Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Electricidad y Computación

Desarrollo de una plataforma web de apoyo académico para la generación de respuesta aumentadas por recuperación utilizando inteligencia artificial

TECH-376

Proyecto Integrador

Previo la obtención del Título de:

Ingeniero en Ciencias de la Computación

Presentado por:

Alfredo Roman Calle Aguilar

Nicolás Enrique Joniaux Echeverria

Guayaquil - Ecuador

Año: 2024

Dedicatoria

El presente proyecto lo dedico con todo mi corazón a mis queridos padres y abuelos. Gracias a su inquebrantable apoyo y a las sabias enseñanzas que me han brindado a lo largo de mi vida, he podido crecer no solo como persona, sino también como el primer profesional de nuestra familia. A pesar de la sencillez con la que siempre han vivido, los valores que me han transmitido han sido la fuerza que me ha impulsado a superar cada obstáculo y a alcanzar mis sueños. Este logro es un reflejo de su amor, su sacrificio y su fe inquebrantable en mí. Hoy, con inmensa gratitud, les ofrezco este trabajo como un humilde tributo a todo lo que me han dado y a todo lo que soy gracias a ellos.

Alfredo Calle Aguilar

Dedicatoria

Dedico el proyecto a mis seres queridos, gracias por su apoyo.

Nicolás Joniaux Echeverria

Agradecimientos

Mi más sincero agradecimiento al M. Sc. Rodrigo Saraguro, tutor de proyecto, y al Ph. D. Boris Vintimilla, profesor de materia integradora, por su guía constante y valiosas orientaciones durante el desarrollo de este proyecto. Asimismo, extiendo mi gratitud al Ph. D. Miguel Realpe por brindarme la oportunidad de colaborar en su proyecto, lo cual ha sido una experiencia fundamental para mi formación académica y profesional. También quiero agradecer profundamente a todos los profesores que me han acompañado a lo largo de mi vida universitaria, quienes con su dedicación y enseñanzas han contribuido significativamente a mi crecimiento personal y académico.

Alfredo Calle Aguilar

Agradecimientos

Agradezco a los profesores de la carrera de Computación, a quienes respeto mucho.

Nicolás Joniaux Echeverria

Declaración Expresa

Nosotros Alfredo Roman Calle Aguilar y Nicolás Enrique Joniaux Echeverria acordamos y reconocemos que:

La titularidad de los derechos patrimoniales de autor (derechos de autor) del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra autorizada a sublicenciar, incluyendo la autorización para su divulgación, y para crear y usar obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores.

La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de nuestra innovación, de ser el caso.

En los casos en que la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique a los autores que existe una innovación patentable sobre los resultados del proyecto de graduación, no se publicarán o divulgarán alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 27 de septiembre del 2024.

Firmado electrónicamente por:
NICOLAS ENRIQUE JONIAUX ECHEVERRIA

Nicolás Enrique Joniaux Echeverria

Firmado electrónicamente por:
ALFREDO ROMAN CALLE AGUILAR

Alfredo Roman Calle Aguilar

Evaluadores

Firmado electrónicamente por:
BORIS XAVIER
VINTIMILLA BURGOS

Ph. D. Boris X. Vintimilla Burgos

Profesor de Materia

Firmado electrónicamente por:
RODRIGO ALEXANDER
SARAGURO BRAVO

M. Sc. Rodrigo A. Saraguro Bravo

Tutor de proyecto

Resumen

El proyecto tiene como objetivo desarrollar un *chatbot* inteligente integrado en una plataforma web para asistir a estudiantes en la materia de Fundamentos de Programación. Este sistema propone mejorar el acceso a información actualizada y precisa, facilitando el aprendizaje de los estudiantes mediante un modelo de recuperación y generación de respuestas (RAG) basado en inteligencia artificial. El proyecto plantea que la integración de un pipeline de ingesta de datos con fuentes confiables incrementará la efectividad del *chatbot* en la asistencia para resolver problemas de programación. Para el desarrollo del proyecto, se utilizó React.js para la interfaz, FastAPI para el *backend*, y un modelo RAG para la generación de respuestas. Los datos se indexaron mediante un pipeline que permitió a los profesores cargar documentos relevantes. Los resultados mostraron que la mayoría de los estudiantes obtuvieron calificaciones destacadas, con un 91.3% logrando "Excelente" o "Muy Bueno". Además, se registró una satisfacción del 60.9% del *chatbot*. En conclusión, el sistema desarrollado cumple con los objetivos planteados, proporcionando un apoyo educativo efectivo en Fundamentos de Programación.

Palabras Clave: Chatbot, Inteligencia Artificial, Programación, Educación.

Abstract

The project aims to develop an intelligent chatbot integrated into a web platform to assist students in the Fundamentals of Programming course. This system is designed to improve access to updated and accurate information, facilitating students' learning through an artificial intelligence-based Retrieval-Augmented Generation (RAG) model. The project proposes that integrating a data ingestion pipeline with reliable sources will enhance the chatbot's effectiveness in assisting with programming problems. During the development of the project, React.js was used for the frontend, FastAPI for the backend, and a RAG model for response generation. Data was indexed through a pipeline that allowed professors to upload relevant documents. The results showed that most students achieved outstanding grades, with 91.3% reaching "Excellent" or "Very Good." Additionally, 60.9% satisfaction with the chatbot was recorded. In conclusion, the developed system meets its objectives, providing effective educational support in Fundamentals of Programming.

Keywords: Chatbot, Artificial Intelligence, Fundamentals of Programming, Education.

Índice general

Abreviaturas

ACM	Asociación de maquinaría computacional
GPT	Modelo “Transformer” Preentrenado Generativo
IA	Inteligencia Artificial
LLMs	Modelos de lenguaje de gran tamaño
NLP	Procesamiento de lenguaje natural
RAG	Generación aumentada por recuperación
SOTA	Estado del arte

Índice de figuras

Índice de tablas

Capítulo 1

1. Introducción

Los modelos de inteligencia artificial (IA), especialmente aquellos basados en el aprendizaje automático y el procesamiento del lenguaje natural, tienen un gran potencial para proporcionar información útil y precisa [1]. Sin embargo, las respuestas generadas por estos modelos suelen tener inconsistencias por la falta de actualización y calidad de las fuentes utilizadas en su entrenamiento. Este problema es constantemente preocupante en contextos educativos y empresariales, donde la precisión y la fiabilidad de la información son cruciales [2].

En el ámbito educativo, las instituciones necesitan respuestas exactas y verificables para apoyar la enseñanza y el aprendizaje. Sin embargo, los modelos de IA frecuentemente generan información basada en datos no confiables o desactualizados [2], lo que puede llevar a la difusión de conocimientos incorrectos. Esta situación afecta especialmente a los estudiantes novatos, quienes enfrentan dificultades para encontrar información precisa y comprensible, complicando su proceso de aprendizaje [2], [3].

El presente proyecto propone solucionar estos desafíos mediante el desarrollo de un *chatbot* integrado a una aplicación web. Este *chatbot* empleará un modelo de IA que utilizará fuentes de datos más confiables y actualizadas, gestionadas por un profesor. Los contenidos utilizados estarán basados en los estándares de la ACM y en libros 100% confiables. Además, se evaluará el impacto de este modelo en entornos académicos, con el propósito de mejorar la calidad, precisión y fiabilidad de la información proporcionada por los sistemas de IA.

1.1 Descripción del problema

Los modelos de inteligencia artificial (IA) tienen un gran potencial para proporcionar información útil y accesible en una amplia variedad de campos, desde la educación hasta en el ámbito empresarial. Sin embargo, una de las mayores preocupaciones cuando se trata de los modelos de IA existentes es su inconsistencia, debido a la calidad y falta de actualización de las fuentes de entrenamiento utilizadas [2], [4]. El estudio [5] menciona que los modelos de IA

presentan problemas de inconsistencia por la información desactualizada que manejan, además no poseen la capacidad para comprender o evaluar la calidad y exactitud de la información que proporcionan, limitándose solo a generar una respuesta.

En la educación, los *chatbots* y otros sistemas que utilizan modelos de IA se utilizan para apoyar la enseñanza y el aprendizaje. Sin embargo, estos sistemas constantemente proporcionan respuestas basadas en información poco confiable o desactualizada, lo que puede llevar a la difusión de información errónea entre los estudiantes [1], [4]. Esto es problemático para los estudiantes novatos, que tienen dificultades para obtener información precisa y clara, necesaria para fortalecer su aprendizaje [3].

Además, los *chatbots* son incapaces de comprender el contexto y los matices del lenguaje humano puede generar respuestas incorrectas o inapropiadas. Por ejemplo, el *chatbot* puede no entender el contexto de una pregunta o confundir términos similares pero diferentes, lo que implica información incorrecta o confusa [4], [6]. También, la falta de transparencia en el funcionamiento de los *chatbots* dificulta la identificación y corrección de errores, lo que incrementa el riesgo de utilizar sistemas poco confiables y precisos [4].

Por último, la responsabilidad sobre la exactitud de la información proporcionada por los *chatbots* a menudo resulta poco clara, lo que puede comprometer la integridad y fiabilidad de los resultados obtenidos [2], [4].

1.2 Justificación del problema

Realizar este proyecto es relevante debido a que es necesario mejorar la precisión y confiabilidad de la información proporcionada por los modelos de IA en los ámbitos educativo y empresarial. La optimización de estos modelos, utilizando fuentes de datos más confiables y actualizadas, contribuirá significativamente a reducir la difusión de información incorrecta y a aumentar la fiabilidad de las respuestas generadas [2], [4].

En el ámbito educativo, es un problema crítico para los estudiantes novatos, que dependen de estos sistemas que usan modelos de IA para obtener información rápida y fiable. Además, los profesores enfrentan dificultades en la provisión de recursos educativos de alta calidad, lo que puede impactar negativamente en los resultados del aprendizaje. Mientras que, en el ámbito empresarial, la falta de información precisa y confiable puede llevar a decisiones mal informadas, disminuyendo la productividad y la eficiencia operativa [3], [7].

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar un *chatbot* interactivo integrado en una plataforma web que, utilizando técnicas de inteligencia artificial, asista a los estudiantes en la asignatura de Fundamentos de Programación mediante la información proporcionada por los profesores y documentos indexados.

1.3.2 Objetivos específicos

1. Diseñar la arquitectura del sistema que define la interacción de los componentes para garantizar la calidad del software, incluyendo la modularidad, escalabilidad y mantenibilidad del sistema.
2. Integrar un pipeline de ingesta de datos que indexe documentos relevantes de diversas fuentes, asegurando que la información esté estructurada y accesible para su recuperación y análisis.
3. Integrar un modelo RAG que combine técnicas de recuperación de información y generación de texto para extraer y generar respuestas precisas y contextualizadas a partir de los documentos indexados.
4. Desarrollar un *chatbot* que interprete las consultas de los usuarios y utilice el modelo RAG para proporcionar respuestas claras y útiles, todo ello integrado en una aplicación web con una interfaz fácil de usar destinada a estudiantes y profesores.

5. Realizar pruebas del sistema con estudiantes de la materia de Fundamentos de Programación para asegurar su funcionalidad, efectividad y facilidad de uso, ajustando y optimizando el sistema basado en los resultados de las pruebas y el *feedback* recibido.

1.4 Marco teórico

En los últimos años se ha dado un gran crecimiento en el uso de *chatbots* en diversos campos, como atención médica, marketing, educación, sistemas de apoyo, patrimonio cultural, entretenimiento y muchos otros. Estas aplicaciones pueden comprender las preguntas mediante el uso de técnicas de procesamiento del lenguaje natural y ontologías de dominio para generar respuestas al usuario. [11]

Los *chatbots* actualmente desempeñan un papel importante en varios campos, y la integración de IA ha permitido crear asistentes tanto para profesores como a estudiantes, optimizando la experiencia educativa [2], [4], [5], ofreciendo tutorías personalizadas, asistencia con tareas, comprensión de conceptos, preparación para exámenes, facilitación de discusiones y colaboración, así como apoyo en la salud mental [2]. Entre los *chatbots* basados en IA más destacados en la educación se encuentran:

- Bard, introducido en 2022 por Google AI, es un *chatbot* con un modelo de lenguaje grande (LLMs) capaz de generar texto, traducir idiomas, producir contenido creativo diverso y proporcionar respuestas informativas a preguntas.
- ChatGPT, lanzado en 2022 por OpenAI, es un *chatbot* de gran escala que puede generar texto, crear contenido diverso y responder preguntas informativamente.
- Ada, lanzado en 2017, ofrece tutoría personalizada a los estudiantes, respondiendo preguntas y proporcionando retroalimentación para facilitar el aprendizaje individualizado.

- Replika, introducida en 2017, es una plataforma de *chatbot* diseñada para ser una compañía y apoyo emocional para los estudiantes, escuchando sus problemas, ofreciendo consejos y ayudándoles a sentirse menos solos.
- Socratic, lanzado en 2013 y adquirido por Google en 2018, tiene como objetivo hacer el aprendizaje accesible para todos los estudiantes. Aunque no es un *chatbot* en el sentido estricto, su interfaz y funcionalidad se asemejan a las de un *chatbot*, ayudando a los estudiantes a aprender nuevos conceptos.
- Habitica, lanzada en 2013, ayuda a los estudiantes a desarrollar buenos hábitos de estudio mediante la gamificación del proceso de aprendizaje, haciéndolo más atractivo y divertido
- Piazza, lanzada en 2009, facilita el debate y la colaboración en entornos educativos, permitiendo a estudiantes e instructores participar en discusiones, hacer preguntas y compartir información relevante sobre el contenido del curso.

