakob Nielsen Flexibilidad y eficiencia de uso
Fuente: Elaboración propia

En cuanto a la heurística de diseño estético y minimalista, los resultados reflejan una percepción mayoritariamente positiva, el **66,7 %** de los participantes evaluó con la puntuación más alta y un **22,2 %** con una calificación alta, mientras que solo un 11,1 % se ubicó en un punto medio. Esto indica que, en general, los expertos valoran la claridad y simplicidad de la interfaz. Este tipo de interfaces con un diseño minimalista no solo resulta más agradable visualmente, sino que también reduce la carga cognitiva del usuario, permitiéndole centrarse en lo esencial sin distracciones.

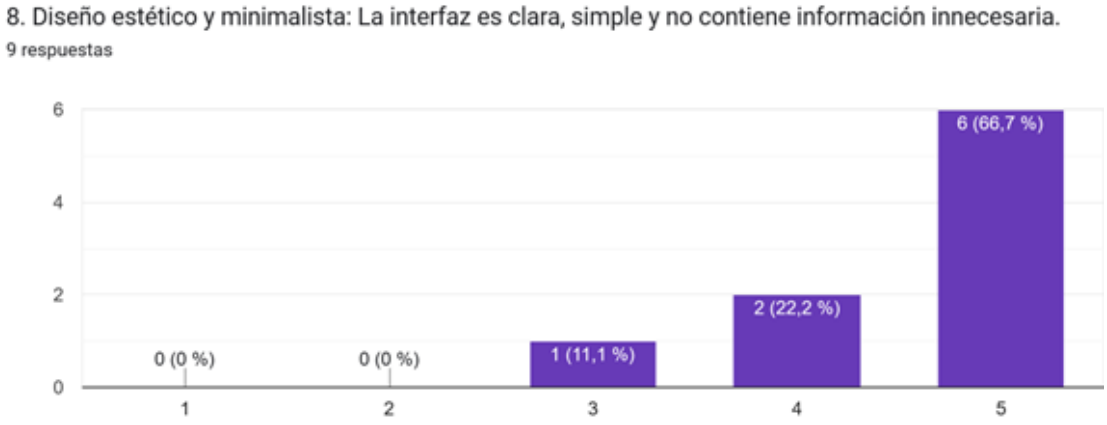

Figura 47: Resultados prueba Jakob Nielsen Diseño estético y minimalista
Fuente: Elaboración propia

Respecto a la heurística de ayuda a los usuarios a reconocer, diagnosticar y recuperarse de errores, la mayoría de los evaluadores tuvo una percepción positiva, el **44,4 %** puntuó con 4 y el **33,3 %** con la calificación máxima, lo que indica que en general se considera que los mensajes de error son adecuados y comprensibles, aunque un **22,2 %** otorgó puntuaciones bajas, lo que sugiere que hay margen de mejora. Este resultado puede explicarse por el hecho de que la aplicación, al estar centrada en la interacción con el chatbot, no cuenta con muchas situaciones en las que se presenten errores visibles para el usuario, los mensajes de error que aparecen provienen directamente de Streamlit y son genéricos, si bien estos mensajes cumplen con informar que ha ocurrido una falla, no explican claramente qué la provocó ni cómo resolverla, se podría mejorar implementando mensajes personalizados que indiquen de forma específica la causa del error.

