UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE SOFTWARE

TEMA:

"DESARROLLO DE UN CHATBOT PARA LA GESTIÓN DEL PORTAFOLIO ESTUDIANTIL BASADO EN UN MODELO DE LENGUAJE DE GRAN ESCALA (LLM) UTILIZANDO SCRUM Y LA ISO/IEC 25022"

TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SOFTWARE

AUTOR

Brayan Orlando de la Cruz Espinosa

DIRECTOR:

Ing. Iván Danilo García Santillán, Ph.D

Ibarra-Ecuador

2026

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	040175122-7		
APELLIDOS Y NOMBRES:	DE LA CRUZ ESPINOSA BRAYAN ORLANDO		
DIRECCIÓN:	TULCÁN – CARCHI, BARRIO PARQUE ARTESANAL, MANZANA 14 – CASA 4		
EMAIL:	bodelacruze@utn.edu.ec		
TELÉFONO FIJO:		**TELF. MOVIL**	0988188838

DATOS DE LA OBRA	
TÍTULO:	DESARROLLO DE UN CHATBOT PARA LA GESTIÓN DEL PORTAFOLIO ESTUDIANTIL BASADO EN UN MODELO DE LENGUAJE DE GRAN ESCALA (LLM) UTILIZANDO SCRUM Y LA ISO/IEC 25022.
AUTOR (ES):	DE LA CRUZ ESPINOSA BRAYAN ORLANDO
FECHA DE APROBACIÓN: AAAAMMDD	07/01/2026
CARRERA/PROGRAMA:	■ PREGRADO ☐ POSTGRADO
TITULO POR EL QUE OPTA:	INGENIERO EN SOFTWARE
ASESOR/DIRECTOR:	PhD. GARCÍA SANTILLÁN IVÁN DANILO

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de esta y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 7 días, del mes de enero de 2026

EL AUTOR:

BRAYAN ORLANDO DE LA CRUZ ESPINOSA

Firmado digitalmente por BRAYAN ORLANDO DE LA CRUZ ESPINOSA
Fecha: 2026.01.07 11:15:17 -05'00'

Brayan Orlando de la Cruz Espinosa

CI: 0401751227

CERTFICACIÓN DEL DIRECTOR

Ibarra, 07 de enero de 2026

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Por medio del presente, yo **PhD. Iván Danilo García Santillán**, certifico que el Sr. **de la Cruz Espinosa Brayan Orlando**, portador de la cédula de ciudadanía número **0401751227**, ha trabajado en el desarrollo del proyecto de grado titulado **"Desarrollo de un chatbot para la gestión del portafolio estudiantil basado en un modelo de lenguaje de gran escala (LLM) utilizando SCRUM y la ISO/IEC 25022"**, previo a la obtención del Título de Ingeniero en Software. Este trabajo se ha realizado con responsabilidad, lo cual certifico con honor a la verdad.

Atentamente:

IVAN DANILO GARCIA SANTILLAN

Digitally signed by IVAN DANILO GARCIA SANTILLAN
Date: 2026.01.07 11:30:08 -05'00'

……

PhD. Iván Danilo García Santillán
C.C.: 1002292603
DIRECTOR DEL TRABAJO DE GRADO

DEDICATORIA

A mi madre, Aida Espinosa, por ser el pilar fundamental de mi vida y la base de todos los valores que me han guiado en este camino. Su fortaleza, amor incondicional y ejemplo han sido la inspiración más grande en cada paso de mi formación.

A mis hermanos mayores, Christian y Fernando, gracias a su apoyo, por dejarme en cada una de sus acciones las enseñanzas para poder solventar problemas en mi vida diaria, brindándome siempre la motivación necesaria para seguir adelante.

Y a mi hermano menor, David, que este trabajo y mi propia trayectoria le sirvan de ejemplo para que siga adelante en el camino que le espera.

- **Brayan Orlando de la Cruz Espinosa**

AGRADECIMIENTO

En primer lugar, deseo expresar mi más profundo agradecimiento a mi madre, Aida Espinosa, por su apoyo incondicional, su amor sin límites y por haber forjado en mí un carácter firme que me ha permitido superar cada obstáculo. Gracias por enseñarme que no existe reto demasiado grande cuando se enfrenta con determinación y valores.

A mis hermanos, que siempre han sido un apoyo firme en mi vida. A Christian, por estar a mi lado en todo momento de la universidad, con palabras, gestos, y apoyo que me ayudaron a llegar hasta este extremo del trayecto universitario. Y a Fernando, que en mi época colegial me dio la guía y el empuje necesarios para crecer, tanto en lo académico como en lo personal.

Expreso también mi sincera gratitud a mi director de tesis, Iván, cuya guía y conocimientos en el campo de la Inteligencia Artificial fueron fundamentales para encaminar el presente estudio y desarrollo. De igual manera, agradezco a mi asesor, Antonio Quiña, por su constante revisión, observaciones y consejos que permitieron llevar este trabajo por el camino correcto.

Extiendo mi agradecimiento al director del Departamento de Tecnologías, Jorge Caraguay, y a todo su equipo, por su disposición y colaboración para que los objetivos de este proyecto se pudieran concretar.

Finalmente, expreso mi gratitud a todos mis docentes universitarios, quienes con su enseñanza, paciencia y dedicación contribuyeron significativamente a mi formación profesional y personal.

- **Brayan Orlando de la Cruz Espinosa**

LISTA DE SIGLAS

AIML: Artificial Intelligence Markup Language (Lenguaje de marcado de inteligencia artificial).

API: Application Programming Interface (Interfaz de Programación de Aplicaciones).

CPU: Central Processing Unit (Unidad Central de Procesamiento).

CRISP-DM: Cross Industry Standard Process for Data Mining (Proceso Estándar de Minería de Datos a Través de la Industria).

CUDA: Compute Unified Device Architecture (Arquitectura Unificada de Cómputo en Dispositivo, desarrollada por NVIDIA).

GPU: Graphics Processing Unit (Unidad de Procesamiento Gráfico).

IA: Inteligencia Artificial.

ISO/IEC 25022: Norma de la serie SQuaRE para la evaluación de calidad en uso de la familia 25000.

JSON: JavaScript Object Notation (Es un formato ligero basado en texto).

LLM: Large Language Model (Modelo de Lenguaje de Gran Escala).

MIT: Massachusetts Institute of Technology (Instituto de tecnología de Masachusetts).

NLP: Natural Language Processing (Procesamiento de Lenguaje Natural).

NPS: Net Promoter Score (Índice de Promotores Netos).

ODS: Objetivos de Desarrollo Sostenible.

PDF: Portable Document Format (Formato de documento portatil).

RAG: Retrieval-Augmented Generation (Generación Aumentada por Recuperación).

SCRUM: Metodología ágil de desarrollo de software.

SQL: Structured Query Language (Lenguaje de consulta estructurada).

SUS: System Usability Scale (Escala de Usabilidad del Sistema).

UTN: Universidad Técnica del Norte.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo aborda el desarrollo de un chatbot para la gestión del portafolio estudiantil de la Universidad Técnica del Norte, fundamentado en la necesidad de mejorar el acceso a la información institucional y reducir la saturación de consultas administrativas. El objetivo principal fue construir un sistema basado en un modelo de lenguaje de gran escala (LLM) bajo la metodología ágil SCRUM, complementada con CRISP-DM para el manejo de datos, y evaluado mediante la norma ISO/IEC 25022. El sistema se diseñó con una arquitectura modular que integra un agente RAG soportado en ChromaDB, un agente SQL conectado a la base de datos institucional y un agente supervisor, orquestados mediante FastAPI y desplegados con Docker. La investigación combinó enfoques cualitativos y cuantitativos, incluyendo entrevistas, ingesta documental, sprints iterativos y la aplicación de métricas de eficacia, eficiencia y satisfacción en pruebas prácticas con estudiantes. Los resultados muestran que el chatbot alcanzó una calidad en uso satisfactoria del 82,1%, con altos niveles de eficacia y satisfacción, y áreas de mejora en eficiencia. Se evidenció que la integración de LLMs con recuperación documental reduce costos de mantenimiento y ofrece interacciones más naturales frente a enfoques tradicionales. Se concluye que la solución propuesta tiene potencial de consolidarse como soporte confiable para los estudiantes, siempre que se mantenga una gobernanza documental y se optimice la latencia bajo escenarios de mayor escala.