En esta sección se presenta una revisión de la literatura relacionada con el uso de *chatbots* educativos, la implementación de RAG y su relevancia en el ámbito académico.

1.4.1 Chatbots en la actualidad, tipos y arquitecturas.

Los *chatbots* han evolucionado significativamente, especialmente con la adopción de la arquitectura *Transformer*, específicamente mediante modelos “decoder-only”, que han dominado el mercado de los *chatbots*. Estos modelos, como GPT-4 de OpenAI y Gemini de Google, se especializan en generar texto complejo, lo que los hace ideales para aplicaciones de chatbot donde se requiere generar respuestas coherentes y contextuales. Empresas como Anthropic continúan desarrollando estos modelos con un enfoque en la seguridad y la interpretabilidad. Para facilitar el desarrollo y despliegue de chatbots, plataformas como Anyscale, TogetherAI y OpenRouter ofrecen herramientas robustas que permiten a los desarrolladores escalar y gestionar sus aplicaciones de chatbot eficientemente, adaptándose a las necesidades de negocios y usuarios finales.

1.4.2 Retrieval Augmented Generation (RAG)

La generación aumentada por recuperación (RAG) es una técnica avanzada que combina modelos de lenguaje grande (LLMs) con sistemas de recuperación de información para mejorar la precisión y relevancia de las respuestas generadas. Los LLMs, como GPT-4, han alcanzado un estado del arte en procesamiento de lenguaje natural, capaces de generar texto coherente y contextual a partir de entradas complejas. Sin embargo, estos modelos pueden sufrir de alucinaciones y proporcionar información desactualizada. Al integrar RAG, se aprovecha el conocimiento almacenado en bases de datos externas, lo que permite a los LLMs generar respuestas más precisas y actualizadas, mejorando significativamente su rendimiento y aplicabilidad en entornos educativos y otros dominios especializados.

La generación aumentada por recuperación (RAG) funciona en dos etapas principales: primero, un componente de recuperación busca información relevante en bases de datos externas basadas en la consulta del usuario; luego, un modelo generativo utiliza esta información recuperada para producir respuestas más precisas y contextualmente relevantes. Esta integración permite a los LLMs complementar su conocimiento interno con datos actualizados y específicos del dominio, mitigando problemas como las alucinaciones y mejorando la confiabilidad de las respuestas.

1.4.3 Progresión y estado del arte

La técnica de *Retrieval Augmented Generation* (RAG) es una solución prometedora para mejorar la precisión y credibilidad de los modelos de lenguaje grande (LLMs) al incorporar conocimiento de bases de datos externas. RAG aborda problemas como la alucinación [8], el conocimiento desactualizado y los procesos de razonamiento no transparentes e inverificables. Esta técnica permite la actualización continua del conocimiento y la integración de información específica del dominio, lo cual es crucial para tareas intensivas en conocimiento.

Se proporciona una revisión detallada de la evolución de los paradigmas RAG, desde el RAG *Naive* hasta el RAG Avanzado y el RAG Modular, examinando las técnicas de recuperación, generación y augmentación involucradas en estos sistemas.

1.4.4 Mejora de la precisión factual

Se abordo [9] el desafío de las alucinaciones en LLMs mediante la integración de RAG para mejorar la precisión factual en consultas específicas de dominio y sensibles al tiempo. Su investigación demostró la efectividad de un sistema RAG en la generación de respuestas más precisas para consultas relacionadas con bases de conocimiento privadas.

1.4.5 Interacción entre el conocimiento interno del modelo y la información recuperada

Se investigo [10] la interacción entre el conocimiento interno de los LLMs y la información recuperada por sistemas RAG. Su análisis reveló que, aunque proporcionar la información recuperada correcta puede corregir la mayoría de los errores del modelo, existe una tensión subyacente entre el conocimiento previo del modelo y la información presentada en los documentos de referencia. Los resultados indicaron que los LLMs son más propensos a recitar información incorrecta cuando su conocimiento interno es más débil y la información recuperada está perturbada, lo que subraya la importancia de la calidad y exactitud de los datos recuperados.

1.4.6 Proyectos similares

Los *chatbots* se han utilizado en varios proyectos de investigación. Uno de ellos fue "*Quizbot*" desarrollado por La Universidad de Stanford, diseñado para ayudar a los estudiantes en la preparación de exámenes mediante preguntas interactivas. Los resultados mostraron que los estudiantes que interactuaron con "*QuizBot*" tuvieron un rendimiento superior en comparación con aquellos que utilizaron métodos tradicionales de estudio. [9]

El presente proyecto se distingue de otros *chatbots* basados en IA porque permite a los profesores actualizar los contenidos, documentos o archivos que el *chatbot* utiliza para proporcionar respuestas. Esto asegura que la información ofrecida sea siempre actualizada y

provenga de fuentes confiables, manteniendo su relevancia y precisión. A diferencia de otros sistemas de IA, nuestra plataforma ofrece respuestas actuales y adaptadas a las necesidades específicas de los usuarios. Además, ofrece la ventaja de crear modelos de IA con material de estudio personalizable e intuitivo, superando en flexibilidad y adaptabilidad de otros modelos disponibles.

1.4.7 Ventajas del proyecto vs LLM en la nube

- Modelo Local + RAG: Ofrece una gran privacidad ya que el procesamiento se realiza localmente y la información sensible no necesita salir del entorno local. Al utilizar RAG, puede accederse a fuentes externas de información, pero esto puede configurarse para que sólo recupere de bases de datos seguras y controladas.
- ChatGPT y Gemini: Como modelos basados en la nube, involucran un grado de riesgo de privacidad más elevado porque los datos se procesan en servidores externos. Esto puede ser una preocupación en entornos académicos donde la privacidad de los datos es crítica.
- Modelo Local + RAG: Esta configuración permite una gran personalización. RAG puede configurarse para consultar específicamente bases de datos académicas o repositorios de investigación, asegurando que la información generada sea relevante y precisa dentro del ámbito académico que se está estudiando.
- ChatGPT y Gemini: Estos modelos están entrenados con una amplia gama de textos de Internet, lo que incluye, pero no se limita a material académico. Si bien pueden proporcionar respuestas de alta calidad basadas en una amplia gama de fuentes, no están específicamente restringidos a materiales académicos y pueden incluir información no verificada o menos relevante.

El proyecto plantea el desarrollo de una plataforma educativa que emplea técnicas de recuperación de información y generación automática de texto (RAG) junto con Inteligencia

Artificial (IA) para asistir a los estudiantes en la asignatura de Fundamentos de Programación. Para lograr el objetivo, el proyecto se divide en 5 módulos, cada uno aborda aspectos específicos del desarrollo de la plataforma. A continuación, se describen brevemente los módulos clave a resolver:

1. Diseño del sistema: Este módulo se enfoca en la arquitectura del sistema y cómo interactúan los componentes parar llevar a cabo sus responsabilidades, asegurando la calidad del software.
2. Integración de un pipeline de ingesta de datos: El segundo módulo se centra en la recolección y estructuración de documentos relevantes de diversas fuentes.
3. Integración del modelo RAG: El tercer módulo implica la integración de un modelo que combine técnicas de recuperación de información con generación de texto.
4. Desarrollo del *chatbot*: El cuarto módulo se enfoca en la creación de un *chatbot* interactivo que interprete las consultas de los usuarios y utilice el modelo RAG para proporcionar respuestas claras y útiles.
5. Pruebas del sistema: El quinto módulo se enfoca en las pruebas del *chatbot* y la aplicación web con los estudiantes de la asignatura de Fundamentos de Programación, asegurando su funcionalidad y efectividad.

En conjunto, estos módulos constituyen la base del desarrollo de la plataforma educativa, abordando tanto la recopilación y organización de información como la interacción con los usuarios y la generación de respuestas educativas.

Capítulo 2

2. Metodología

En este capítulo se describe la metodología utilizada para desarrollar un *chatbot* de inteligencia artificial integrado a una aplicación web, destinado a asistir a los estudiantes de la asignatura de Fundamentos de Programación.

Para iniciar, se realizó una reunión con el cliente con el objetivo de levantar los requerimientos del proyecto. Durante esta reunión, se expuso la problemática actual en la enseñanza de Fundamentos de Programación. El cliente destacó la necesidad de proporcionar a los estudiantes una herramienta que mejore su aprendizaje y permita a los profesores controlar los contenidos utilizados por el *chatbot* para asegurar respuestas actualizadas y claras. Basado en esta información, se definieron los requerimientos funcionales y no funcionales del sistema. Cada requerimiento fue revisado y aprobado por el cliente y el equipo del proyecto para garantizar su relevancia y viabilidad.

2.1 Requerimientos del sistema

Los siguientes requerimientos se adaptan a las necesidades del cliente, que busca una herramienta de IA para apoyar la enseñanza de estudiantes y, a su vez, gestionar su contenido para tener respuestas actuales. Estos requerimientos se clasifican en funcionales y no funcionales.

Requerimientos funcionales:

- Diseñar la arquitectura del sistema que define la interacción de los componentes para garantizar la calidad del software.
- Desarrollar sistemas de entrada y registro para que los usuarios puedan tener acceso al sistema.
- Integrar un pipeline de ingesta de datos que procese documentos y los estructure para su uso por el modelo RAG.
- Integrar el modelo RAG basado en los documentos indexados.
- Desarrollar un *chatbot* donde los estudiantes puedan ingresar sus preguntas.

- Desarrollar sistema de gestión de archivos de texto plano (.txt, .pdf, .html) usado por el modelo RAG.
- Integrar *REST API* para permitir una comunicación eficiente entre el *frontend* y el *backend*.
- Realizar pruebas con estudiantes de la asignatura de Fundamentos de Programación.

Requerimientos no funcionales:

- La interfaz de usuario debe ser intuitiva y fácil de usar tanto para estudiantes como para profesores.
- La experiencia de usuario debe ser fluida y sin interrupciones.
- El sistema debe ser escalable para manejar un aumento de documentos indexados sin afectar el rendimiento.
- Implementación de medidas de seguridad para proteger contra accesos no autorizados.
- Uso de prácticas de desarrollo de software que aseguren la calidad y mantenibilidad del código.
- La aplicación debe ser compatible con los navegadores web modernos.

Este proyecto se estructura en cinco fases: 1. El diseño del sistema, 2. La integración del pipeline de ingesta de datos, 3. La integración del modelo de Recuperación y Generación (RAG), 4. El desarrollo del *chatbot* integrado en una aplicación web interactiva y 5. Las pruebas del sistema. La metodología se fundamenta en la necesidad de proporcionar respuestas con información precisa, clara y actualizada, además de permitir la gestión del contenido utilizado por el modelo RAG por parte del profesor de la asignatura, tal como se expuso en el capítulo 1, superando las limitaciones de los *chatbots* educativos de IA actuales en términos de calidad y actualidad de la información proporcionada.

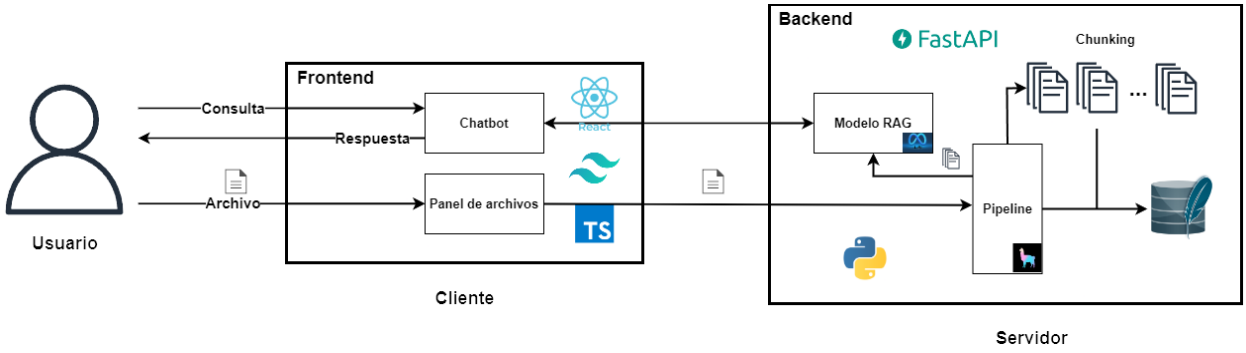
2.2 Diseño del sistema

2.2.1 Arquitectura del sistema

El sistema se desarrolló utilizando una arquitectura cliente-servidor, tal como se muestra en la Figura 1, para gestionar de manera eficiente las interacciones entre el *frontend* y el *backend*. Esta arquitectura permite una separación clara de responsabilidades, donde el *frontend* se encarga de la presentación y la interacción con el usuario, y el *backend* maneja el procesamiento de datos y la lógica del negocio.