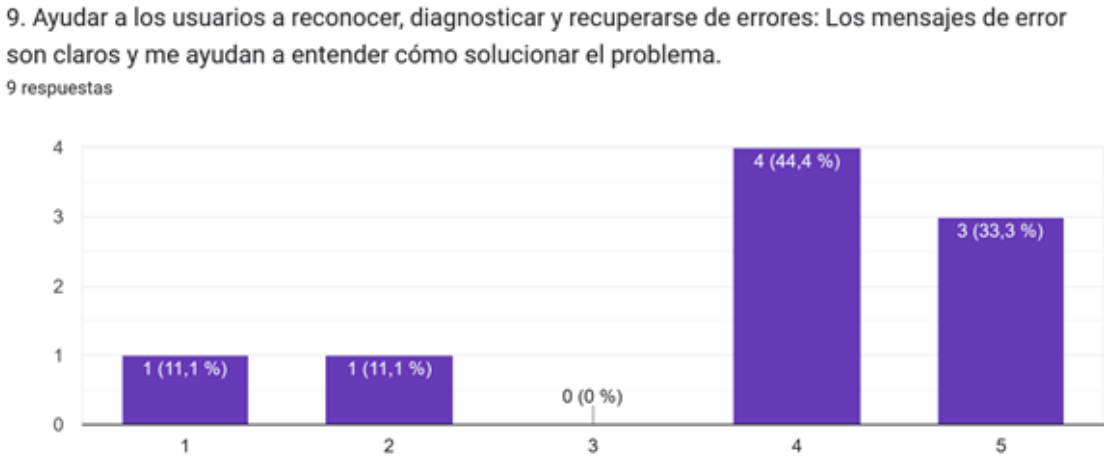

Figura 48: Resultados prueba Jakob Nielsen Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores

Fuente: Elaboración propia

Por último, en la heurística de ayuda y documentación, los resultados estuvieron divididos, el **33,3 %** de los participantes se mostró muy de acuerdo y otro **33,3 %** de acuerdo con la facilidad para encontrar ayuda o instrucciones, sin embargo, el **33,3 %** restante tuvo una percepción baja, lo que sugiere que aunque para algunos usuarios la aplicación es clara por sí misma, sería recomendable incluir una sección breve de ayuda o guía rápida para mejorar esta percepción y apoyar especialmente a quienes no están familiarizados con este tipo de herramientas.

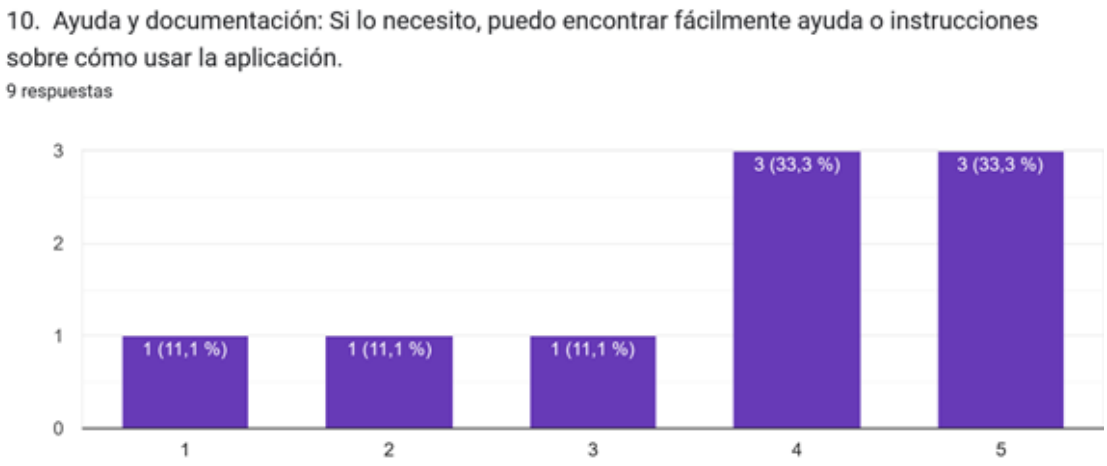

Figura 49: Resultados prueba Jakob Nielsen Ayuda y documentación
Fuente: Elaboración propia

En general, los resultados obtenidos a partir de la evaluación con las heurísticas de Jakob Nielsen reflejan una percepción positiva del diseño y funcionamiento de la aplicación. Se destacaron aspectos como la visibilidad del estado del sistema, la coherencia en la interfaz y la facilidad de uso tanto para usuarios nuevos como experimentados. Aunque se identificaron algunos puntos con oportunidades de mejora, especialmente en lo relacionado con el control de errores, la documentación y el control del usuario, la mayoría de los expertos valoró favorablemente la experiencia de uso, estos resultados permiten afirmar que la aplicación ofrece una interacción clara, fluida y coherente, y que con algunos ajustes podría mejorar aún más su usabilidad.