Palabras clave: Chatbot, LLM, RAG, SCRUM, ISO/IEC 25022, Portafolio estudiantil.

ABSTRACT

This research addresses the development of a chatbot for managing the student portfolio at Universidad Técnica del Norte, grounded on the need to improve access to institutional information and reduce the overload of administrative inquiries. The main objective was to build a system based on a Large Language Model (LLM) using the agile SCRUM methodology, complemented by CRISP-DM for data handling, and evaluated through the ISO/IEC 25022 standard. The system was designed with a modular architecture integrating a RAG agent supported by ChromaDB, an SQL agent connected to the institutional database, and a supervisor agent, orchestrated via FastAPI and deployed with Docker. The study combined qualitative and quantitative approaches, including interviews, document ingestion, iterative sprints, and the application of ISO/IEC 25022 metrics of effectiveness, efficiency, and satisfaction in practical workshops with students. The results indicated that the chatbot reached a quality-in-use score of 82.1%. It showed high levels of effectiveness and user satisfaction but also highlighted some aspects where efficiency could be improved. These findings suggest that the integration of LLMs with retrieval-based architecture not only reduces maintenance costs but also enables more natural interactions when compared to traditional intent- or flow-based systems.

In conclusion, the proposed solution demonstrates promising potential as a reliable tool to support students, if document management practices and latency optimization are properly addressed in large-scale contexts.

Keywords: Chatbot, LLM, RAG, SCRUM, ISO/IEC 25022, Student portfolio.

INTRODUCCIÓN

1.1 Planteamiento del problema

El portafolio estudiantil es una herramienta fundamental para los estudiantes de la Universidad Técnica del Norte [1]. Muchos de ellos enfrentan dificultades al utilizar la plataforma debido al desconocimiento sobre el contenido y propósito de las secciones. Esta falta de información genera errores frecuentes, como la carga incorrecta de documentos, bloqueos de acceso y procesos incompletos, afectando el desempeño académico, y desaprovechando los recursos que ofrece la universidad y el estado. Cuando los estudiantes buscan información sobre algún trámite deben recurrir a la página oficial [2], la cual presenta contenido disperso, dificultando el acceso a la información específica, dejando como solución el acudir al personal administrativo y a veces saturando su limitada disponibilidad en consultas repetitivas.

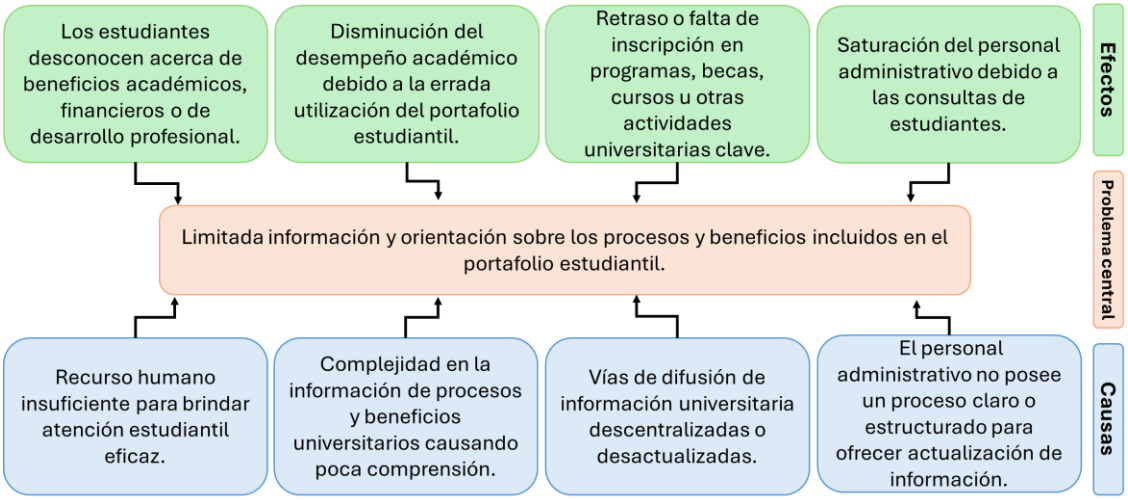


Fig. 1 Árbol de problemas

Fuente: Autoría propia

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar un chatbot para la gestión de portafolio estudiantil de la Universidad Técnica del Norte basado en un modelo de lenguaje de gran escala (LLM) utilizando el marco de trabajo SCRUM y la norma ISO/IEC 25022.

1.2.2 Objetivos Específicos

- Establecer un marco teórico sobre chatbots para la gestión de un Portafolio Estudiantil.

- Desarrollar un chatbot para la gestión de portafolio estudiantil utilizando el marco de trabajo SCRUM y el lenguaje Python.
- Evaluar la calidad en uso del chatbot diseñado para la gestión del portafolio estudiantil, tomando como referencia los lineamientos establecidos en la norma ISO/IEC 25022.

1.3 Alcance

El presente trabajo se enmarca en la investigación y desarrollo de un chatbot para la gestión de portafolio estudiantil de la Universidad Técnica del Norte, en dónde se emplearán diversas tecnologías avanzadas en procesamiento de lenguaje natural (NLP) y recuperación de información. Se realizará un análisis de los diversos tipos y características de los modelos de lenguaje (LLM) más utilizados actualmente (benchmarking) y elegir el más eficiente para las necesidades del proyecto.

El chatbot se construirá sobre un sistema de RAG (retrieval-augmented generation). Para ello se utilizará la base de datos vectorial ChromaDB, en conjunto con un modelo de IA. La idea es que este modelo genere representaciones numéricas (embeddings) de los documentos, es decir, vectores obtenidos a partir de archivos de texto plano. A su vez, el sistema tendrá acceso a una base de datos universitaria (Oracle) para brindar información en tiempo real gracias a consultas generadas por el LLM a partir de lenguaje natural, como se muestra en la Figura 2.

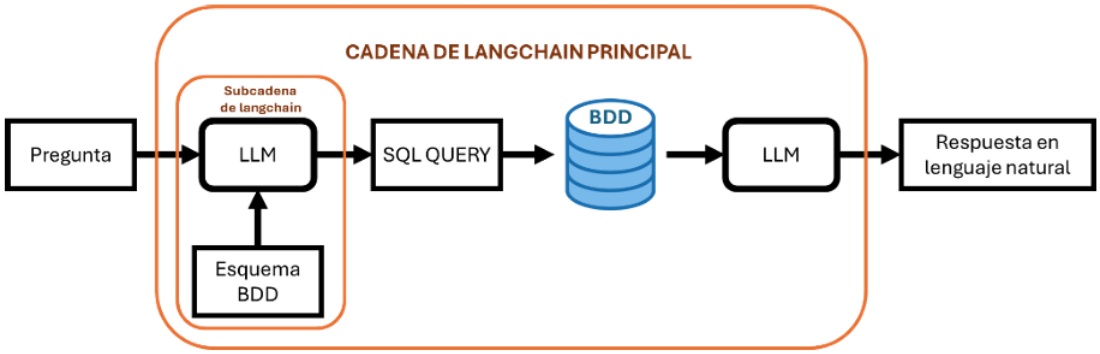


Fig. 2 Diagrama de consultas a la base de datos en lenguaje natural

Fuente: Autoría propia

Tanto el sistema RAG como la estructura de consultas trabajarán en conjunto con un modelo LLM a través de Ollama para ofrecer respuestas precisas y robustas, como se muestra en la Figura 3. Ollama es una plataforma y herramienta diseñada para facilitar la ejecución, despliegue y uso de modelos de lenguaje en dispositivos locales, especialmente optimizada para aprovechar el hardware local del usuario, como GPUs y CPUs.