Figura 1

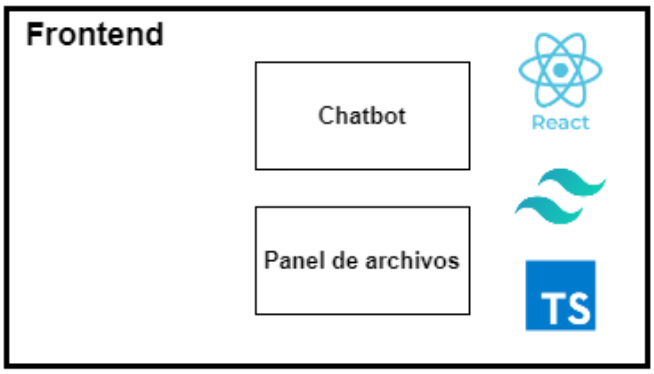
Arquitectura del sistema



El *frontend* fue desarrollado utilizando React.js, TypeScript y TailwindCSS (ver Figura 2). React.js es una biblioteca de JavaScript ideal para construir interfaces de usuario intuitivas y accesibles, permitiendo una experiencia dinámica y reactiva. TypeScript aporta tipado estático, mejorando la robustez y el mantenimiento del código. TailwindCSS facilita el diseño de interfaces modernas y adaptativas con su sistema de clases utilitarias. Además, el *frontend* integró un *chatbot* interactivo, proporcionando asistencia en tiempo real y mejorando la interacción con los usuarios. Este *chatbot* utiliza tecnologías avanzadas de procesamiento de lenguaje natural (NLP) para comprender y responder a las consultas de los estudiantes de manera eficiente y precisa. La combinación de estas tecnologías proporciona una interfaz de usuario atractiva, funcional y altamente interactiva.

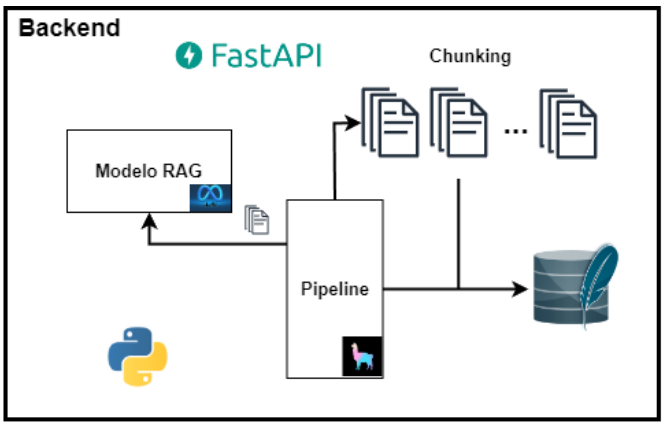
Figura 2

Diseño del frontend.



El *backend* se lo implementó con FastAPI (ver Figura 3), un *framework* de alto nivel en Python, que proporciona un entorno robusto y seguro para gestionar el servidor y la base de datos SQLite. Este *backend* también integró *REST API* para permitir la comunicación entre el *frontend* y el *backend*, mejorando así la experiencia del usuario con actualizaciones fluidas y rápidas. Por otro lado, se integró un pipeline de ingesta de datos para la extracción, estructuración y almacenamiento eficiente de documentos en PDF, HTML y texto plano, utilizando la librería LlamaIndex. Además, se integró un modelo RAG para proporcionar respuestas precisas a preguntas de programación para novatos. Este modelo emplea la similitud de coseno para recuperar información relevante y alimentar al generador de texto *open source* Llama, permitiendo así sesiones continuas y sostenidas de chat con los estudiantes. Este enfoque asegura un sistema robusto y eficiente que satisface las necesidades de nuestros usuarios.

Figura 3
Diseño del backend.

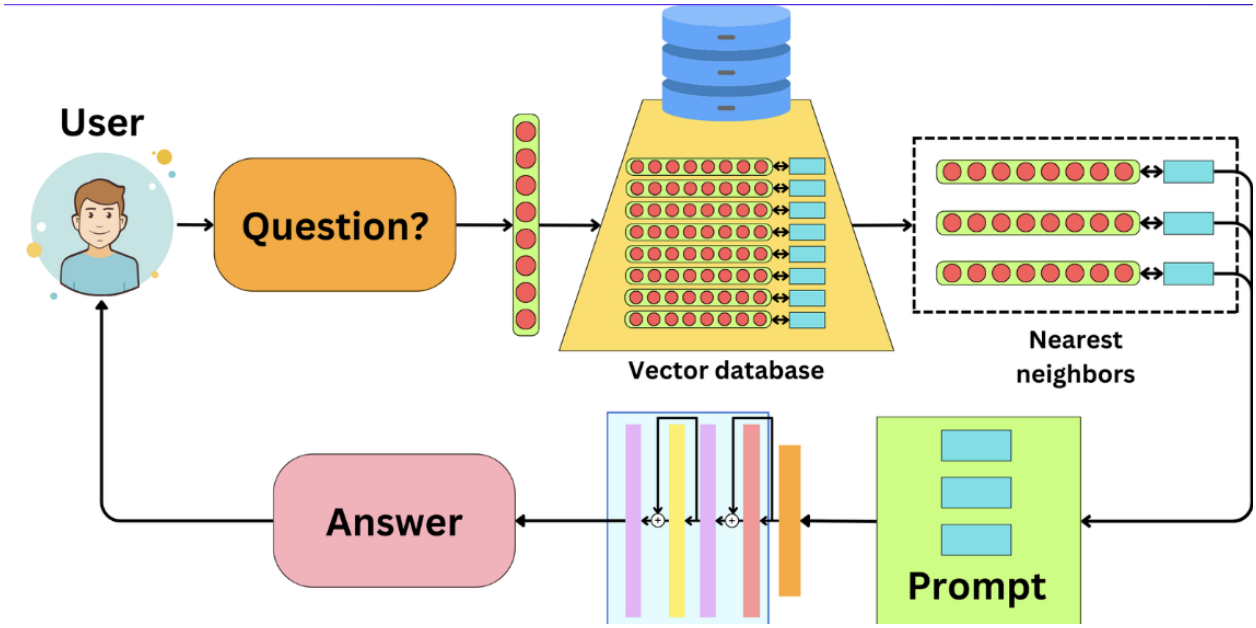


2.3 Integración del pipeline de ingesta de datos

El proceso de *chunking* e *indexing*, ver Figura 4, utiliza un modelo *embedder* y un *vector store*, se implementará para la gestión y búsqueda de grandes volúmenes de datos textuales. Aquí se detalla cómo se llevará a cabo este proceso.

Figura 4

Proceso de recuperación RAG. Obtenida de https://newsletter.theaiedge.io/p/how-to-optimize-your-rag-pipelines

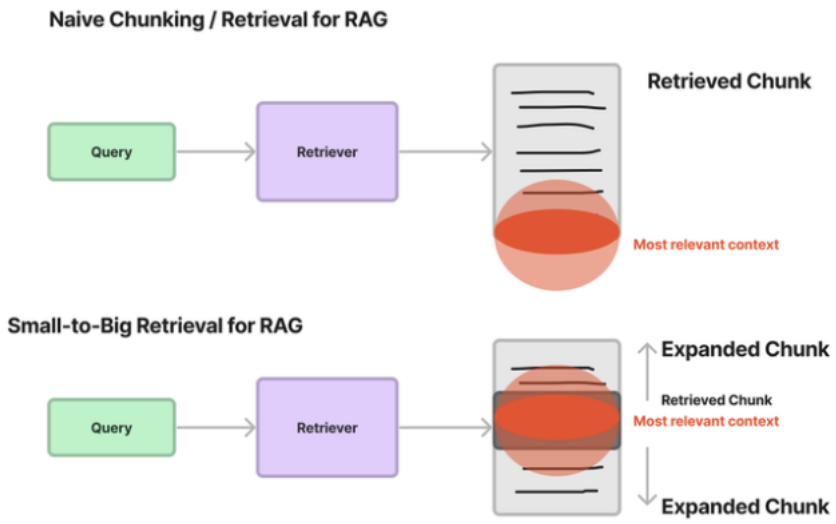


Primero, se realizará la división del documento en segmentos más pequeños, conocidos como *chunks*. Esta división se hará por párrafos, oraciones o una cantidad específica de palabras, dependiendo de la naturaleza del contenido y de los requisitos del sistema de procesamiento. Posteriormente, cada *chunk* se preprocesará para mejorar la calidad del texto, lo que incluirá la eliminación de *stop words*, lematización, tokenización y eliminación de caracteres no deseados [5].

Luego, se seleccionará un modelo *embedder*, para convertir los *chunks* de texto en *embeddings* (ver Figura 5). Estos modelos, entrenados para capturar la semántica del texto, generarán *embeddings* que representarán la información de cada *chunk* en un formato vectorial de alta dimensión.

Figura 5

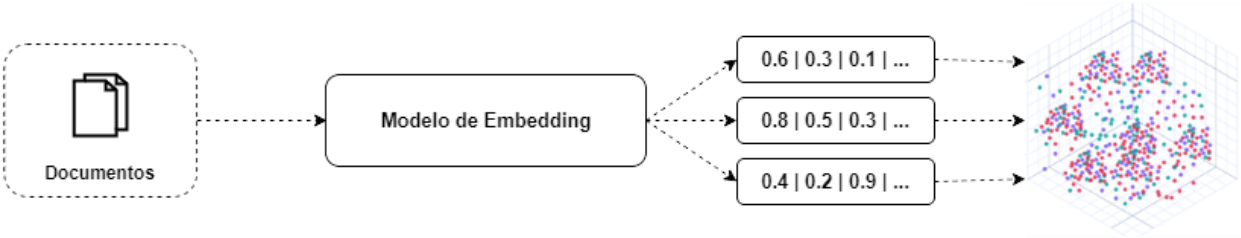
Proceso de conversión de chunk en embeddings. Obtenida de https://x.com/jerryjliu0/status/1732503009891127676



Posteriormente, se diseñará un vector store optimizado para almacenar y buscar *embeddings*, utilizando herramientas como FAISS o LlamaIndex se indexarán los *embeddings* generados (ver Figura 6), creando estructuras de datos que permitan realizar búsquedas eficientes, basadas en árboles k-dimensional, grafos de proximidad o métodos de hash.[6]

Figura 6

Proceso de embedding de documentos.



Para realizar búsquedas y recuperar información, se convertirá la consulta en un *embedding* utilizando el mismo modelo *embedder*. Este *embedding* de consulta se comparará con los *embeddings* almacenados en el *vector store*, identificando los vecinos más cercanos mediante algoritmos de búsqueda de vecinos más cercanos (*nearest neighbor search*). Finalmente, se recuperarán los *chunks* de texto correspondientes a los *embeddings* más cercanos y se presentarán al usuario, permitiendo una búsqueda rápida y precisa en grandes bases de datos textuales. [7]

2.4 Integración del modelo RAG

Para realizar aplicaciones RAG se tiene las siguiente opciones SOTA:

Tabla 1

Opciones SOTA para realizar aplicaciones RAG.

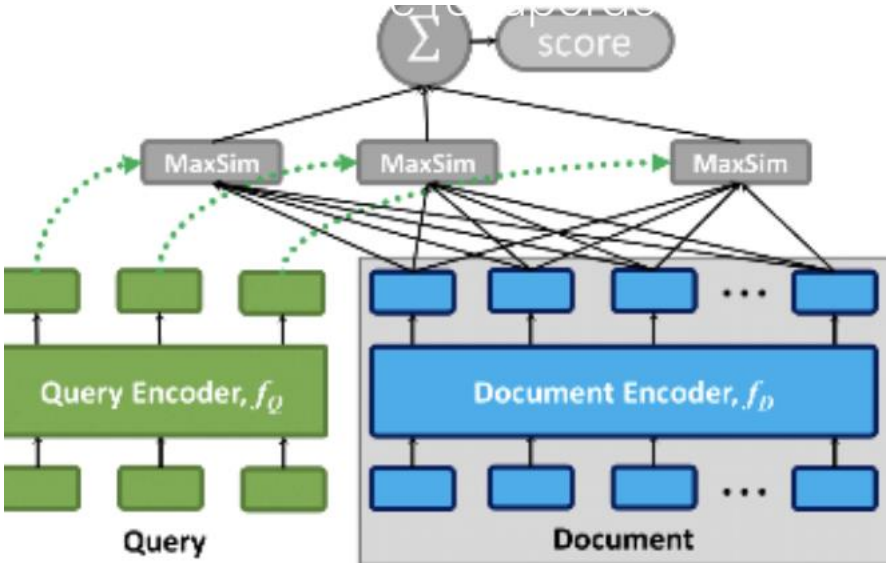
Método	Ventajas	Desventaja
Embedding con similitud de coseno	Simplicidad, velocidad, eficiencia en almacenamiento.	Precisión y escalabilidad limitadas.
Embedding con *reranker*	Mejorar de precisión y flexibilidad.	Complejidad adicional y requiere más recursos.
Usando COLBERT	Alta precisión y optimización de recursos.	Complejidad técnica y requiere recursos significativos.

Usando COLBERT con *reranker*	Máxima precisión y contextualización profunda.	Altamente intensivo en recursos y complejidad de implementación.