12. Conclusiones

Después de culminar todas las etapas de diseño, implementación y evaluación del proyecto, es posible concluir que:

- A lo largo del proyecto se evidenció, a partir de la recopilación de información sobre procesos administrativos y preguntas frecuentes que interrumpen el flujo de trabajo del personal, que los recursos humanos disponibles para atender consultas son limitados, lo cual puede generar cuellos de botella que afectan la experiencia de los estudiantes, por lo que herramientas como los chatbots toman relevancia, permitiendo atender un gran número de consultas al mismo tiempo, a diferencia del personal humano, lo que mejora la eficiencia en la atención a estudiantes, además de permitir al personal administrativo enfocar sus recursos en actividades que generan mayor valor.

- La Generación Aumentada por Recuperación (RAG) se destacó como la técnica de PLN más adecuada para este proyecto, tras la investigación de trabajos y antecedentes en el campo de la IA. Esta solución resultó práctica y eficiente, lo que permitió construir un chatbot sin la necesidad de entrenar un modelo desde cero, recuperar información de la base de conocimiento creada y dar respuestas precisas y confiables a los usuarios sin requerir grandes cantidades de procesamiento computacional. Además, esta técnica facilita la actualización de la información, lo que resulta indispensable en un entorno universitario donde constantemente hay cambios en los procesos y en la normativa.

- Durante el desarrollo del sistema se comprobó que Langchain es un marco de trabajo muy completo que permitió implementar el modelo de PLN seleccionado. Su compatibilidad con herramientas como ChromaDB, OpenAI API y Streamlit facilitó la integración de la técnica RAG y la construcción del chatbot, logrando una implementación funcional en un tiempo relativamente corto.

- Las pruebas de software demostraron que la aplicación web, aún funcionando con una key de OpenAI para uso personal, puede gestionar un número significativo de usuarios. En las pruebas por lotes el chatbot fue capaz de manejar hasta 100 usuarios simulando 10 de manera simultánea, registrando un tiempo promedio de 12.54 segundos, sin embargo, al simular 200 usuarios con lotes de 20 al mismo tiempo, el 96 % de las peticiones falló debido al límite de solicitudes por minuto (RPM) de la key usada, este límite se puede aumentar elevando el nivel de la key para aumentar el límite de RPM, aunque también implicaría un incremento en el costo de uso. De forma escalonada, lanzando 2 usuarios por segundo, la aplicación se mantuvo estable sin errores y alcanzó una capacidad teórica de 120 usuarios por minuto, con un tiempo de respuesta promedio de 6,9 segundos en la prueba de throughput.

- Si bien los chatbots demuestran ser herramientas poderosas, aún presentan limitaciones, y no son capaces de manejar todos los casos posibles, principalmente aquellos que se salen del dominio para el que fueron diseñados, por ello, no deben verse como

un reemplazo del personal humano, sino como un complemento que potencia sus capacidades y reduce su carga.

- Los modelos de lenguaje de gran tamaño (LLM) representan una revolución tecnológica, a pesar de que requieren altos costos, grandes cantidades de datos y tiempo para su entrenamiento, cada vez son más refinados, inteligentes y útiles, además de servir para crear chatbots, estos modelos se están utilizando en campos como la generación de código, el análisis de datos, la redacción de textos técnicos, y muchos más.

- La experiencia de este proyecto demuestra que los chatbots no solo tienen aplicación en entornos educativos, sino que su versatilidad les permite adaptarse a casi cualquier sector, como el de la salud, comercio, atención al cliente, soporte legal y más, esta capacidad de adaptación los convierte en una herramienta capaz de transformar la interacción entre instituciones y usuarios.

- Optar por una aplicación web permite que los usuarios pudieran acceder al sistema desde cualquier lugar, sin necesidad de instalar dependencias ni preocuparse por configuraciones complicadas o muy técnicas, esto facilitó la ejecución de las pruebas de usabilidad, ya que los participantes solo requerían conexión a internet y un navegador actualizado para interactuar con la aplicación y responder la encuesta.

- Los resultados de las pruebas de usabilidad evidencian que fue acertada la decisión de construir una interfaz inspirada en los chatbots modernos, ya que los usuarios pudieron utilizarla sin capacitación previa y se adaptaron con facilidad a su uso, lo que permitió evitar curvas de aprendizaje innecesarias. No obstante, se identificaron algunos puntos de mejora como aumentar las opciones para deshacer acciones, personalizar los mensajes de error y ofrecer ayuda más accesible, según lo evidenciado en la evaluación con las heurísticas de Nielsen.