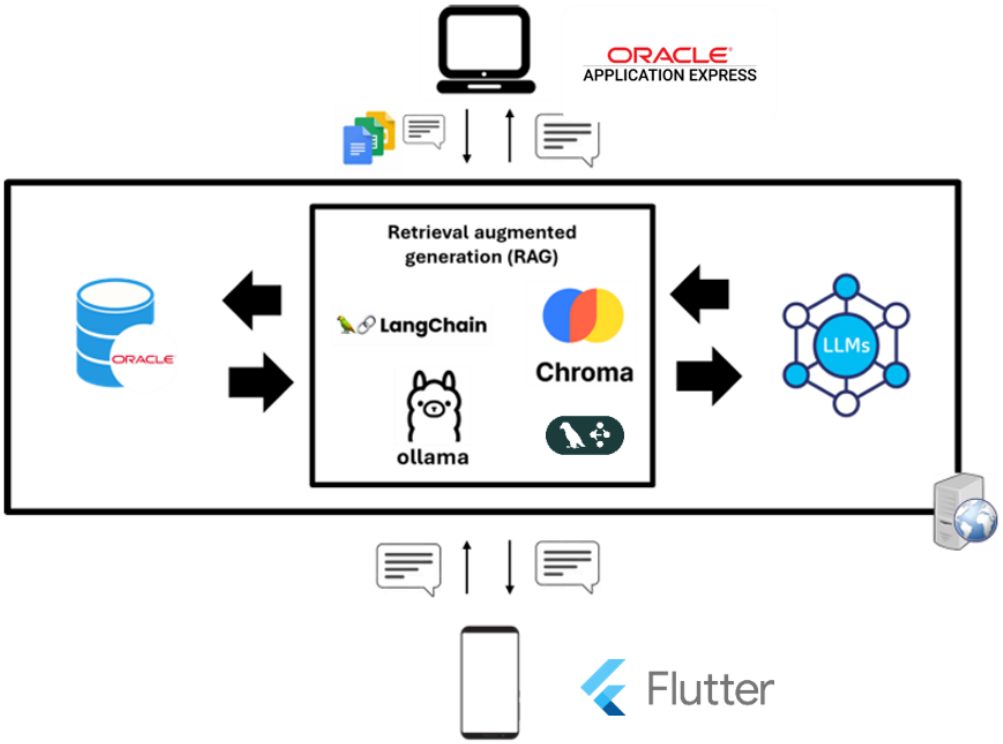


Fig. 3 Diagrama de composición de tecnologías

Fuente: Autoría propia

No solo se priorizará la precisión y coherencia de las respuestas, sino también la experiencia de usuario, medida mediante la calidad en uso del chatbot, asegurando que los estudiantes puedan acceder a la información de forma intuitiva y eficiente. Se implementarán contenedores Docker para garantizar la compatibilidad de los paquetes y librerías utilizadas en el desarrollo, facilitando la portabilidad y el despliegue eficiente del sistema en diferentes entornos. El servicio de chatbot podrá ser integrado a futuro en el aplicativo "UTN Móvil", que permitirá a los usuarios (estudiantes) acceder al chatbot y consultar información específica sobre beneficios, procesos y procedimientos del portafolio estudiantil.

1.4 Justificación

La presente investigación está alineada con el objetivo cuatro de los Objetivos de Desarrollo Sostenible (ODS), que busca garantizar una educación inclusiva, equitativa y de calidad, y promover oportunidades de aprendizaje durante toda la vida para todos [3]. Cabe destacar que, dentro del Plan de Creación de Oportunidades (2021-2025) de Ecuador, los lineamientos territoriales de la primera directriz ponen especial atención en el fortalecimiento de las TIC y en los sistemas de información vinculados al desarrollo humano [4].

Justificación Tecnológica. – El Diagnóstico sobre la Inteligencia Artificial en el Ecuador realizado por el Ministerio de Telecomunicaciones y de la Sociedad de la Información, específicamente en el punto 3.3, establece la necesidad de gobernanza y adopción de la inteligencia artificial en su desarrollo constante. Además, las conclusiones exponen la necesidad urgente de la creación de proyectos que puedan adherirse al surgimiento de nuevas tecnologías en inteligencia artificial [5].

Justificación Social. – El objetivo 7 del eje social, incluido en el Plan de Creación de Oportunidades 2021-2025 de Ecuador, subraya que el uso de tecnologías debe servir para ampliar las capacidades de los estudiantes. Se plantea que estas herramientas, al ser accesibles y de última generación, contribuyan a un entorno educativo más dinámico, eficiente y efectivo [4].

En el marco de este proyecto, los estudiantes de la UTN son quienes reciben el beneficio directo, sin importar la modalidad en la que cursen sus estudios. Al mismo tiempo, el personal administrativo y académico, así como la comunidad universitaria en general, también se ven favorecidos de forma indirecta, gracias a la implementación de un sistema más accesible y eficiente para manejar el portafolio estudiantil.

CAPÍTULO II
MARCO TEÓRICO

2.1 Modelos de lenguaje de gran escala (LLM)

2.1.1 Evolución de los modelos de lenguaje de gran escala (LLM)

Los LLM han atravesado un proceso de evolución significativo desde sus primeros días, convirtiéndose en la avanzada tecnología actual. Este progreso tiene sus raíces en la década de 1960, cuando Joseph Weizenbaum, en el Massachusetts Institute of Technology (MIT), presentó ELIZA, el primer chatbot de la historia. Este sistema se fundamentaba en el reconocimiento de patrones para replicar conversaciones humanas, lo que dio origen al interés y la exploración en el campo del procesamiento de lenguaje natural (NLP) [6].

En 1995 se presentó ALICE, el Artificial Linguistic Internet Computer Entity. Su llegada significó un salto importante respecto a ELIZA, que hasta entonces solo podía repetir patrones simples y producía conversaciones algo superficiales. ALICE, en cambio, utilizó técnicas de procesamiento de lenguaje natural y un motor basado en AIML, lo que le permitió dar respuestas mejor contextualizadas y manejar una gran variedad de preguntas. Gracias a ello, las interacciones resultaban más naturales y fluidas, marcando un cambio decisivo en la historia de los chatbots [6].

El aparecimiento de asistentes virtuales como Siri, Alexa y Google Assistant impulsaron significativamente el desarrollo de los LLMs. Siri, lanzado por Apple en 2011, incorporó el procesamiento del lenguaje natural y comandos de voz en dispositivos móviles. [7]. Alexa, presentada por Amazon en 2014, llevó esta tecnología al hogar con el control de dispositivos inteligentes mediante comandos de voz [8]. En 2016, Google Assistant combinó las búsquedas contextuales con un asistente virtual personalizado [9]. Concluyendo estos avances en un importante cambio en la interacción de las personas con la inteligencia artificial, y a su vez, siendo un preludio del paso agigantado de la IA hacia el surgimiento de los LLMs.

En 2017 el desarrollo del modelo de transformadores marcó un punto de inflexión en la evolución hacia los LLMs actuales. Presentado dentro del artículo “Attention is All You Need” [10], este modelo revolucionó el procesamiento de lenguaje natural al implementar mecanismos de atención que permitieron analizar relaciones entre palabras en secuencias completas, independientemente de su posición. Esto superó las limitaciones

de arquitecturas previas como Redes Neuronales Recurrentes, las cuales tenían dificultades para manejar dependencias largas en el texto. Los transformadores sirvieron como base para modelos como GPT, BERT y otros LLMs modernos. Este avance ha permitido generar texto más coherente y contextualizado, consolidando a los LLMs como herramientas clave en inteligencia artificial [11].

2.1.2 Retrieval-Augmented Generation (RAG): Concepto, funcionamiento y modelos

La estrategia RAG representa un método innovador en el campo de la inteligencia artificial al integrar dos procesos fundamentales.: recuperación de información relevante de una base de datos y la generación de respuestas textuales utilizando modelos generativos. A través de este enfoque, se superan las limitaciones típicas de los modelos generativos convencionales, los cuales dependen exclusivamente de su conocimiento interno para generar respuestas. La inclusión de un mecanismo de recuperación permite que RAG utilice datos adicionales y actualizados, lo cual incrementa la exactitud y pertinencia de las respuestas producidas [12].