Se va a utilizar el modelo COLBERT (ver Figura 7), conocido por su enfoque de interacción tardía (*late-interaction*) para la recuperación de información, integrándolo a través de LlamaIndex y la biblioteca RAGatouille. COLBERT destaca por su alta eficiencia y capacidad de generalización a nuevos dominios, lo que lo convierte en una opción ideal para nuestro proyecto de tesis. Con RAGatouille, se facilitará su uso y optimización, permitiendo aprovechar sus capacidades avanzadas para mejorar el rendimiento del sistema de recuperación de información. [10]

Figura 7

Modelo COLBERT. Obtenida de https://medium.com/@varun030403/colbert-a-complete-guide-1552468335ae

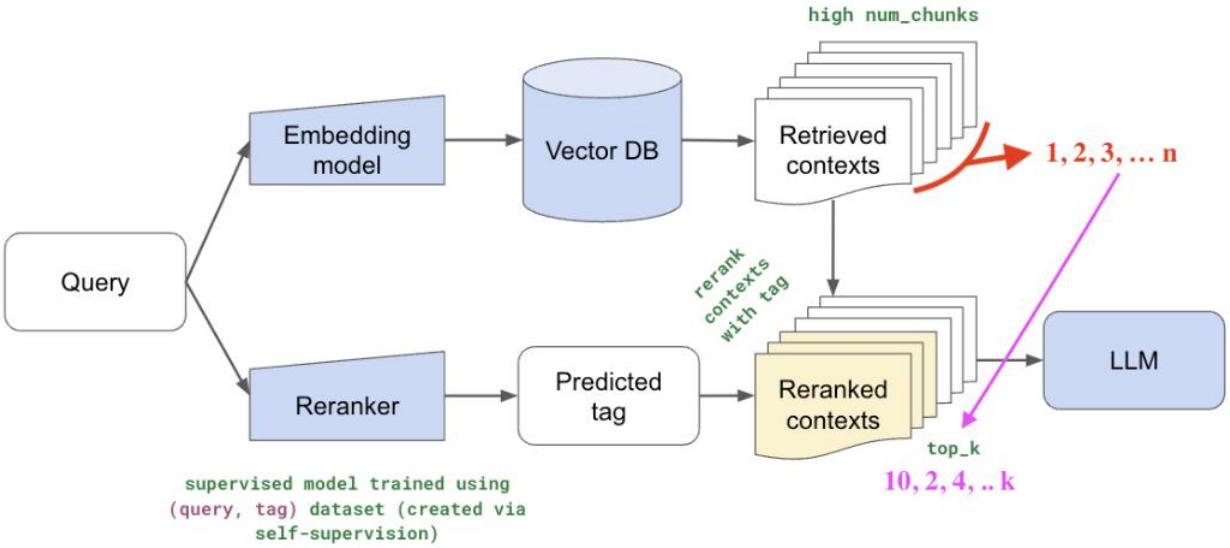


Se utilizará el modelo Ada2 como *embedder* y COLBERT como *reranker*. La importancia del *reranker* radica en su capacidad para refinar y ordenar los documentos recuperados inicialmente por el *embedder* (ver Figura 8), asegurando que los más relevantes aparezcan primero. COLBERT, con su enfoque de interacción tardía, permite una evaluación más precisa de la

relevancia de los documentos, mejorando significativamente la precisión de los resultados de búsqueda. [11]

Figura 8

Proceso de reranking. Obtenida de https://medium.com/etoai/simplest-method-to-improve-rag-pipeline-re-ranking



2.5 Desarrollo del *chatbot* integrado en una aplicación web

El objetivo de este módulo es proporcionar un *chatbot* basado en el modelo RAG e integrado a una aplicación web con una interfaz interactiva y amigable. La aplicación está diseñada para que los estudiantes puedan consultar y los profesores puedan subir contenido, que será procesado por el pipeline de ingesta de datos para usarlo con el modelo RAG. La aplicación debe proporcionar una interfaz fácil de usar tanto para estudiantes como para profesores, facilitando el acceso a información precisa y actualizada en la asignatura de Fundamentos de Programación.

Para cumplir con este objetivo, la aplicación web cuenta con funcionalidades diseñadas para optimizar la experiencia del usuario. Entre las principales funcionalidades se encuentra el panel de control para profesores, una característica esencial que permite a los profesores subir documentos en diversos formatos (PDF, HTML, y texto plano). El pipeline de ingesta de datos procesa automáticamente estos documentos, asegurando que el contenido esté estructurado y listo para usarlo por el modelo RAG. Además, el *chatbot* interactivo es una de las funcionalidades

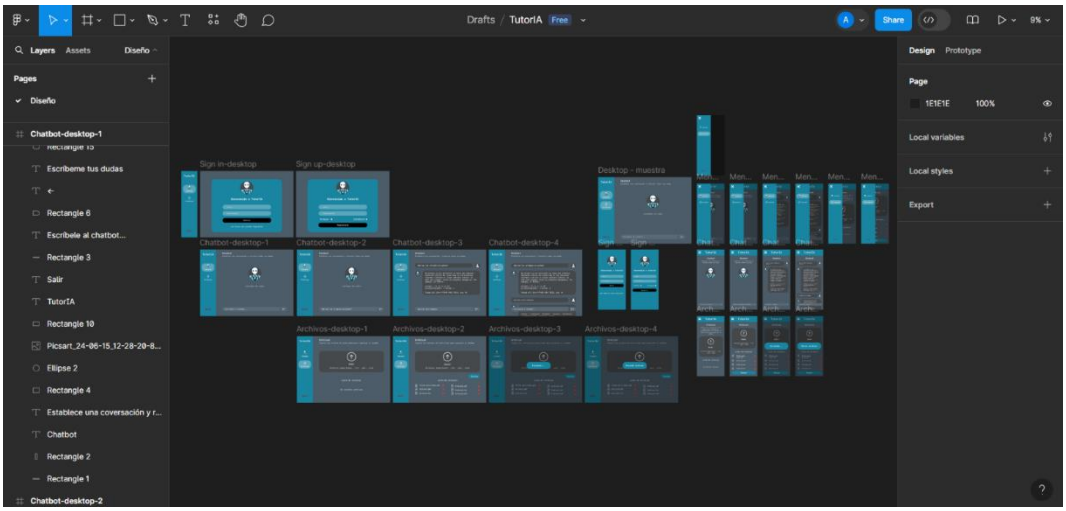
centrales, permitiendo a los estudiantes realizar consultas y obtener respuestas precisas y contextuales generadas por el modelo RAG. Esta característica es fundamental para apoyar el aprendizaje de los estudiantes, proporcionándoles acceso a respuestas claras y confiables a sus preguntas sobre programación.

2.5.1 Prototipo

Para asegurar una experiencia de usuario óptima y un diseño intuitivo, se desarrolló un prototipo de la aplicación utilizando la herramienta Figma (ver Figura 9). Este prototipo incluye versiones tanto para escritorio como para dispositivos móviles, abarcando todas las pantallas y funcionalidades necesarias.

Figura 9

Diseño del prototipo.

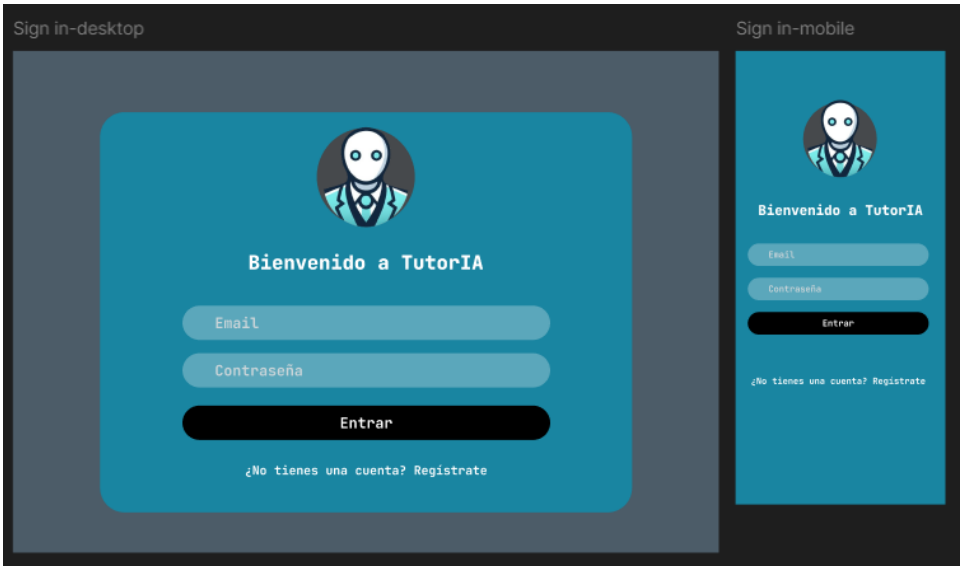


Las pantallas principales diseñadas en el prototipo incluyen:

Pantalla de entrada: Esta pantalla permite a los usuarios autenticarse en el sistema mediante sus credenciales (ver Figura 10). El diseño se centra en ser sencillo y claro, facilitando un acceso rápido y seguro a la aplicación. Se ha optimizado tanto para versiones de escritorio como para móviles, asegurando una experiencia uniforme en todos los dispositivos.

Figura 10

Diseño de la pantalla de entrada.



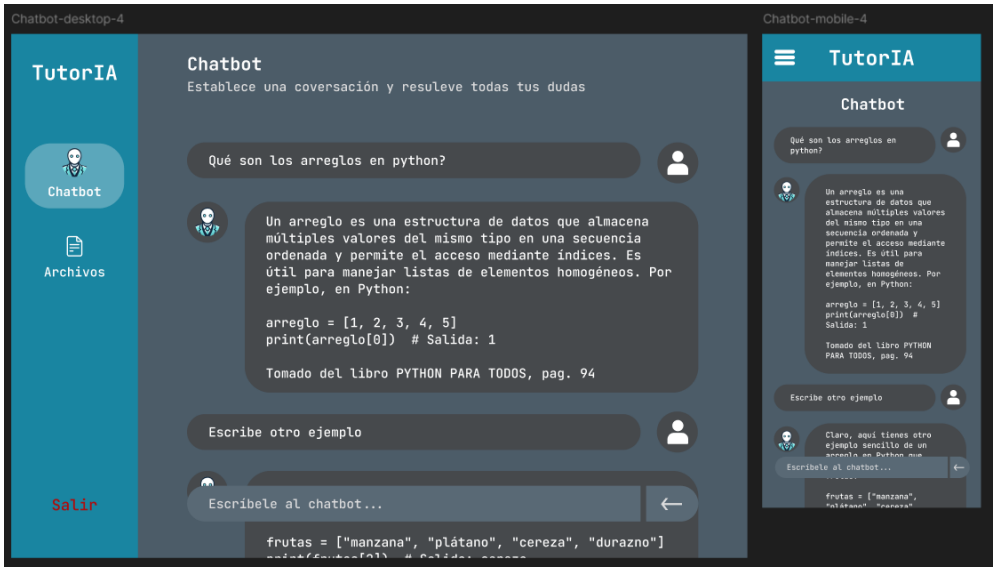
Pantalla de registro: Proporciona un formulario para que los nuevos usuarios se registren en la plataforma (ver Figura 11). Los campos necesarios incluyen correo electrónico, y contraseña. El diseño está orientado a ser amigable y accesible, reduciendo cualquier fricción que pueda experimentar el usuario durante el proceso de registro. En el prototipo, se presenta una interfaz limpia y estructurada que facilita el llenado del formulario.

Figura 11
Diseño de la pantalla de registro.

Pantalla del *Chatbot*: Esta es la interfaz principal para los estudiantes, donde pueden interactuar con el *chatbot* (ver Figura 12). El diseño incluye un área de texto para ingresar preguntas y una sección donde se muestran las respuestas generadas por el *chatbot*. La interfaz está optimizada para facilitar una conversación fluida y natural, asegurando que los estudiantes puedan encontrar la información que necesitan de manera eficiente.

Figura 12

Diseño de la pantalla del chatbot.



Pantalla de archivos (Panel de control para profesores): Diseñada solo para profesores, permite gestionar contenidos que usará el pipeline de ingesta de datos (ver Figura 13). Los profesores pueden subir, eliminar y organizar documentos relevantes que el modelo RAG utilizará para generar respuestas. Esta sección no está visible para los estudiantes, garantizando una gestión segura y controlada de los contenidos. Esta pantalla muestra un diseño estructurado y funcional, facilitando la gestión de documentos.

Figura 13

Diseño de la pantalla de archivos (panel de control).

2.5.2 Comunicación

La comunicación eficiente es un componente crucial para el funcionamiento del *chatbot*. Se integró *REST API* para soportar interacciones fluidas, permitiendo una experiencia de usuario eficiente y sin interrupciones. Esta tecnología es esencial para mantener un chat continuo y dinámico entre los estudiantes y el *chatbot*, asegurando que las consultas se gestionen de manera oportuna y eficiente. La implementación de *REST API* permitió que las respuestas del *chatbot* sean inmediatas y que las conversaciones se desarrollen sin demoras perceptibles, mejorando significativamente la experiencia del usuario.

2.5.3 Base de datos

El almacenamiento y la gestión eficiente de datos son aspectos críticos para el funcionamiento del sistema. Los contenidos de estudio y documentos relevantes se almacenaron en una base de datos SQLite, lo que permitió una gestión eficiente y una rápida integración para su uso por el modelo RAG. SQLite es una opción ideal para este proyecto debido a su naturaleza ligera y su capacidad para manejar datos estructurados de manera eficiente. Este enfoque aseguró que la información esté siempre disponible y actualizada, mejorando la calidad de las respuestas

generadas por el *chatbot*. Además, el sistema de almacenamiento está diseñado para ser escalable, permitiendo futuras expansiones y actualizaciones según sea necesario.