- La implementación de un sistema de atención automatizada para resolver las consultas frecuentes de los estudiantes representa una estrategia eficaz para disminuir las interrupciones en el trabajo del personal administrativo. Implementando la Generación Aumentada por Recuperación se combina la recuperación de información con los LLM, lo que permite desarrollar chatbots capaces de actuar como medio de información para grandes volúmenes de consultas, evitando al personal administrativo tareas repetitivas y permitiendo optimizar el rendimiento de la oficina.

13. Trabajos futuros

A lo largo del desarrollo del proyecto y gracias a la retroalimentación obtenida durante las pruebas de usabilidad, se identificaron varias oportunidades de mejora de la interfaz y funcionalidades. Los resultados de las encuestas aplicadas con base en las heurísticas de Nielsen mostraron la necesidad de implementar mejoras en aspectos como la visibilidad del estado del sistema, la ayuda y documentación, y el manejo adecuado de errores, por lo que se plantea incorporar mensajes de error más claros y específicos, así como agregar secciones de ayuda que orienten al usuario en el uso correcto de la aplicación.

Adicionalmente, con el objetivo de mejorar la experiencia de navegación, se podrían incluir funcionalidades como un botón que permita ir hasta el final de la conversación, y una opción para limpiar el historial del chat, borrando el contexto anterior e iniciar una nueva conversación.

Una de las mejoras que se tenía planeada como trabajo futuro, pero se terminó implementando debido a las recomendaciones recurrentes, fue mejorar el tiempo de respuesta percibido del sistema. Inicialmente el chatbot mostraba la respuesta completa una vez se había generado en su totalidad, lo que generaba una sensación de lentitud, por eso se implementó el renderizado de la respuesta token a token, que permite al usuario visualizar la respuesta de manera progresiva a medida que se genera, resultando en un impacto notable en la percepción de agilidad y en la experiencia de uso.

En cuanto a las funcionalidades, un trabajo futuro fundamental es ampliar la base de conocimiento del sistema, de modo que el chatbot sea capaz de resolver un mayor número de dudas estudiantiles, esta expansión podría incluir tanto información del mismo programa académico, como también extenderse a otros programas de la Universidad del Valle, haciendo que la herramienta sea de utilidad a nivel general en toda la Univerdidad. Además, LangChain ofrece la capacidad de agregar contenido proveniente de páginas web dentro de la base de conocimiento, al implementar esta característica, sería posible hacer que periódicamente se actualice la información tomada de las páginas, teniendo así respuestas siempre actualizadas y pertinentes para los usuarios.

14. Referencias

[1] M. A. P. Arteaga, C. R. C. Plúa, H. B. D. Lucas, and L. R. M. Quimiz, "Chatbots para ventas y atención al cliente," *Journal TechInnovation*, vol. 1, 2022.

[2] R. A. Manjarrés-Betancur and M. M. Echeverri-Torres, "Asistente virtual académico utilizando tecnologías cognitivas de procesamiento de lenguaje natural," *Revista Politécnica*, vol. 16, 2020.

[3] J. Medina, E. M. Eisman, and J. L. Castro, "Asistentes virtuales en plataformas 3.0. (spanish)," *Informática Educativa Comunicaciones*, 2013.

[4] M. de Educación Nacional, "Aumentó la matrícula en educación superior en 2022 según el sistema nacional de información de educación superior (snies) — ministerio de educación nacional," 8 2023.

[5] F. Carvajal, D. A. Guerrero, L. M. Parra, Y. B. Vélez, D. P. Sevilla, and M. P. Osorio, "Dirección universitaria presentó la rendición de cuentas 2022 - universidad del valle / cali, colombia," 3 2023.

[6] F. Carvajal, D. A. Guerrero, L. M. Parra, Y. B. Vélez, D. P. Sevilla, and M. P. Osorio, "Así avanza la matrícula en univalle - universidad del valle / cali, colombia," 12 2021.

[7] U. del Valle Sede Tuluá, "Informe de rendición de cuentas 2020," tech. rep., Universidad del Valle Sede Tuluá, 6 2021.

[8] A. T. Alzate, "Metodología teoría general de sistemas," *Revista del Departamento de Ciencias*, vol. 8, 1999.