El funcionamiento de un sistema RAG puede entenderse en dos momentos. Primero, se realiza una búsqueda en la base de conocimiento con ayuda de modelos de embeddings, que convierten las palabras en vectores dentro de un espacio matemático. Después, un modelo de lenguaje de gran escala utiliza esa información como contexto para construir la respuesta. Esta forma de trabajo resulta dinámica, porque la base de datos puede actualizarse fácilmente sin necesidad de reentrenar el modelo generativo, lo que asegura mayor flexibilidad y precisión en escenarios cambiantes [13].

El enfoque RAG surgió en 2020 gracias a un grupo de investigadores de Facebook AI Research (FAIR), liderados por Patrick Lewis y su equipo. En su artículo mostraron cómo esta técnica podía fortalecer tareas de procesamiento de lenguaje natural que requieren acceder a información externa, algo clave para sistemas como los chatbots o las aplicaciones de preguntas y respuestas, donde la exactitud de los datos es fundamental [13].

Funcionamiento [14]**:** la estructura del enfoque RAG puede ser explicada a través de un “naive RAG” (RAG básico o “ingenuo”), que contiene las etapas fundamentales para la construcción de este sistema, siguiendo un proceso de indexación, recuperación y generación.

Para indexar la información, lo primero que se hace es tomar los datos en texto plano, ya sea que provengan de un PDF o de formatos semiestructurados como JSON o

Markdown. Todo se unifica en un mismo formato y, después, se divide en fragmentos pequeños a partir de separadores definidos. Cada uno de esos fragmentos se transforma en un vector mediante un modelo de IA y se guarda en una base de datos vectorial, en el caso de nuestro proyecto ChromaDB. Gracias a este procedimiento, los LLM pueden trabajar con grandes volúmenes de información sin perder eficiencia.

Durante la recuperación, después de recibir una entrada del usuario, el sistema convierte el texto plano de dicha entrada en una representación vectorial con el mimo método de transformación utilizado durante la indexación (se utilizan modelos como el all-MiniLM-L6-v2 de hugging face). A continuación, se analizan varios puntajes de similitud entre la representación de entrada y las representaciones guardadas anteriormente en la base de datos, el sistema da prioridad y recupera los K fragmentos con mayor similitud a la consulta, transformándolos de vuelta a texto plano.

En la generación, la consulta planteada y los fragmentos seleccionados se combinan para crear un prompt coherente, al cual un modelo de lenguaje de gran escala debe formular una respuesta. El sistema puede ajustar la forma de responder del LLM (puede ser uno local como llama3 o Saas como GPT4) dependiendo de lo que se necesite para la tarea, siendo posible usar su conocimiento interno o limitarse a la información que se le haya dado en los documentos. En conversaciones largas, se puede incluir el historial previo como parte del prompt, lo que le permite manejar diálogos más fluidos y en múltiples turnos. Para entender mejor el flujo de información a partir de este enfoque, nos podemos guiar a partir de la Figura 4.

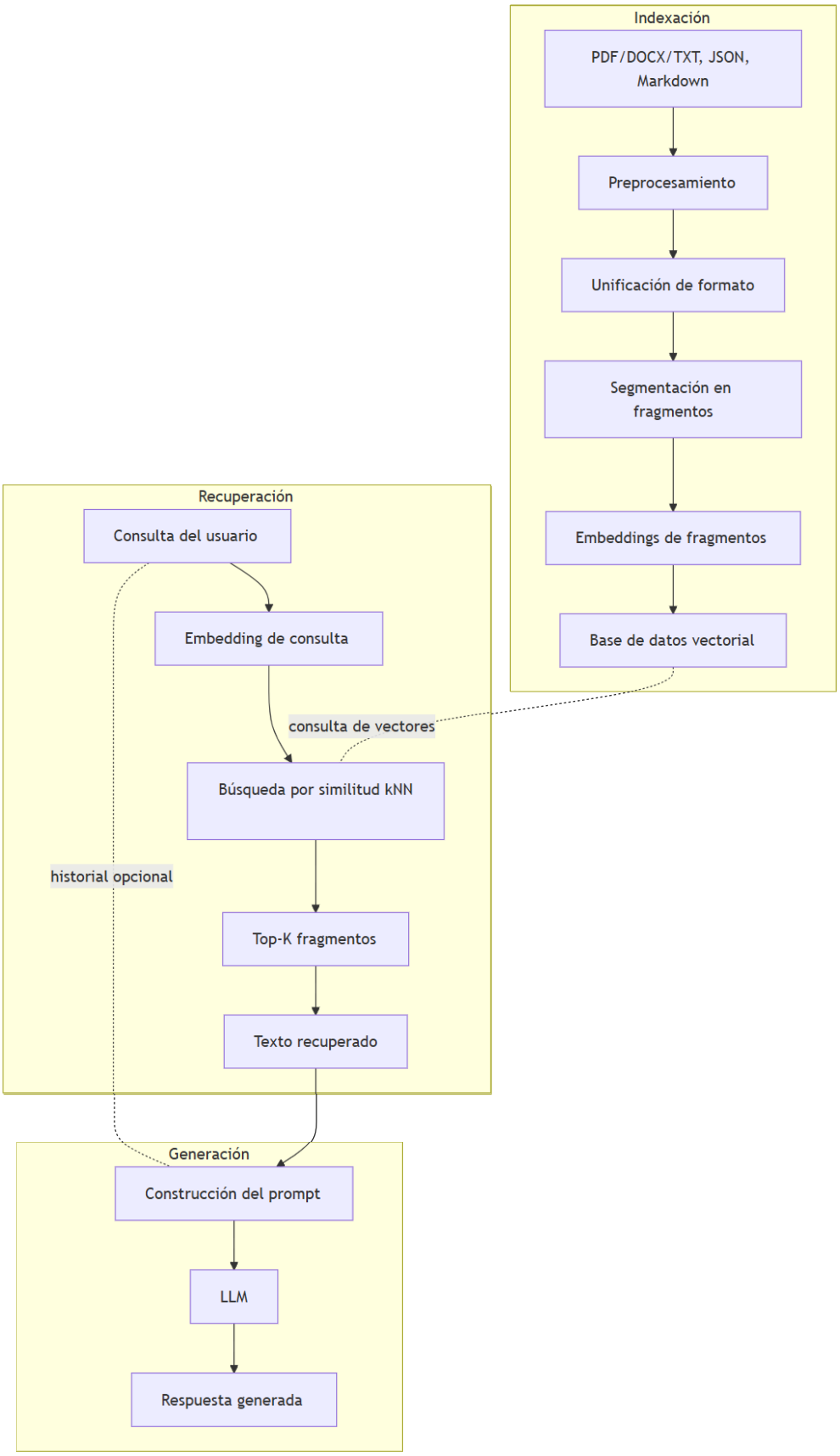


Fig. 4. Diagrama de flujo del funcionamiento de un sistema RAG simple

Fuente: Autoría propia

Si bien se detalla de forma esencial el funcionamiento del paradigma, un sistema RAG básico puede enfrentar diversos problemas en sus fases clave. Durante la recuperación, puede fallar en precisión, seleccionando fragmentos irrelevantes. En la generación, existe el riesgo que el modelo cree contenido no respaldado por el contexto, además de producir respuestas irrelevantes o "alucinaciones". Al integrar la información recuperada, pueden surgir salidas incoherentes, redundantes o mal conectadas, mientras que evaluar la relevancia y mantener la coherencia estilística añade complejidad.