2.5.4 Implementación del diseño en desarrollo

El desarrollo de la aplicación web siguió el diseño establecido en el prototipo de Figma. Se utilizaron las tecnologías antes mencionadas (React.js para el *frontend* y FastAPI para el *backend*) para implementar la funcionalidad del *chatbot*, las interfaces de usuario y el panel de control para profesores. La sincronización de la base de datos con el pipeline de ingesta de datos y el modelo RAG permitió una integración fluida y eficiente, garantizando que el sistema funcione según lo planeado. Cada componente del sistema será desarrollado y probado minuciosamente para asegurar su funcionalidad y rendimiento óptimo.

2.6 Pruebas del sistema

El objetivo de este módulo es realizar pruebas del sistema con estudiantes de la asignatura de Fundamentos de Programación para asegurar su funcionalidad, efectividad y facilidad de uso. Las pruebas fueron diseñadas para evaluar el rendimiento del *chatbot* y la aplicación web en un entorno real, ajustando y optimizando el sistema basado en los resultados de las pruebas y el *feedback* recibido. Se buscó garantizar que el sistema cumpla con sus objetivos, proporcionando respuestas precisas y útiles a las consultas de los estudiantes, y facilitando la gestión de contenidos por parte de los profesores.

Para realizar una evaluación exhaustiva del sistema, se implementaron dos tipos de pruebas. Estas pruebas incluyeron:

- Pruebas de Usabilidad: Evaluaron la facilidad de uso del sistema, asegurando que los estudiantes puedan navegar por la plataforma y utilizar el *chatbot* sin dificultades. Los aspectos evaluados incluyeron la claridad de la interfaz, la facilidad de acceso a las diferentes funcionalidades y la eficiencia en la interacción con el *chatbot*.

- Pruebas de Funcionalidad: Evaluaron que todas las funcionalidades del sistema, desde la autenticación de usuarios hasta la gestión de contenidos por parte de los profesores, funcionen correctamente. Estas pruebas se centraron en verificar que el pipeline de ingesta de datos, el modelo RAG y el *chatbot* operen de manera coordinada y eficiente.

2.6.1 Proceso de pruebas

1. Se seleccionaron estudiantes y un profesor de la asignatura de Fundamentos de Programación para participar en las pruebas. Se incluyó un grupo de estudiantes de distintas carreras para asegurar una evaluación representativa del sistema.
2. Se diseñaron escenarios de uso realistas que incluyan una variedad de consultas y tareas típicas que los estudiantes podrían realizar utilizando el *chatbot*. Estos escenarios ayudaron a evaluar la capacidad del *chatbot* para proporcionar respuestas precisas y útiles en diferentes contextos.
3. Los estudiantes utilizaron la aplicación web para interactuar con el *chatbot* y responder a una serie de preguntas preparadas. Durante esta fase, se recogieron datos sobre la precisión de las respuestas del *chatbot* y la experiencia general del usuario.
4. Al finalizar las pruebas, se solicitó a los participantes que proporcionen retroalimentación detallada sobre su experiencia. Esto incluyó aspectos como la facilidad de uso de la interfaz, la utilidad de las respuestas proporcionadas por el *chatbot* y cualquier problema encontrado durante el uso del sistema.
5. Los datos recopilados durante las pruebas se analizaron para identificar áreas de mejora y confirmar que el sistema cumple con sus objetivos. Se prestó especial atención a la precisión y relevancia de las respuestas del *chatbot*, así como a la usabilidad general de la aplicación.

2.6.2 Pruebas de usabilidad

Estas pruebas se enfocaron en cómo los estudiantes interactúan con la aplicación web y el *chatbot*. Se utilizaron los siguientes métodos:

- Ejercicios de programación: Los estudiantes recibieron tareas que debieron resolver utilizando el *chatbot*. Por ejemplo, se les pidió que escriban un programa que utilice Numpy, y deberán usar el *chatbot* para obtener ayuda con el algoritmo adecuado.
- Encuestas de satisfacción: Después de realizar los ejercicios, se solicitó a los estudiantes que completen una encuesta de satisfacción. Esta encuesta incluyó preguntas con una escala de Likert de 5 puntos para medir la facilidad de uso de la interfaz, la claridad de las respuestas del *chatbot*, y la eficiencia general del sistema.

  Ejemplos de preguntas:

  "¿Qué tan fácil fue navegar por la aplicación web?" (1 = Muy difícil, 5 = Muy fácil)

  "¿Qué tan útiles fueron las respuestas del *chatbot*?" (1 = Nada útiles, 5 = Muy útiles)

  "¿Qué tan estable fue la aplicación durante su uso?" (1 = Muy inestable, 5 = Muy estable)
- Observación directa: Durante las sesiones de prueba, se observó a los estudiantes mientras interactúan con la aplicación web y el *chatbot* para identificar problemas de usabilidad y áreas que requieran mejoras. Se registraron comentarios y comportamientos que indiquen confusión o dificultad.

2.6.3 Pruebas de funcionalidad

Estas pruebas aseguraron que todas las características del sistema funcionen correctamente:

- Escenarios de prueba: Se crearon escenarios de prueba que reflejen situaciones de uso real. Los estudiantes y el profesor realizaron tareas como autenticación de

usuario, subida de documentos, y consultas al *chatbot*. Cada función del sistema fue probada para verificar su correcto funcionamiento.

- Registro de incidencias: Durante la ejecución de estas pruebas, se registraron cualquier fallo o comportamiento inesperado del sistema. Esto incluyó problemas con la autenticación, fallos en la ingesta de datos, y errores en las respuestas del *chatbot*.
- Verificación de funcionalidades principal: Se probaron funciones específicas como la actualización de contenidos por parte de los profesores, la recuperación de información por el modelo RAG y la generación de respuestas por el *chatbot*, asegurando que cada componente funcione de acuerdo con las especificaciones.

Según los resultados alcanzados y la retroalimentación de los usuarios, se realizaron los ajustes necesarios para mejorar el sistema. Esto incluyó mejoras en la interfaz de usuario, optimización del rendimiento del *chatbot* y ajustes en el modelo RAG para aumentar la precisión de las respuestas. El objetivo es asegurar que el sistema final sea robusto, eficiente y capaz de satisfacer las necesidades de los usuarios.

Capítulo 3

3. Resultados y análisis

En este capítulo se presentan los resultados obtenidos del desarrollo del chatbot de IA integrado a una aplicación web, así como el análisis de su desempeño y la evaluación de su efectividad en el apoyo a los estudiantes de Fundamentos de Programación. La sección se organiza en los siguientes apartados: resultados del diseño del sistema, del pipeline de ingesta de datos, del modelo RAG, del desarrollo del chatbot y de los resultados del sistema con los estudiantes de fundamentos de programación.

3.1 Resultados del diseño del sistema

El diseño del sistema se completó con éxito, logrando una arquitectura que cumple con los requerimientos funcionales y no funcionales definidos en el capítulo anterior. La arquitectura del sistema incluye un *frontend* intuitivo desarrollado con React.js, TypeScript y TailwindCSS, y un *backend* robusto construido con FastAPI y SQLite para la base de datos. La interacción entre los componentes del sistema fue fluida, garantizando la calidad del software.

La arquitectura cliente-servidor adoptada para este proyecto proporcionó una separación clara de responsabilidades, donde el *frontend* se encargaba de la presentación y la interacción con el usuario, y el *backend* manejaba el procesamiento de datos y la lógica del negocio. Esta separación facilitó el desarrollo modular y la mantenibilidad del sistema.

El *frontend* fue desarrollado utilizando React.js, TypeScript y TailwindCSS, lo que permitió la creación de una interfaz de usuario (ver Figura 14) intuitiva y accesible. Las pruebas de usabilidad realizadas con un grupo de estudiantes y profesores indicaron que la interfaz era fácil de usar y facilitaba una experiencia de usuario positiva.

Figura 14

Resultado del desarrollo del frontend del sistema.

Los estudiantes encontraron que el *chatbot* integrado en el *frontend* era una herramienta útil para obtener respuestas rápidas a sus preguntas sobre Fundamentos de Programación. La tecnología de procesamiento de lenguaje natural (NLP) utilizada por el *chatbot* permitió una comprensión precisa de las consultas de los estudiantes y proporcionó respuestas relevantes de manera eficiente.

El *backend*, implementado con FastAPI y SQLite, demostró ser robusto y seguro. Las pruebas de rendimiento realizadas en el *backend* mostraron que el sistema podía manejar correctamente los archivos y solicitudes simultáneas sin problemas significativos. Además, para enviar los archivos al sistema, se utilizaron las rutas de la API del *backend*, detalladas en la Tabla 2, que facilitaban la comunicación con el pipeline de ingesta de datos. Este pipeline, integrado

utilizando la librería LlamaIndex, permitió la extracción, estructuración y almacenamiento eficiente de documentos en PDF, HTML y texto plano. Esta funcionalidad fue crucial para asegurar que el modelo RAG tuviera acceso a datos relevantes y bien estructurados para proporcionar respuestas precisas a las consultas de los estudiantes.

Tabla 2

Rutas de API para conectarse al backend

Método	Ruta	Descripción
POST	/auth/register	Permite el registro de los usuarios en el sistema.
POST	/auth/token	Permite el inicio de sesión de los usuarios en el sistema.
POST	/api/chat	Permite realizar consultas al modelo RAG.
POST	/api/datasource/upload	Permite el envío de archivos al modelo RAG.
GET	/api/datasource/files/<datasource>	Permite obtener todos los nombres de los archivos enviados al modelo RAG.
GET	/auth/verify_role?required_role=<ROLE>	Verifica el rol de los usuarios.

3.2 Resultados del pipeline de ingesta de datos

El pipeline de ingesta de datos, implementado utilizando la librería LlamaIndex, demostró ser eficaz en la extracción, estructuración y almacenamiento de documentos en formatos PDF, HTML y texto plano. Los resultados obtenidos en esta fase fueron cruciales para el funcionamiento óptimo del modelo RAG y, por ende, para la precisión de las respuestas del *chatbot*.

El proceso de *chunking* y *indexing* se llevó a cabo con éxito, dividiendo los documentos en segmentos más pequeños (*chunks*) y procesándolos para mejorar la calidad del texto. Esta etapa incluyó la eliminación de *stop words*, lematización, *tokenización* y eliminación de caracteres no deseados, lo que resultó en una mejora significativa en la calidad de los datos ingresados al sistema.

Se utilizó el modelo *embedder* Ada2 para convertir los *chunks* de texto en *embeddings*, capturando eficazmente la semántica del texto. Estos *embeddings* se almacenaron en un *vector store* optimizado, permitiendo búsquedas eficientes basadas en la similitud de coseno.

3.3 Resultados del modelo RAG

El modelo de Recuperación y Generación (RAG) implementado en este proyecto demostró ser efectivo en la provisión de respuestas precisas y relevantes a las consultas de los estudiantes.

El uso de COLBERT como *reranker* mejoró significativamente la precisión de los resultados de búsqueda. Este enfoque permitió una evaluación más precisa de la relevancia de los documentos recuperados, asegurando que las respuestas generadas por el *chatbot* fueran las más apropiadas y útiles para los estudiantes.

El modelo es lo suficientemente rápido como para servir a una clase de 30 estudiantes sin utilizar paralelismo en el *querying*, sin embargo, se recomienda para una instancia futura el hacerlo.

Los resultados de las pruebas con los estudiantes validaron la efectividad del modelo RAG:

- El 56.5% de los estudiantes obtuvieron calificaciones de Excelente (9-10 puntos) en los ejercicios resueltos con la ayuda del *chatbot*.
- El 34.8% obtuvieron calificaciones de Muy Bueno (7-8 puntos).

- Solo el 8.7% alcanzaron una calificación de Suficiente (5-6 puntos).

Estos resultados indican que el modelo RAG fue capaz de proporcionar información precisa y relevante, permitiendo a los estudiantes resolver eficazmente los ejercicios propuestos.

Además, la capacidad del modelo para manejar consultas de estudiantes con diferentes niveles de habilidades de programación (desde básico hasta intermedio) demuestra su versatilidad y adaptabilidad. El modelo RAG pudo proporcionar respuestas adecuadas tanto para conceptos fundamentales como para temas más avanzados, adaptándose a las necesidades individuales de los estudiantes.

3.4 Resultados del desarrollo del chatbot integrado en una aplicación web

El desarrollo del *chatbot* se centró en proporcionar una interfaz intuitiva, sencilla y accesible, logrando así una experiencia de usuario altamente satisfactoria. Utilizando tecnologías como React.js, TypeScript y TailwindCSS, se creó una aplicación web que permite a los usuarios interactuar fácilmente con el sistema, tanto en dispositivos desktop como móviles.

La elección de desarrollar una aplicación web fue estratégica, con el objetivo de maximizar la accesibilidad. Al ser una aplicación web, cualquier usuario con un navegador moderno puede acceder al sistema sin necesidad de instalar software adicional. Esto no solo facilita el acceso desde diversos dispositivos, incluidos computadores de escritorio, laptops, tabletas y teléfonos inteligentes, sino que también asegura que los estudiantes y profesores puedan utilizar el *chatbot* sin restricciones geográficas o de hardware. La compatibilidad multidispositivo garantiza que la herramienta esté siempre disponible, aumentando su utilidad y alcance.