[9] A. S. Tejeda, "Mejoras de lean manufacturing en los sistemas productivos," *Ciencia y Sociedad*, vol. 36, 2011.

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. 3 ed., 12 2009.

[11] IBM, "¿qué es el procesamiento del lenguaje natural (pln)?," 2024. Consultado en mayo de 2025.

[12] AWS, "¿qué es rag?: explicación de la ia de generación aumentada por recuperación, aws."

[13] IBM, "¿qué es un chatbot? — ibm."

[14] U. I. de La Rioja, "¿qué es la productividad laboral y cómo se mide? — unir," 10 2022.

[15] L. Shklar and R. Rosen, "Web application architecture: Principles, protocols and practices," tech. rep., 2002.

[16] AWS, "¿qué es un llm? - explicación de los modelos de lenguaje grandes - aws," 2020.

[17] J. Barnard, "¿qué es embedding? — ibm," 12 2023.

[18] D. Hill, "Introducing martha: Leveraging cognitive automation to create an exceptional student experience – bmc software — blogs," 2 2019.

[19] Y. Li, J. Zhao, M. Li, Y. Dang, E. Yu, J. Li, Z. Sun, U. Hussein, J. Wen, A. M. Abdelhameed, J. Mai, S. Li, Y. Yu, X. Hu, D. Yang, J. Feng, Z. Li, J. He, W. Tao, T. Duan, Y. Lou, F. Li, and C. Tao, "Refai: a gpt-powered retrieval-augmented generative tool for biomedical literature recommendation and summarization," *Journal of the American Medical Informatics Association*, vol. 31, pp. 2030–2039, 9 2024.

[20] A. F. T. Cambizaca, "Desarrollo de un chatbot que ayude a responder a preguntas frecuentes referentes a becas en la universidad técnica particular de loja," 2018.

[21] Colciencias, "Departamento administrativo de ciencia, tecnologÍa e innovaciÓn-colciencias-2ª convocatoria ecosistema cientÍfico para la financiaciÓn de programas de i+d+i que contribuyan al mejoramiento de la calidad de las instituciones de educaciÓn superior colombianas-2017 anexo 13 niveles de madurez tecnolÓgica," tech. rep., 2016.

[22] J. Canós, P. Letelier, and C. Penadés, "Metodologías Ágiles en el desarrollo de software organización," *DSIC -Universidad Politécnica de Valencia*, 2003.

[23] B. Marcos, "El kanban," *Uiversitat Oberta de Ctalunya*, vol. universida, 2011.

[24] Atlassian, "Jira — software de seguimiento de proyectos e incidencias — atlassian."

[25] M. Rehkopf, "Historias de usuario — ejemplos y plantilla — atlassian."

[26] N. Segovia-García, "Optimización de la atención estudiantil. una revisión del uso de chatbots de ia en la educación superior," *European Public & Social Innovation Review*, vol. 9, pp. 1–20, 7 2024.

[27] J. Cordero, A. Toledo, F. Guaman, and L. Barba-Guaman, "Use of chatbots for user service in higher education institutions," *Iberian Conference on Information Systems and Technologies, CISTI*, vol. 2020-June, 6 2020.

[28] O. D. L. Granizo and M. L. Granizo, "Desarrollo de un asistente virtual (chatbot) para mejorar el acceso a la información recurrente por los estudiantes de instituciones de educación superior," *Ecuadorian Science Journal, ISSN-e 2602-8077, Vol. 4, Nº. 2, 2020 (Ejemplar dedicado a: Septiembre), págs. 111-116*, vol. 4, pp. 111–116, 2020.

[29] A. R. Kandula, M. Tadiparthi, P. Yakkala, S. Pasupuleti, P. Pagolu, and S. M. C. Potharlanka, "Design and implementation of a chatbot for automated legal assistance using natural language processing and machine learning," *2023 Annual International Conference on Emerging Research Areas: International Conference on Intelligent Systems, AICERA/ICIS 2023*, 2023.

[30] D. Miriam, Z. S. Hernández, D. María, Y. V. Flores, M. C. Abel, A. P. Estrada, D. Yarib, and G. Luna, "Implementación de un software inteligente (chatbot) para agilizar los procesos de comunicación y servicios coadyuvando en la transformación digital del sector educativo de nivel superior," tech. rep., 2021.