Sistema RAG avanzado: es una implementación más precisa del paradigma, se pone más énfasis en la indexación de información a partir de preprocesamiento del texto plano, esto puede realizarse mediante añadido de texto, o reestructuración de este para dar coherencia a la recuperación. Posterior a la recuperación de dicho texto, se emplea técnicas para el manejo y posicionamiento de fragmentos clave, de tal manera que el LLM pueda entender el contexto y generar respuestas más precisas. Es posible reestructurar la base de datos de conocimiento del sistema a otros enfoques, a partir de bases de datos vectoriales, alternativamente se puede usar una red de grafos, que permita el ordenamiento de la indexación de la información de manera más coherente [15].

Sistema RAG modular [16]**:** El diseño modular del RAG no es rígido ni lineal, sino que se organiza en tres capas. En la primera se ubican los módulos centrales, que realizan las funciones básicas: indexar, recuperar y generar información. Luego, un segundo nivel incorpora submódulos que afinan y especializan esas funciones. Finalmente, en la capa más baja están los operadores, encargados de tareas concretas como manejar consultas o elegir qué fragmentos usar. Esta estructura en niveles permite que el sistema sea más flexible y fácil de ajustar frente a arquitecturas más tradicionales.

Esta variante tiene capacidad para adaptarse a consultas complejas y gestionar datos provenientes de diversas fuentes. Este paradigma utiliza mecanismos avanzados como el ruteo, que dirige las consultas a flujos específicos (lleva la consulta a módulos y fragmentos relevantes), y la programación, que permite la ejecución secuencial o paralela de procesos según sea necesario (Necesario para el manejo de consultas de gran volumen). Además, cuenta con mecanismos de fusión que integran múltiples resultados en respuestas coherentes y completas.

Modelo RAG- Sequence: En este modelo de RAG, el sistema se encarga de recolectar todos los fragmentos relevantes en la base de conocimiento, para al final, brindar una respuesta con toda la información recolectada. Genera la respuesta final en un solo paso [13].

Modelo RAG- Token: El sistema se encarga de buscar los fragmentos más relevantes, pero no los utiliza de inmediato, en cambio, funciona de manera iterativa en la generación de la respuesta, realizando la recuperación de información relacionada a medida que va generando texto en lenguaje natural. Este modelo puede reducir la redundancia de las respuestas, pero puede resultar en un mayor costo computacional, debido a la eficiencia que debe tener el "recuperador" en cada operación [13].

2.1.3 Aplicaciones de LLMs y sistemas RAG en la educación

La efectividad de los sistemas RAG y su integración con LLMs ha permitido la transparencia de la información al reducir drásticamente la redundancia en la información que se requiere ser obtenida, evitando la generación de texto aleatorio que pueda resultar en información falsa. Este sistema presenta una mayor precisión factual, acceso dinámico al conocimiento y una mejor interpretabilidad, lo que fortalece la confiabilidad y aplicabilidad de la IA en diversos contextos [17].

Mejora en la evaluación de prácticas de tutorías a partir de un sistema RAG: Este trabajo examina el uso LLMs, como GPT-3.5 y GPT-4, para evaluar automáticamente las competencias de los tutores novatos en la aplicación de estrategias de tutoría socioemocional. Los datos previos para compararse se obtuvieron a partir de sesiones de tutoría de matemáticas de escuelas secundarias en los Estados Unidos. Se concluyó que un sistema RAG muestra un mejor rendimiento en precisión y menores costos financieros en comparación otras estrategias de prompt. Los hallazgos indican que este enfoque puede servir como base para crear programas de capacitación adaptados a las necesidades de los tutores, lo que contribuiría a aumentar la eficacia de la enseñanza personalizada. [18].

Mejora en la generación de respuestas de un LLM en el campo de las matemáticas a partir de un sistema RAG: El trabajo explora el uso de este sistema para mejorar la calidad de las respuestas generadas por LLMs en preguntas y respuestas sobre matemáticas para estudiantes de secundaria. El estudio se centra en cómo los prompts que incorporan contenido de un libro de texto de matemáticas de alta calidad pueden mejorar las respuestas a preguntas reales de estudiantes en álgebra y geometría. A través de una encuesta, los resultados muestran que los estudiantes prefieren las respuestas generadas a través del sistema RAG cuando no están demasiado centradas en el contenido del libro de texto [19].

Implementación de un sistema RAG en un curso online masivo: En una prueba realizada con un curso en línea de la Universidad TELUQ sobre inteligencia artificial,

que incluye tanto teoría como ejercicios de práctica (por ejemplo, verdadero/falso y opción múltiple), se compararon distintos modelos. Los resultados fueron claros: GPT-4 combinado con RAG logró cerca de un 80% de precisión, frente al 60% alcanzado por versiones anteriores. Esta diferencia sugiere que el uso de RAG puede abrir la puerta a experiencias de aprendizaje más personalizadas y a evaluaciones educativas más efectivas [20].

Desarrollo de un tutor virtual: En este trabajo se desarrolló y evaluó un tutor virtual, capaz de comunicarse con estudiantes en lenguaje natural al momento de requerir ayuda con métodos de investigación cuantitativa. Para que el tutor pueda entender las preguntas de los estudiantes, se utilizó la API de ChatGPT. Para evitar que el tutor proporcione información incorrecta, se empleó un sistema RAG en su desarrollo. El estudio resalta el potencial de los tutores basados en este sistema para ofrecer experiencias de aprendizaje personalizadas y precisas, reduciendo el riesgo de información incorrecta asociada a los LLMs [21].

El uso del enfoque RAG toma cada vez más fuerza en ámbitos educativos, su integración con modelos de lenguaje de gran escala permite brindar asistencia en una gran cantidad de ámbitos, suponiendo una potenciación al paradigma de aprendizaje actual, se puede concluir que el futuro del aprendizaje va de la mano con la actualización de estas tecnologías, ofreciendo una experiencia personalizada tanto a docentes como estudiantes, mejorando la calidad de información a investigarse, siendo estas herramientas cada vez más incidentes en la actualidad [22].

2.2 Herramientas de desarrollo

2.2.1 Bases de datos de embeddings: ChromaDB y su rol en sistemas RAG

Una base de datos de embeddings es un concepto que ha tomado fuerza recientemente, siendo bases de datos que almacenan información a modo de representaciones matemáticas de datos como texto, imágenes o audio, siendo una herramienta esencial para la comparación de características de alto nivel a través de dichas representaciones [23].

ChromaDB es una base de datos diseñada específicamente para gestionar embeddings, esta tecnología surgió para satisfacer la necesidad de manejar datos vectoriales de manera eficiente y, al ser de código abierto, facilita la construcción de soluciones innovadoras en el campo de la inteligencia artificial [24].

La principal función de ChromaDB es almacenar y buscar información basada en similitudes, lo que la hace especialmente útil en aplicaciones de inteligencia artificial

como chatbots, sistemas de recuperación de información o recomendaciones, siendo una alternativa sutil en cuanto a implementaciones que requieran un tiempo de respuesta corto en cuanto a un contexto amplio [25].

En lugar de quedarse únicamente en las palabras clave, ChromaDB transforma los textos en vectores densos que capturan su significado. De este modo, las búsquedas no dependen solo de coincidencias literales, sino de la similitud semántica, lo que las vuelve más precisas y cercanas a lo que el usuario realmente quiere encontrar [26], de esta manera se facilita la conexión entre bases de datos y modelos de lenguaje ajustados, procesando consultas de usuarios al identificar y clasificar los "pedacitos" de las fuentes más relevantes. Esto resulta en chatbots más efectivos y eficientes, capaces de generar respuestas contextuales e informativas [27].

En un enfoque RAG, ChromaDB actúa como la base donde se guardan los vectores que representan la información. Esto le da al sistema la capacidad de buscar por sentido y no únicamente por coincidencias literales de palabras. De esta manera, aunque el usuario formule su consulta con términos distintos, el sistema puede devolver resultados relevantes [28].

El uso de ChromaDB en un sistema RAG puede optimizarse si este es un modelo secuencial, ya que el uso iterativo en una misma instancia de base de datos vectorial puede resultar en cuellos de botella, es necesario optimizar el proceso si se requiere un modelo iterativo, o en el caso de presentarse escenarios de consultas masivas se debe optar por datos estructurados [29].