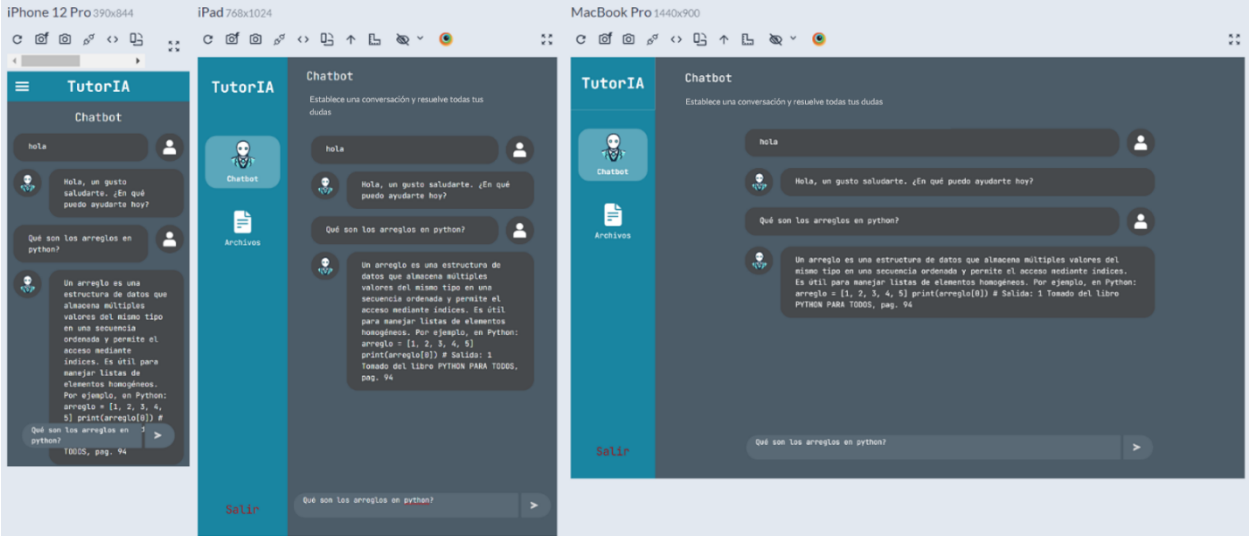
3.4.1 Chatbot

El desarrollo del *chatbot* se llevó a cabo utilizando las tecnologías mencionadas previamente: React.js, TypeScript y TailwindCSS. Como resultado (ver Figura 15), se logró una interfaz minimalista, moderna y sencilla, basada fielmente en el diseño inicial propuesto en Figma. Esta implementación no solo cumple con los estándares visuales de modernidad y simplicidad,

sino que también asegura una versión responsive que se adapta a dispositivos desktop, tablet y móvil. La funcionalidad del *chatbot* permite establecer conversaciones de manera similar a los *chatbots* de IA actuales y modernos, ofreciendo una experiencia de usuario intuitiva y agradable.

Figura 15

Pantallas del chatbot en diferentes dispositivos.

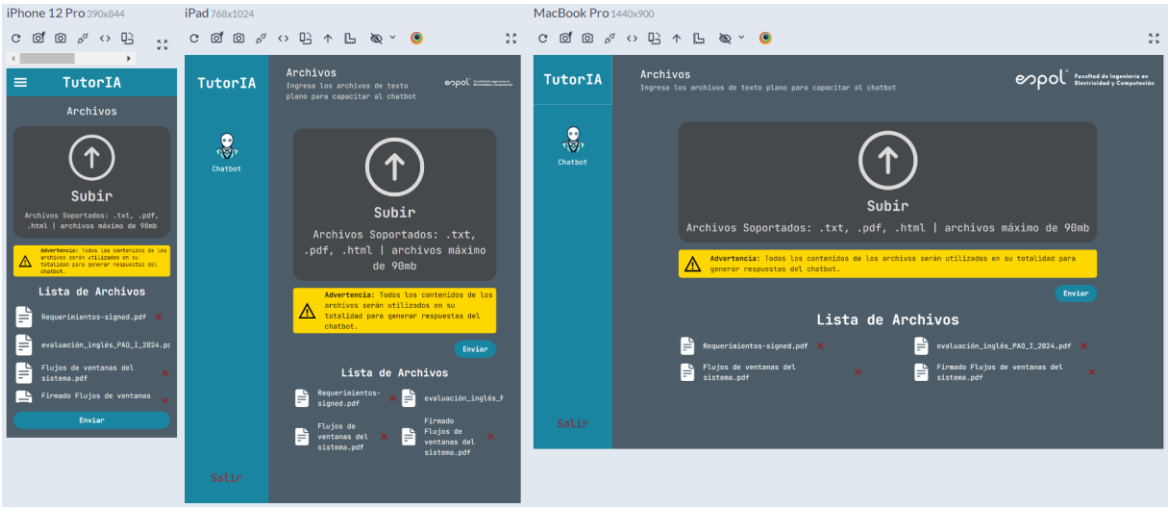


3.4.2 Panel de archivos

El panel de archivos, que actúa como el panel de control para los profesores (ver Figura 16), refleja el mismo diseño moderno y minimalista del *chatbot*, manteniendo la fidelidad del diseño planteado en Figma y asegurando un buen renderizado y facilidad de uso. Este panel permite a los profesores visualizar, subir, eliminar y gestionar múltiples archivos, ya sea haciendo clic o simplemente arrastrando y soltando los archivos en el área designada. Una vez cargados, los archivos pueden visualizarse, lo que facilita la gestión y actualización del contenido educativo de manera eficiente y sin complicaciones que utilizará después el modelo RAG. Además, está diseñado para ser responsive, asegurando su funcionalidad en dispositivos desktop, tablet y móvil. Esta accesibilidad multidispositivo permite a los profesores gestionar el contenido desde cualquier lugar y en cualquier momento, facilitando la actualización constante de los materiales educativos.

Figura 16

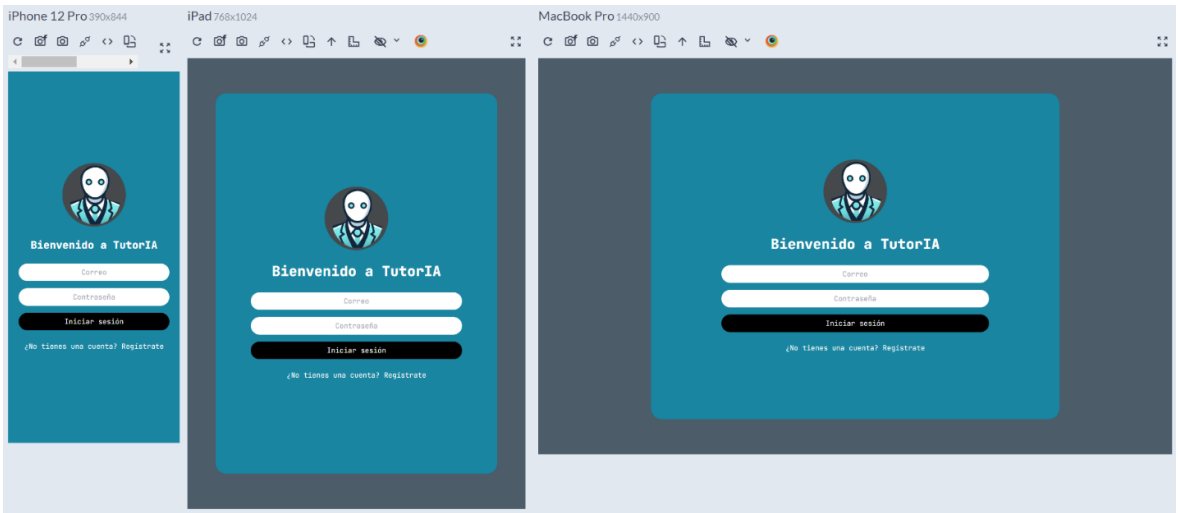
Pantallas del panel de archivos en diferentes dispositivos.



3.4.3 Resultados de las pantallas de login y registro

La pantalla de *login*, que se muestra en la Figura 17, se diseñó para ser sencilla y clara, facilitando un acceso rápido y seguro a la aplicación. Optimizada para versiones desktop, tablets y móviles, esta pantalla permite a los usuarios autenticarse en el sistema mediante sus credenciales. Las pruebas mostraron que los usuarios encontraron el proceso de *login* intuitivo y eficiente. Además, el *login* es necesario para controlar el acceso al panel de archivos, asegurando que solo los profesores puedan utilizar esta funcionalidad crítica.

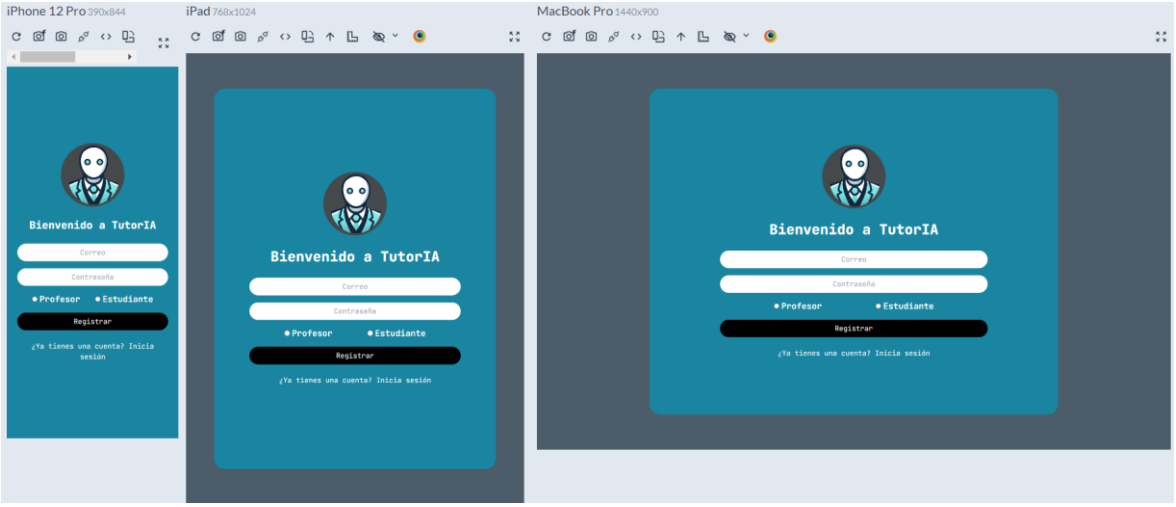
Figura 17
Pantallas del login en diferentes dispositivos.



La pantalla de registro (ver Figura 18) proporciona un formulario para que los nuevos usuarios se registren en la plataforma. El diseño está orientado a ser amigable y accesible, con una interfaz limpia y estructurada que facilita el llenado del formulario. Las pruebas de usabilidad indicaron que los usuarios encontraron el proceso de registro simple y directo.

Figura 18

Pantallas del registro en diferentes dispositivos.



3.4.4 Resultados de las pruebas del sistema

Las pruebas del sistema se llevaron a cabo en un entorno real en una clase presencial de Fundamentos de Programación, con la autorización del profesor del curso. Durante esta sesión, el profesor permitió que los estudiantes utilizaran el *chatbot* para resolver ejercicios relacionados con la unidad de Numpy para trabajar con arreglos. Esta actividad permitió evaluar la funcionalidad del sistema en un contexto práctico y obtener retroalimentación directa de los estudiantes.

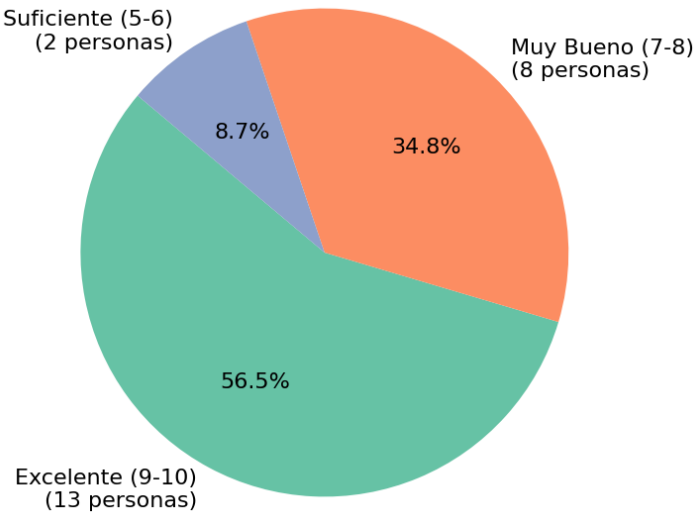
Se explicó el objetivo del proyecto a los estudiantes, quienes luego completaron un formulario de evaluación diseñado para recolectar datos sobre la experiencia de uso. El formulario incluía las siguientes secciones:

- Preguntas Demográficas: Para obtener información sobre el perfil de los estudiantes que realizaron las pruebas.
- Escala de Likert: Para medir el grado de acuerdo con diferentes afirmaciones relacionadas con la funcionalidad y usabilidad del sistema.
- Preguntas Abiertas: Para obtener comentarios detallados y sugerencias sobre el uso del *chatbot* y el panel de archivos.
- Opciones Múltiples: Para evaluar las preferencias y experiencias de los estudiantes con respecto a las funcionalidades de la aplicación.