[31] G. da Silva Gonçalves, T. de Luca Sant'ana Ribeiro, J. E. V. Teixeira, and B. K. Costa, "The deployment of chatbot to improve customer service in higher education institutions during covid-19," *International Journal of Innovation: IJI Journal, ISSN-e 2318-9975, Vol. 10, Nº. 1, 2022, págs. 178-203*, vol. 10, pp. 178–203, 2022.

[32] J. R. C. Mamani, Y. J. R. Alamo, J. A. A. Aguirre, and E. E. G. Toledo, "Cognitive services to improve user experience in searching for academic information based on chatbot," *Proceedings of the 2019 IEEE 26th International Conference on Electronics, Electrical Engineering and Computing, INTERCON 2019*, 8 2019.

[33] M. Mittal, G. Battineni, D. Singh, T. Nagarwal, and P. Yadav, "Web-based chatbot for frequently asked queries (faq) in hospitals," *Journal of Taibah University Medical Sciences*, vol. 16, pp. 740–746, 10 2021.

[34] W. José, B. León, M. Antonio, and M. Baltodano, "Integración de un chatbot basado en chatgpt para optimizar la gestión administrativa en inblen sa," *REICE: Revista Electrónica de Investigación en Ciencias Económicas*, vol. 11, pp. 274–299, 12 2023.

[35] A. Bansal and S. Suddala, "Enhancing generative ai capabilities through retrieval-augmented generation systems and llms," *Library Progress International*, vol. 44, pp. 17765–17775, 10 2024.

[36] F. Shah-Mohammadi and J. Finkelstein, "Extraction of substance use information from clinical notes: Generative pretrained transformer-based investigation.," *JMIR medical informatics*, vol. 12, p. e56243, 8 2024.

[37] G. Lang and T. Gürpinar, "Ai-powered learning support: A study of retrieval-augmented generation (rag) chatbot effectiveness in an online course.," *Information Systems Education Journal*, vol. 23, pp. 4–13, 2025.

[38] A. Kumar, P. Kumar, A. Kumar, and V. Kumar, "Integrating retrieval-augmented generation (rag) in large language models (llms): An approach for faster and accurate search," *Library Progress International*, vol. 44, pp. 1070–1078, 10 2024.

[39] Z. Chen, D. Zou, H. Xie, H. Lou, and Z. Pang, "Facilitating university admission using a chatbot based on large language models with retrieval-augmented generation," *Educational Technology & Society*, vol. 27, pp. 454–470, 10 2024.

[40] OpenAI, "Pricing - openai api," 2025.

[41] OpenAI, "Models - openai api," 2025.

[42] OpenAI, "Introducing gpt-4.1 in the api — openai," 4 2025.

[43] Z. Mowshowitz, “Gpt-4.1 is a mini upgrade - by zvi mowshowitz,” 4 2025.

[44] LangChain, “Chatopenai — langchain.”

[45] LangChain, “Langchain.”

[46] LangChain, “Streaming — langchain.”

[47] F. Pedrazzini, “Métodos rag avanzados: explicación de métodos simples, híbridos, agentes y gráficos,” 9 2024.

[48] R. U. Shin, “Principales técnicas y algoritmos de recuperación de información,” 9 2024.

[49] S. A. Mukund and K. S. Easwarakumar, “Optimizing legal text summarization through dynamic retrieval-augmented generation and domain-specific adaptation.,” *Symmetry (20738994)*, vol. 17, p. 633, 4 2025.

[50] Streamlit, “Streamlit documentation.”

[51] IONOS, “Websocket — un canal de comunicación para la web en tiempo real - ionos,” 7 2020.

[52] IONOS, “¿qué es chroma db? - ionos,” 3 2025.

[53] C. Cookbook, “Storage layout - chroma cookbook.”

[54] QAlified, “Pruebas end-to-end: definición, ejemplos y herramientas,” 11 2022.

[55] MongoDB, “Chatbot demo builder.”

[56] NNgroup, “10 heurísticas de usabilidad para el diseño de interfaces de usuario - nn/g,” 1 2024.

15. Anexos

1. *Historias de usuario*
2. *Procesos Académicos y Administrativos - Ingeniería de Sistemas Univalle*
3. *Repositorio de la aplicación*
4. *Repositorio de las pruebas de software*