2.2.2 Ollama: Implementación de LLMs

Ollama es una plataforma diseñada para manejar modelos de lenguaje de gran escala. Una de sus ventajas es que puede ejecutarse localmente o en la nube, lo que la convierte en una opción flexible para quienes desean añadir capacidades de IA generativa a sus aplicaciones [30]. Lo más atractivo de esta herramienta es que no depende de servidores externos: los modelos de IA pueden ejecutarse directamente en el ordenador del usuario. Eso mejora la privacidad y hace que las respuestas sean más rápidas. Además, su interfaz es bastante simple, lo que ayuda a integrar modelos ya entrenados en proyectos personalizados, por ejemplo en el desarrollo de chatbots o asistentes virtuales [31].

En un sistema RAG, Ollama juega un papel clave al proporcionar LLMs generan respuestas basadas en información recuperada de bases de datos externas. Ollama se integra en el flujo de este enfoque, donde primero se realiza una recuperación de datos desde una base de datos vectorial (como ChromaDB) y luego, el modelo de Ollama genera

respuestas contextuales y detalladas utilizando esa información recuperada junto a un LLM. En este proceso, Ollama optimiza la generación de texto, mejorando la relevancia y coherencia de las respuestas del chatbot o asistente, existe una buena compatibilidad entre ChromaDB y Ollama si se requiere un chatbot con herramientas de código abierto [32].

Ollama acerca el uso de modelos de lenguaje avanzados a cualquier entorno, permitiendo que incluso se ejecuten en sistemas RAG locales. De esta manera, se pueden crear asistentes o chatbots sin depender de la nube. Esto también da la oportunidad de comparar cómo se comportan los modelos cuando corren en un equipo local frente a cuando se despliegan en servidores externos. Además, brinda la flexibilidad de elegir el modelo que mejor se ajuste a nuestras necesidades específicas [33].

2.2.3 Desarrollo de aplicaciones con LLMs: LangChain

LangChain es un marco de trabajo diseñado para la integración y manejo de LLMs en el desarrollo de aplicaciones de manera eficiente. Esta herramienta es especialmente útil para construir sistemas avanzados como chatbots, motores de búsqueda personalizados, asistentes virtuales, sistemas de recomendación y más [26].

La arquitectura de LangChain busca hacer más sencillo el trabajo con aplicaciones complejas. Con sus componentes es posible integrar distintos procesos, desplegar APIs que escalen sin problemas y vigilar el estado de la aplicación en tiempo real. Además, soporta enfoques como RAG, que consiste en complementar las respuestas de los modelos con datos externos, lo que se traduce en resultados más relevantes y precisos [34].

Trabajar con LangChain plantea un reto importante en materia de seguridad. Como muchas aplicaciones se apoyan en servicios externos, existe la posibilidad de que se expongan datos sensibles. La herramienta ya ofrece opciones como permisos granulares y entornos aislados, pero aun así se necesita una gestión cuidadosa para evitar riesgos y garantizar que la información se mantenga privada [34].

El uso de LangChain en conjunto con un LLM puede facilitar el uso de bases de datos mediante lenguaje natural. Siendo posible a partir de 2 métodos: convertir preguntas en lenguaje natural en consultas SQL, o extraer palabras clave de las preguntas y usarlas en herramientas de búsqueda. Langchain puede crear un flujo de trabajo para el LLM incorporado, que permita manejar no solo la creación de peticiones en SQL, sino también interpretar las consultas de forma simple [35]. En la Figura 2, se presenta un esquema o

cadena para el uso de Langchain como herramienta clave en la consulta a bases de datos mediante lenguaje natural.

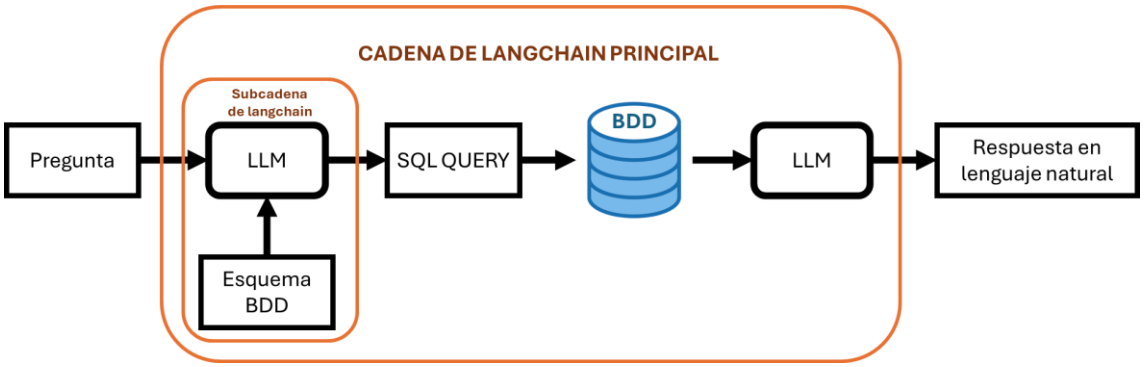


Fig. 5. Ejemplo de una cadena de langchain para consultas SQL en lenguaje natural

Fuente: Autoría propia

2.2.4 Contenedores Docker y su papel en sistemas RAG

Docker es una herramienta que permite crear contenedores, que son como pequeñas cajas que empaquetan una aplicación con todo lo que necesita para funcionar (bibliotecas, configuraciones, etc.), asegurando que se ejecute igual en cualquier computadora. Es como llevar una aplicación en una maleta lista para usar, sin importar dónde se abra: en un PC, un servidor o la nube. Esto hace que sea más fácil desarrollar, probar y desplegar software sin preocuparse por problemas de compatibilidad [36].

Uno de los aportes de Docker en materia de seguridad es que mantiene cada contenedor aislado: lo que ocurre dentro de uno no afecta directamente al sistema operativo ni a los demás contenedores. Esto ya reduce bastante el riesgo de vulnerabilidades. También es útil que se puedan usar imágenes verificadas y aplicar políticas de control de acceso para decidir quién puede modificarlos o ejecutarlos, sin embargo, si se descuida la configuración de red, los permisos o las actualizaciones, un contenedor mal configurado puede terminar siendo un punto de entrada para un ataque [37].

Docker es ideal para un sistema RAG porque permite empaquetar y desplegar todas las herramientas necesarias, como bases de datos (como Chromadb), modelos de lenguaje y servidores de aplicaciones, en contenedores independientes pero integrados, asegurando consistencia entre entornos y facilidad para escalar. En este contexto, CUDA [38] (la plataforma de NVIDIA para computación en GPU) es relevante si el sistema utiliza GPUs para acelerar el entrenamiento o la inferencia de modelos, como LLMs. Docker facilita el uso de CUDA al permitir contenedores que acceden directamente a las

GPUs con los controladores y bibliotecas necesarios, optimizando el rendimiento de los cálculos intensivos [39].

2.2.5 Flutter: Framework de desarrollo móvil

Flutter es una herramienta de desarrollo de código abierto creada por Google, siendo eficaz en la creación de aplicativos móviles, web y escritorio a partir de un solo código base. Este framework permite crear interfaces de usuario altamente personalizables y fluidas mediante el uso de widgets, permitiendo desarrollos rápidos y consistentes en múltiples plataformas. Además, utiliza el lenguaje de programación Dart, optimizado para garantizar un rendimiento eficiente y una experiencia nativa en las aplicaciones desarrolladas [40].

En el contexto del desarrollo de un chatbot, Flutter ofrece múltiples herramientas que facilitan su integración en una aplicación móvil. Flutter puede ser fundamental para desarrollar una interfaz que facilite la interacción de los usuarios con el sistema de forma sencilla y en tiempo real. [41].

El uso de Flutter para el desarrollo de interfaz de usuario final podría aprovechar su arquitectura de widgets para construir interfaces de conversación modernas, como burbujas de chat, paneles interactivos y botones dinámicos [41].