En términos generales, los estudiantes obtuvieron buenos resultados utilizando el *chatbot* para resolver los ejercicios propuestos. Como se muestra en la Figura 19, la mayoría de los estudiantes alcanzaron calificaciones destacadas. El 56.5% obtuvieron calificaciones de Excelente (9-10 puntos), el 34.8% obtuvieron Muy Bueno (7-8 puntos), y el 8.7% lograron una calificación de Suficiente (5-6 puntos). Esto sugiere que el *chatbot* cumplió eficazmente su objetivo de asistir a los estudiantes en la resolución de ejercicios, proporcionando respuestas útiles y precisas que contribuyeron a un buen desempeño en las tareas.

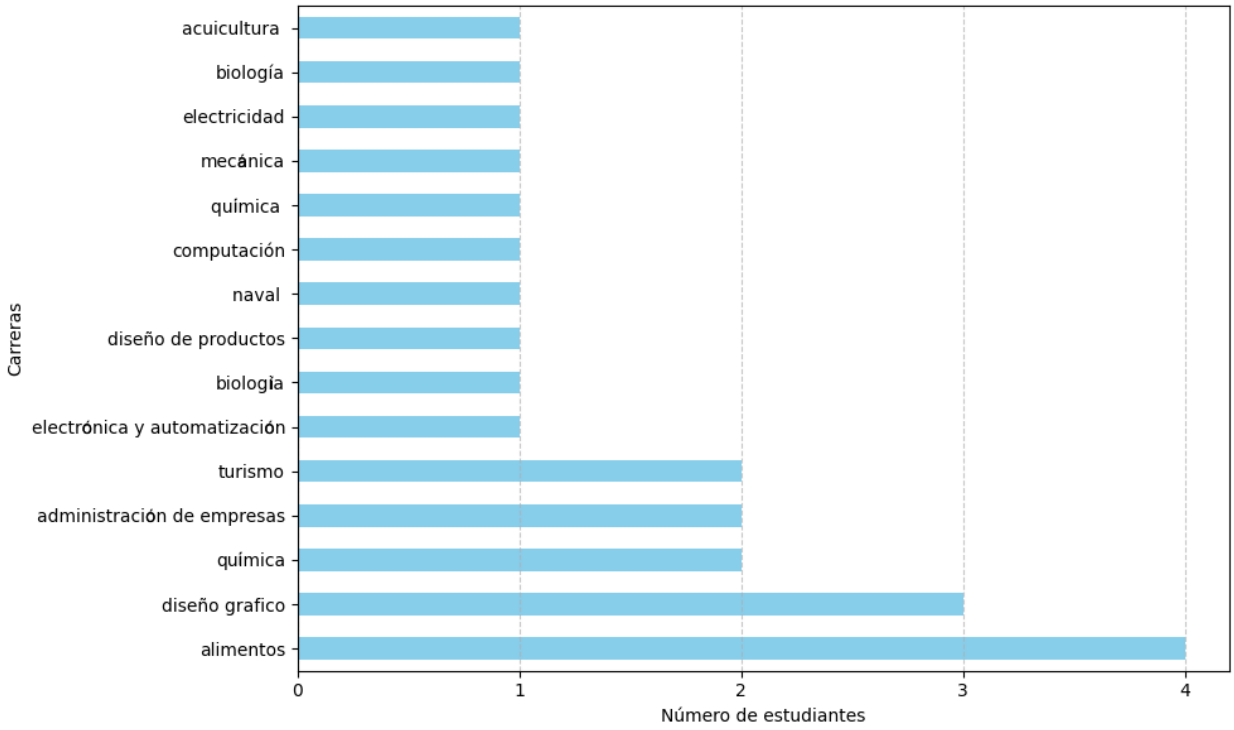
Figura 19

Resultados de los ejercicios resueltos usando el chatbot.



Adentrándonos en los resultados demográficos, se observa una diversidad en las carreras de los estudiantes participantes. La Figura 20 muestra que la carrera de Ingeniería en Alimentos tiene la mayor representación con 4 estudiantes. Le sigue Diseño Gráfico con 3 estudiantes, mientras que Turismo, Ingeniería Química y Administración de Empresas tienen 2 estudiantes cada una. En contraste, Computación está representada por solo 1 estudiante.

Figura 20
Distribución de las carreras de los estudiantes.



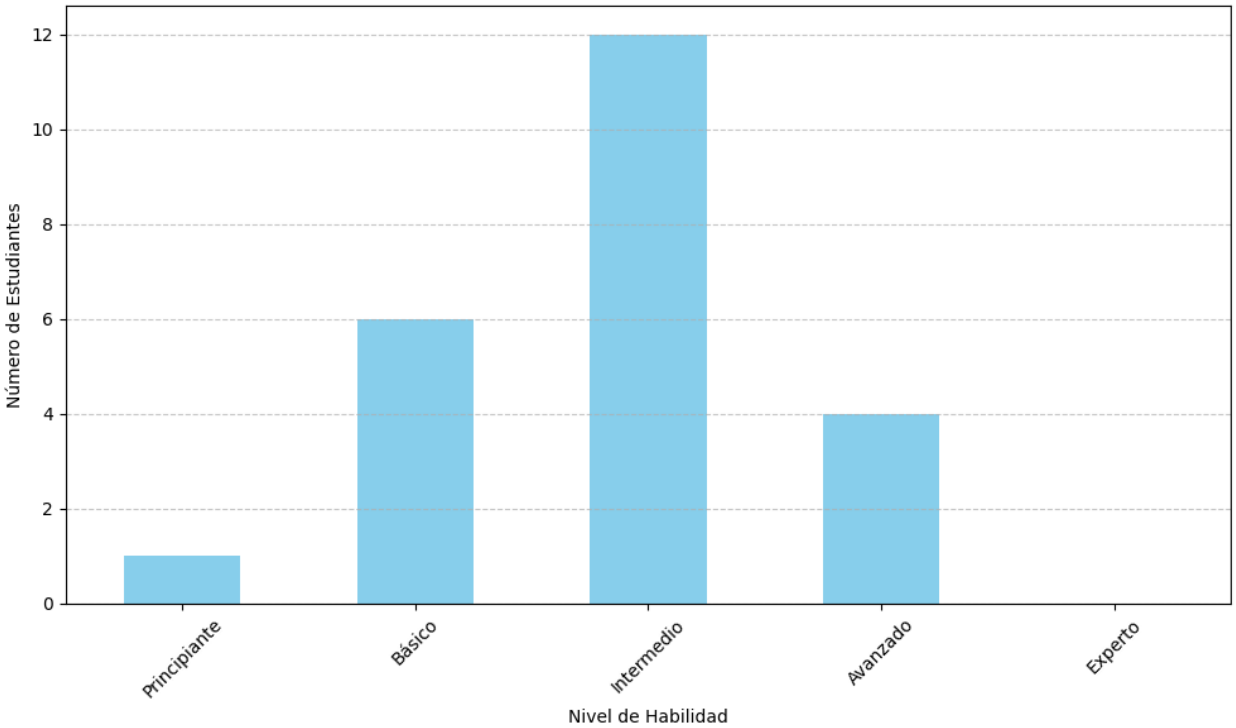
Esta distribución resalta que la mayoría de los participantes provienen de áreas no relacionadas con la programación, lo que valida el enfoque del *chatbot* en asistir a estudiantes de disciplinas no centradas en la programación. Este resultado es coherente con el objetivo del proyecto de proporcionar apoyo a estudiantes de diversas áreas, especialmente aquellos que pueden no estar familiarizados con la programación. La presencia de solo un estudiante de Computación refuerza la idea de que los resultados obtenidos son objetivos. El *chatbot* está

diseñado para ser una herramienta útil para aquellos con menos experiencia tecnológica, cumpliendo así su propósito de facilitar el aprendizaje en áreas no tecnológicas.

En términos de habilidades de programación, se puede observar en la Figura 21 que la mayoría de los estudiantes consideran que tienen habilidades intermedias (12 estudiantes) y básicas (6 estudiantes). Este perfil fue ideal para el *chatbot*, ya que permitió asistir a estudiantes con conocimientos en programación, especialmente en temas nuevos. La evaluación de los ejercicios propuestos, que trataba un tema nuevo de programación, reforzó los resultados generales obtenidos por los estudiantes. La capacidad del *chatbot* para proporcionar apoyo a estudiantes con habilidades intermedias y básicas valida su rol como herramienta efectiva para enfrentar desafíos educativos en programación.

Figura 21

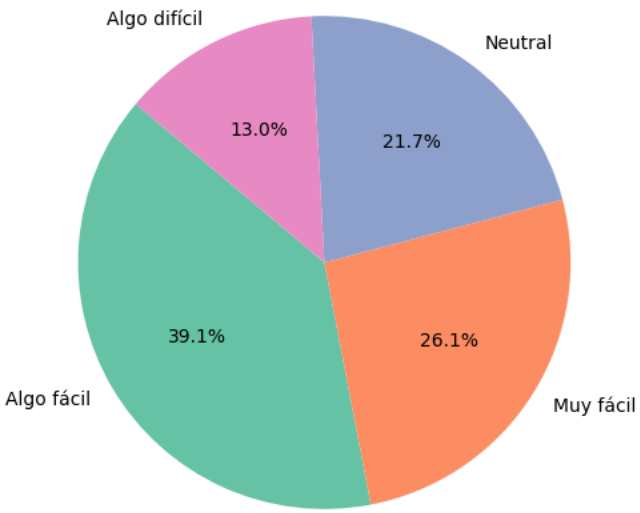
Nivel de habilidades de programación de los estudiantes.



Ahora entrando a la usabilidad del *chatbot*, los resultados muestran una recepción mayoritariamente positiva (ver Figura 22). El 39.1% de los estudiantes consideró que la

interacción con el *chatbot* fue "Algo fácil", seguido del 26.1% que lo encontró "Muy fácil". El 21.6% de los estudiantes mantuvo una postura "Neutra" respecto a la facilidad de uso, mientras que el 13% opinó que la interacción era "Algo difícil".

Figura 22
Resultado de la interacción del chatbot con los estudiantes.



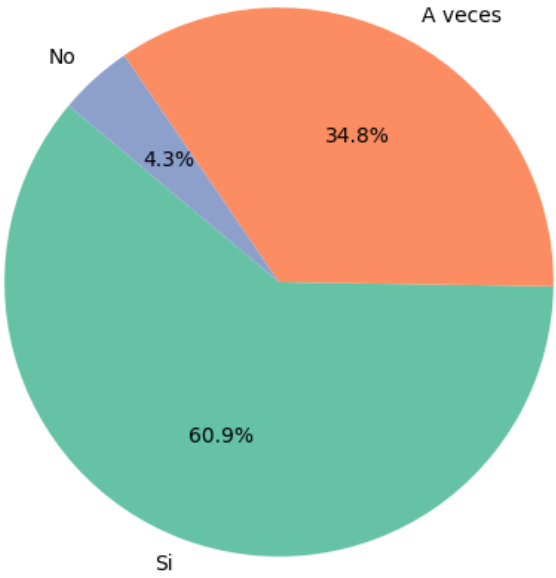
El hallazgo más significativo es que la mayoría de los estudiantes percibió la interacción con el *chatbot* como una experiencia algo fácil o muy fácil, lo que refleja un alto nivel de satisfacción y sugiere que el diseño y la funcionalidad del *chatbot* cumplen con las expectativas de los estudiantes. Este resultado es un testimonio del éxito en la creación de una interfaz intuitiva y accesible que facilita una comunicación fluida entre los estudiantes y el sistema. Sin embargo, es importante tener en cuenta que el 13% de los usuarios encontró la interacción "algo difícil". Este porcentaje, aunque menor, indica que aún se debe mejorar la experiencia del usuario.

En cuanto a la interfaz del *chatbot*, los resultados reflejan una percepción positiva por parte de los estudiantes. La figura 23 muestra que la mayoría, un 60.9%, seleccionó que la interfaz era intuitiva y fácil de usar, lo que subraya el éxito en el diseño de una plataforma accesible para los usuarios. Además, el 34.8% de los estudiantes mencionó que la interfaz era a veces intuitiva, lo

que sugiere que, en la mayoría de los casos, la navegación y la interacción con el sistema fueron satisfactorias. Solo un 4.3% de los estudiantes consideró que la interfaz no era intuitiva, lo que representa una minoría y refuerza la efectividad general de la interfaz de usuario del *chatbot*.

Figura 23

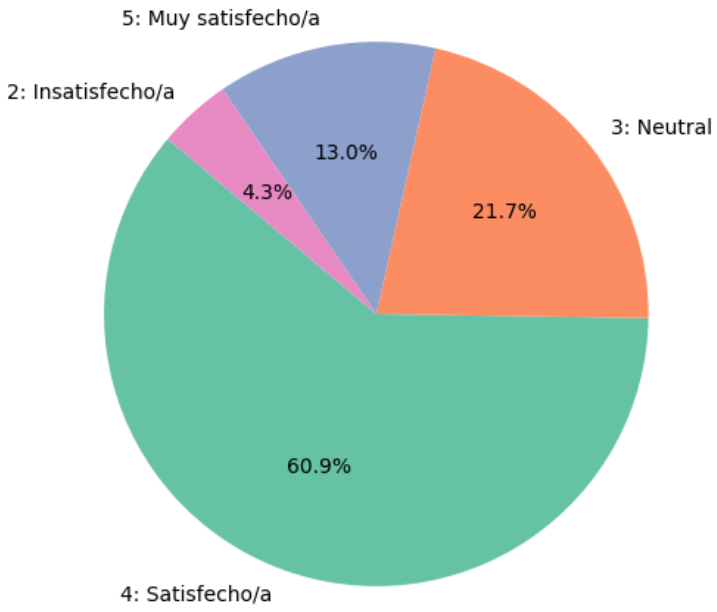
Resultado de la interfaz de usuario del chatbot.