Una de las ventajas prácticas de Flutter es que se lleva bien con APIs y servicios backend. Se puede conectar tanto a modelos de lenguaje (como los de OpenAI o Hugging Face) como a bases de datos vectoriales como ChromaDB. Así, la aplicación envía las consultas al sistema RAG, obtiene la información y la devuelve al usuario casi en tiempo real. En Dart, herramientas como “http” o “dio” permiten manejar estas solicitudes sin trabas, de modo que la interacción con el chatbot resulta fluida y rápida. [42].

2.3 Metodologías de desarrollo

2.3.1 Cross Industry Standard Process for Data Mining (CRISP-DM)

Esta es una metodología estandarizada para análisis de datos y minería, estructurada en seis fases: entendimiento del negocio, donde se definen objetivos y métricas; entendimiento de los datos, que implica explorar y evaluar la calidad de los datos disponibles; la preparación, donde se limpian, transforman y escogen los más relevantes; modelado, que incluye la selección y entrenamiento de modelos; evaluación, donde se valida si el modelo cumple con los objetivos del negocio; y despliegue, que consiste en implementar la solución en un entorno real. Esta metodología es iterativa y flexible, adaptándose a las necesidades específicas de cada proyecto [43].

La metodología se puede aplicar eficazmente en la creación de un sistema RAG, siguiendo el ejemplo de este caso de estudio [44], es posible a partir de seguir sus seis fases estructuradas: En la fase de Comprensión del Negocio, es esencial entender los requisitos del proyecto desde una perspectiva sistemática. Por ejemplo, si el objetivo es desarrollar un chatbot que brinde información de conveniencia estudiantil, se debe definir claramente el tipo de información que se necesita recuperar y cómo debe interactuar con los estudiantes.

Durante la Comprensión de los Datos, se recolectan los datos iniciales, como textos del reglamento universitario en nuestro caso, y se realizan procedimientos para la relación y entendimiento de ellos. Esto incluye identificar problemas dentro su estructura, información faltante o cualquier otro inconveniente que comprometa la calidad de los datos para así obtener una comprensión inicial de su contenido y estructura.

El primer paso en un sistema de este tipo es preparar los datos: limpiarlos, quitar redundancias y darles un formato uniforme. Solo así se pueden dividir en fragmentos y transformarlos en vectores que luego serán útiles en la búsqueda semántica.

Después viene el modelado. Aquí lo importante es decidir qué modelos de embeddings se van a usar e integrarlos al sistema, de modo que el chatbot pueda recuperar información con mayor precisión. Con esa base, los LLM tienen mejores condiciones para generar respuestas adaptadas al contexto.

La Evaluación de los modelos desarrollados es crucial para asegurarse de que cumplen con los objetivos. En nuestro caso significa verificar que el sistema pueda recuperar información del reglamento estudiantil que sea relevante y precisa de manera efectiva. Se pueden usar métricas de precisión, cobertura y satisfacción del usuario para esta evaluación.

Finalmente, la fase de despliegue, en dónde se implementa la solución en un entorno real. Para un chatbot del contexto universitario significa que el sistema RAG sea accesible a los usuarios finales (estudiantes), permitiéndoles interactuar con él para obtener asistencia en cuanto a la normativa estudiantil. También se debe considerar el monitoreo y mantenimiento continuo del sistema para asegurar su rendimiento y actualización con nueva información universitaria.

2.3.2 Metodología de desarrollo SCRUM

Scrum es un enfoque ágil diseñado para organizar y llevar a cabo proyectos, con especial aplicación en el desarrollo de software. Esta metodología se distingue por priorizar la adaptabilidad, el trabajo en equipo y la entrega constante de valor al usuario

final. A diferencia de los enfoques tradicionales, Scrum evita adherirse a un plan estricto y lineal, sino que se adapta a los cambios y necesidades del proyecto a medida que este avanza. El origen de Scrum se remonta a 1986, cuando Takeuchi y Nonaka publicaron un artículo que proponía una nueva forma de gestionar proyectos, inspirada en la dinámica de los equipos de rugby. Más tarde, en la década de 1990, Jeff Sutherland y Ken Schwaber estructuraron esta metodología y la incorporaron a la Alianza Ágil [45].

Scrum comprende varios enfoques, los cuales se pueden explicar a partir del análisis de la memoria de trabajo acerca del mismo [46]. Scrum se organiza en periodos de trabajo conocidos como "sprints", que generalmente tienen una duración de dos a cuatro semanas.

Un sprint en Scrum empieza con una reunión de planificación y termina con una revisión del trabajo. En el camino, el equipo se reúne cada día en encuentros breves para mantener el rumbo y resolver lo que haga falta. El marco también define tres roles clave. El Product Owner decide qué es prioritario en el producto y orienta al equipo. El Scrum Master se asegura de que el proceso fluya, ayudando a quitar obstáculos. Y el equipo de desarrollo, organizado de manera autónoma, se encarga de concretar lo planificado.

El trabajo se gestiona con el Product Backlog, una lista que recoge todo lo que el producto requiere y que se ajusta constantemente. En cada sprint se elige una parte de esa lista para crear el Sprint Backlog. Al final, se lleva a cabo la retrospectiva, un espacio donde el equipo conversa sobre lo aprendido y sobre cómo hacerlo mejor en la siguiente iteración.

2.4 Evaluación del chatbot

2.4.1 Fundamentos de la norma ISO/IEC 25022 en el desarrollo de software

La norma ISO/IEC 25022, perteneciente a la serie SQuaRE (Systems and Software Quality Requirements and Evaluation), está diseñada para abordar la evaluación de la calidad en uso de sistemas y productos de software. Este estándar proporciona un conjunto de medidas destinadas a analizar cómo un sistema o producto satisface las necesidades de los usuarios dentro de su contexto operativo [47]. Estas medidas están alineadas con las características establecidas en la ISO/IEC 25010 referentes a la calidad en uso, siendo estas la efectividad, eficiencia, satisfacción, seguridad y cobertura del contexto [48].

Esta norma no dicta números cerrados para las métricas, sino que propone guías para que cada organización pueda aplicarlas según su propio contexto. De ese modo, lo

importante no es cumplir con un valor fijo, sino adaptar la evaluación del software a las condiciones reales de uso y a lo que los usuarios necesitan. [47].

El objetivo esencial de la ISO/IEC 25022 es ofrecer un marco organizado que apoye toda gestión y garantía de la calidad, así como permitir una evaluación sistemática del rendimiento del software. Este estándar puede ser aplicado tanto en sistemas complejos como en los más simples e individuales, siendo relevante para cualquier sistema informático utilizado por personas [49].

La norma incluye anexos informativos que ofrecen ejemplos prácticos sobre la medición de la cobertura del contexto, la normalización de medidas y el proceso de evaluación de la calidad en uso. Asimismo, explora cómo los modelos de calidad y los conceptos de medición se relacionan entre sí [49].

La ISO/IEC 25022 es una herramienta fundamental para el desarrollo, adquisición, evaluación y mantenimiento de sistemas y productos de software. Su aplicación garantiza que los productos finales satisfagan las expectativas y requerimientos de los usuarios en sus entornos operativos reales, contribuyendo al mejoramiento continuo de la calidad en uso.[49]

2.4.2 Relevancia de la norma en proyectos de IA

La ISO/IEC 25022 tiene una relevancia significativa en proyectos de inteligencia artificial debido a su enfoque en la medición objetiva de la calidad del software desde la perspectiva del usuario final. En el desarrollo de sistemas de IA, donde la funcionalidad y la satisfacción del usuario son factores clave, este estándar proporciona un marco que permite evaluar el desempeño del sistema en términos como efectividad, eficiencia, seguridad y satisfacción. Estas dimensiones son esenciales para garantizar que las soluciones basadas en IA no solo sean técnicamente sólidas, sino también usables y confiables [50].

La practicidad de la ISO/IEC 25022 permite atender un reto frecuente en el desarrollo de IA: que lo que entrega el modelo realmente coincida con lo que el usuario necesita. Este estándar propone medir no solo la precisión de los algoritmos, sino también factores como la rapidez con la que responden en tiempo real y la calidad que perciben quienes los usan. Con ello, se busca que la inteligencia artificial no se limite a cumplir metas técnicas, sino que ofrezca confianza y valor en la práctica [51].