Siguiendo con la satisfacción del uso del *chatbot* (Ver Figura 24), el 60.9% de los estudiantes indicaron que se encuentran satisfechos, seguido del 21.7% que se mostró neutral, el 13% que se declaró muy satisfecho y solo el 4.3% manifestó estar insatisfecho. Estos resultados destacan una recepción positiva generalizada del *chatbot*, con un porcentaje significativo de estudiantes satisfechos o muy satisfechos. Aunque una pequeña fracción de estudiantes expresó insatisfacción, la mayoría de los usuarios considera que el *chatbot* cumple con sus expectativas, lo que respalda su utilidad como herramienta educativa en el aula.

Figura 24

Sentimientos de los estudiantes al usar el chatbot.



En términos de funcionalidad, la mayoría de los estudiantes mencionaron que ninguna funcionalidad les resultó confusa. Sin embargo, tres estudiantes indicaron que, al llegar a cierto nivel de interacción, el *chatbot* comenzaba a perder el hilo de la conversación, lo que afectaba la coherencia en las respuestas. Además, se mencionó que las respuestas en pseudocódigo no eran tan eficientes para el aprendizaje y que la velocidad de generación de respuestas debería ser más rápida. Asimismo, se sugirió que las respuestas a preguntas sencillas deberían ser más concisas y no muy extensas para evitar sobrecargar al usuario con información innecesaria.

A pesar de estas observaciones, los estudiantes también destacaron aspectos positivos en el *chatbot*. La Figura 25, que muestra una nube de palabras, revela que los aspectos más útiles que encontraron en el *chatbot* incluyen el nivel muy bueno de explicación, las respuestas claras, la formulación de preguntas efectivas, el código proporcionado, la ayuda ofrecida y los ejemplos presentados, entre otros. Estos elementos contribuyen a una experiencia de usuario positiva y refuerzan la efectividad del *chatbot* como herramienta educativa.

Figura 25

Resultado de los aspectos más importantes del chatbot.

Finalmente, los estudiantes han señalado que el *chatbot* si tiene el potencial de mejorar la productividad en el desarrollo de tareas. Destacaron la ayuda a las dudas, explicación, las respuestas proporcionadas, y los ejemplos ofrecidos. Estos aspectos se reflejan en la Figura 26, una nube de palabras que resalta los términos más frecuentemente mencionados por los estudiantes.

Figura 26

Resultado de las consideraciones de los estudiantes al usar el chatbot.

La implementación del *chatbot* en el entorno educativo ha demostrado ser una herramienta valiosa para mejorar la productividad en el desarrollo de tareas de programación. La mayoría de los estudiantes destacaron que el *chatbot* no solo facilitó la resolución de dudas y la obtención de respuestas, sino que también contribuyó significativamente a su desempeño en los ejercicios. La recepción positiva generalizada, combinada con la retroalimentación constructiva, proporciona una base sólida para continuar perfeccionando la herramienta y ajustarla para satisfacer aún mejor las necesidades de los usuarios. El *chatbot* ha mostrado ser una solución efectiva para apoyar a los estudiantes en su aprendizaje de la programación, y las mejoras continuas prometen potenciar aún más su eficacia y utilidad en futuros escenarios educativos. Los resultados obtenidos durante las pruebas del sistema, realizadas en un entorno educativo real, reflejan el grado de cumplimiento de los objetivos propuestos al inicio del proyecto.

3.4.5 Requerimientos de hardware

El mayor costo en términos de FLOPS (operaciones de punto flotante por segundo) en el sistema se presenta durante la consulta al modelo RAG y la inferencia del modelo generativo. Para evaluar la factibilidad del despliegue y optimizar los parámetros, se probaron varios modelos. El objetivo era realizar consultas de manera eficiente, lo que llevó a un enfoque centrado en la GPU utilizada.

Se establecieron dos configuraciones de modelo: Llama3-8b y Phi3-3.8b. A continuación, se presenta la Tabla 3 que detalla los costos de memoria en diferentes precisiones, así como la velocidad de inferencia de cada configuración.

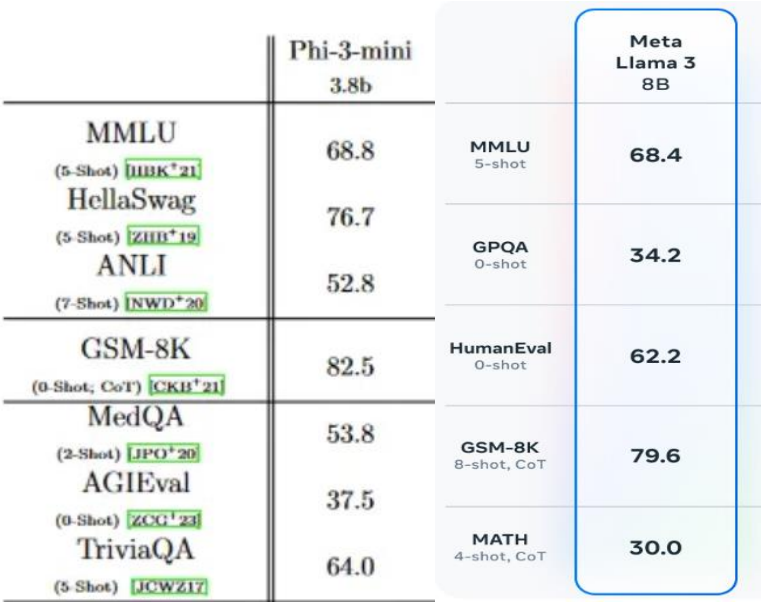
Tabla 3

Tabla comparativa de modelos a elegir para implementar el sistema.

		Llama3-8B	Phi3-3.8B
Costo en memoria En GB	fp16	14.5	6.9
	Fp8 (Hopper+)	6.9	3.5
Tokens por segundo (en promedio, bs=1)	fp16	36	77
	fp8 (Hopper+)	83	164

Figura 2277

Evaluaciones de modelos. Fuente: Guia de llama3 y de phi3 en https://ai.meta.com/blog/meta-llama-3/



Para ambos modelos, Llama3-8b y Phi3-3.8b, es esencial utilizar GPUs basadas en la arquitectura Hopper o posterior, ver la Figura 27, para garantizar una experiencia rápida. Estas GPUs cuentan con un motor de cálculo en FP8, lo que proporciona una ventaja significativa en términos de velocidad y *throughput* durante la generación. Entre las opciones recomendadas se incluyen la RTX 4090, L40, L40S y L4.

Las GPUs de la generación anterior, Ada Lovelace, también son una opción viable; sin embargo, al usarlas, el modelo requerirá mucha más memoria y generará a velocidades considerablemente más lentas.

Es crucial, independientemente de la GPU elegida, contar con al menos 24 GB de RAM para poder procesar en lotes y aprovechar al máximo las capacidades de procesamiento en paralelo. Además, se recomienda utilizar una CPU con al menos 8 núcleos y 32 GB de RAM para asegurar un rendimiento eficiente del sistema.

Capítulo 4

4.1 Conclusiones y recomendaciones

En este capítulo se presentan las conclusiones derivadas del desarrollo y evaluación del *chatbot* educativo. Además, se incluyen recomendaciones para futuras mejoras y ampliaciones del sistema, con el fin de maximizar su impacto y utilidad en varias asignaturas de la malla curricular.

4.1.1 Conclusiones

- La implementación del *chatbot* basado en IA e integrado en una plataforma web, demostró ser capaz de asistir eficazmente a los estudiantes en la asignatura de Fundamentos de Programación, proporcionando un apoyo educativo efectivo, ofreciendo respuestas claras y precisas a las consultas de los estudiantes en el ámbito de Fundamentos de Programación.
- El diseño de la arquitectura del sistema, basado en una estructura cliente-servidor con un *frontend* en React.js y un *backend* en FastAPI, demostró ser efectivo en garantizar la calidad del software. La separación clara de responsabilidades entre *frontend* y *backend* facilitó la modularidad, escalabilidad y mantenibilidad del sistema, cumpliendo así con el primer objetivo específico.
- La implementación del pipeline para la ingesta de datos utilizando la librería LlamaIndex, demostró indexar eficientemente documentos de diversas fuentes (PDF, TXT, texto plano), estructurando la información accesible para su recuperación y análisis por el modelo RAG. Esto cumple con el segundo objetivo específico y contribuye significativamente a la precisión y relevancia de las respuestas del *chatbot*.
- El desarrollo del *frontend* utilizando React.js, TypeScript y TailwindCSS resultó en una interfaz de usuario intuitiva y accesible. Las pruebas de usabilidad indicaron una alta satisfacción de los estudiantes, con la facilidad de uso de la plataforma.

4.1.2 Recomendaciones

En función de los resultados obtenidos y las experiencias durante el desarrollo del proyecto, se proponen las siguientes recomendaciones para trabajos futuros y mejoras de la plataforma:

- Implementar un escalado horizontal para mejorar el rendimiento del *chatbot* y del sistema en general. Esto permitirá manejar una mayor cantidad de estudiantes simultáneamente y mejorar la velocidad de respuesta, garantizando una experiencia más fluida y eficiente para los usuarios.
- Integrar funcionalidades que permitan monitorear el progreso de los estudiantes a lo largo del tiempo, proporcionando informes detallados que puedan ser utilizados por los profesores para identificar áreas de mejora.
- Desarrollar algoritmos que adapten las respuestas del *chatbot* al nivel de conocimiento y estilo de aprendizaje de cada estudiante, ofreciendo una experiencia más personalizada y efectiva.
- Desarrollar funcionalidades que permitan a los estudiantes compartir y discutir las respuestas del *chatbot* entre ellos podría fomentar un aprendizaje colaborativo más efectivo.
- Investigar la viabilidad de adaptar el sistema para su uso en otras asignaturas más allá de Fundamentos de Programación, lo que podría requerir ajustes en el modelo RAG y en la estructura de ingesta de datos.
- Aplicar los últimos avances en modelos de lenguaje y técnicas de recuperación de información para mejorar constantemente la precisión y relevancia de las respuestas del *chatbot*.

Referencias

[1] N. Ahmad, S. Murugesan and N. Kshetri, "Generative Artificial Intelligence and the Education Sector," in Computer, vol. 56, no. 6, pp. 72-76, June 2023, doi: 10.1109/MC.2023.3263576.

[2] L. Labadze, M. Grigolia & L. Machaidze, "Role of AI chatbots in education: A systematic literature review," Int J Educ Technol High Educ 20, 56 (2023).

[3] A. S. Sreelakshmi, S. B. Abhinaya, A. Nair and S. Jaya Nirmala, "A Question Answering and Quiz Generation Chatbot for Education," 2019 Grace Hopper Celebration India (GHCI), Bangalore, India, 2019, pp. 1-6, doi: 10.1109/GHCI47972.2019.9071832.

[4] C. Kooli, "Chatbots in Education and Research: A Critical Examination of Ethical Implications and Solutions," Sustainability 2023, 15, 5614. https://doi.org/10.3390/su15075614

[5] A. Martínez Cenalmor, "Impacto de Chat GPT en el entorno educativo: posibilidades y riesgos," Repositorio Institucional de la Universidad de Oviedo, 2023, uri: http://hdl.handle.net/10651/69004

[6] F. Colace, M. De Santo, M. Lombardi, F. Pascale, A. Pietrosanto, S Lemma, "Chatbot for E-Learning: A Case of Study," International Journal of Mechanical Engineering and Robotics Research. Vol. 7, No. 5, September 2018.

[7] S. Tadvi, S. Rangari, A. Rohe, "HR Based Interactive Chat bot (PowerBot)," University of Exeter, 2020, pp. 1-6, doi: 10.1109/ICCSEA49143.2020.9132917.

[8] K. Wu, E. Wu, and J. Zou, "How faithful are rag models? quantifying the tug-of-war between Rag and llms' internal prior," arXiv.org, https://arxiv.org/abs/2404.10198 (accessed Jun. 3, 2024).

[9] Y. Huang and J. Huang, “A survey on retrieval-augmented text generation for large language models,” arXiv.org, https://arxiv.org/abs/2404.10981v1 (accessed Jun. 3, 2024).

[10] Y. Gao et al., “Retrieval-augmented generation for large language models: A survey,” arXiv.org, https://arxiv.org/abs/2312.10997 (accessed Jun. 3, 2024).

[11] Clarizia, F., Colace, F., Lombardi, M., Pascale, F., & Santaniello, D. (2018). Chatbot: An education support system for student. In Cyberspace Safety and Security: 10th International Symposium, CSS 2018, Amalfi, Italy, October 29–31, 2018, Proceedings 10 (pp. 291-302). Springer International Publishing.