En la práctica, la ISO/IEC 25022 se convierte en una herramienta útil para validar proyectos de IA en sectores donde la confianza es indispensable, como el educativo, el médico o el financiero. Sus métricas hacen posible comparar de forma justa el

rendimiento de distintos sistemas y ayudan a los desarrolladores a detectar qué se puede mejorar. De esta manera, los productos finales no solo cumplen estándares de calidad, sino que también ganan mayor aceptación entre usuarios y organizaciones, fortaleciendo la credibilidad de estas tecnologías [52].

2.4.3 Uso de la norma en aplicativos móviles

La implementación de la ISO/IEC 25022 en aplicaciones móviles no solo asegura la entrega de productos de calidad, sino que también fomenta la confianza y satisfacción del usuario. Este estándar constituye un recurso indispensable para diseñadores y desarrolladores que buscan crear soluciones robustas, escalables y adaptadas a las necesidades dinámicas de sus usuarios [53].

En la práctica, el estándar revisa si una aplicación puede adaptarse a distintos escenarios de uso. Toma en cuenta, por ejemplo, los recursos con los que cuenta el usuario y sus características particulares. Gracias a este análisis, se busca que el software siga siendo accesible y útil en una amplia variedad de dispositivos y situaciones [53].

En un caso de estudio enfocado en una aplicación móvil para el cálculo de la densidad mineral ósea, el uso de esta norma resultó en una calidad en uso del 93%, destacando su capacidad para evaluar aspectos como la precisión de los resultados y la experiencia del usuario durante el manejo de datos complejos [53].

2.4.4 Métricas y criterios de evaluación del chatbot

En el desarrollo y evaluación de un chatbot, la ISO/IEC 25022 permite medir la eficacia mediante métricas como la proporción de tareas completadas exitosamente y la frecuencia de errores cometidos por los usuarios al interactuar con el sistema. Estas métricas se aplican evaluando, por ejemplo, si el chatbot logra interpretar correctamente las peticiones de los usuarios y proporcionar respuestas precisas [52].

En la práctica, usar la norma significa decidir qué métricas se aplican y adaptarlas al chatbot y al contexto donde opera. Con ello, se asegura que los resultados obtenidos muestren realmente cómo se está desempeñando el sistema. En la tabla 1 se presentan diferentes métricas que pueden ser utilizadas y personalizadas de acuerdo con el uso o implementación de un producto de software.

Tabla 1 Métricas de calidad en uso aplicables a un chatbot.

Fuente: [52].

METRICAS DE CALIDAD EN USO APLICABLES A UN CHATBOT		
Características	**Sub características**	**Métricas**

Eficacia	• Exactitud. • Completitud.	• Tasa de éxito de las tareas. • Precisión de las respuestas.
Eficiencia	• Consumo de recursos. • Rapidez.	• Tiempo de respuesta promedio. • Número de interacciones por tarea.
Satisfacción	• Confianza. • Preferencia de usuario.	• Puntuación de satisfacción del usuario. • Net Promoter Score (NPS): Evaluación de la probabilidad de que los usuarios recomienden el chatbot a otros.
Seguridad	• Confidencialidad. • Integridad.	• Tasa de error en privacidad. • Nivel de protección de datos.
Cobertura de objetivos	• Completitud funcional. • Adaptabilidad.	• Porcentaje de objetivos cumplidos. • Ámbito de funcionalidad cubierto

Para el desarrollo de este chatbot, se ha decidido utilizar las métricas relacionadas a: La eficacia, eficiencia y satisfacción [52]:

La eficacia de un chatbot puede medirse a través de dos indicadores clave: la tasa de éxito en las interacciones y la precisión de las respuestas. La primera representa el porcentaje de conversaciones en las que los usuarios logran completar sus objetivos, como encontrar información o resolver dudas. Por otro lado, la precisión de las respuestas evalúa la proporción de respuestas correctas entregadas por el chatbot, lo que refleja la capacidad de este para ofrecer información fiable y relevante.

En la práctica, la eficiencia se observa viendo cuánto tarda un usuario en cumplir una tarea y cuántos pasos debe dar para lograrlo. Si el chatbot reduce ese esfuerzo, entonces está cumpliendo bien su función. La satisfacción, en cambio, se mide a partir de lo que opinan los propios usuarios: si están conformes con la experiencia o si el chatbot responde como esperaban. Para recoger esas opiniones se pueden usar encuestas o pequeños formularios al cierre de cada interacción.

2.5 Trabajos relacionados

2.5.1 Chatbots en la Universidad Técnica del Norte

En el estudio [54]**:** se propuso la implementación de un chatbot para mejorar la asistencia bibliotecaria en la Universidad Técnica del Norte (UTN), con el fin de optimizar el acceso a información y resolver inquietudes que pueda tener la comunidad universitaria eficientemente. Se emplearon métodos de investigación exploratoria y descriptiva, así como técnicas de procesamiento de lenguaje natural para el desarrollo del chatbot, que se diseñó para comunicarse con los usuarios y ofrecer soluciones a sus consultas frecuentes, horarios de atención ytrámites disponibles en la biblioteca.

La solución se desarrolló utilizando Microsoft Bot Framework, al que se añadieron los servicios de procesamiento de lenguaje natural de Azure, donde además se desplegó tanto el bot como el API. La experiencia evidenció mejoras en la interacción con los estudiantes, ya que recibieron respuestas más rápidas y útiles. Aun así, se presentaron dificultades, entre ellas la falta de capacitación suficiente del personal y la inversión económica que demanda una implementación a gran escala. Para trabajos futuros, se sugiere realizar una evaluación continua del chatbot y explorar la integración de inteligencia artificial avanzada para optimizar aún más la interacción y el aprendizaje del sistema.

El proyecto [55]: tuvo como objetivo desarrollar e implementar un chatbot con inteligencia artificial para el blog de la Carrera de Software, siendo su objetivo el automatizar la atención a preguntas frecuentes y optimizar procesos administrativos en la Universidad Técnica del Norte. Se utilizó la metodología XP y herramientas como Dialogflow, Python y Firebase, esto basándose en técnicas de procesamiento de lenguaje natural y machine learning, permitiendo una interacción lógica y en tiempo real con los usuarios. El chatbot fue integrado mediante la plataforma Flask y desplegado en Heroku para garantizar su disponibilidad.

El uso del chatbot trajo beneficios claros: las respuestas fueron más rápidas y el área administrativa tuvo menos carga de trabajo porque la información se mantenía actualizada de manera automática. Sin embargo, no todo fue perfecto. El modelo sigue dependiendo mucho de los datos de entrenamiento y, en ocasiones, se confunde con preguntas difíciles de interpretar. Pensando en el futuro, sería útil reforzar su aprendizaje y añadir la capacidad de reconocer emociones, lo que haría la interacción más cercana y positiva para los usuarios.

2.5.2 Uso de sistemas RAG para la creación de chatbots en otros contextos

El objetivo principal del estudio [56]**:** fue desarrollar GastroBot, un chatbot chino especializado en enfermedades gastrointestinales, utilizando un sistema RAG para mejorar la precisión y relevancia en las respuestas clínicas. Para lograr esto, se utilizó un conjunto de datos compuesto por 25 guías clínicas y 40 artículos recientes sobre gastroenterología, afinando el modelo de embeddings gte-base-zh con técnicas específicas para el dominio médico.

El análisis de resultados mostró que el modelo tuvo un desempeño superior al compararse con las versiones anteriores: la tasa de aciertos fue un 18% más alta que en el modelo base y un 20% mayor que en el modelo de embeddings de OpenAI. En las pruebas con RAG, GastroBot alcanzó cifras destacadas, con un 95% en recuperación de contexto, 93.73% en fidelidad y 92.28% en relevancia de las respuestas. Pese a estos logros, aún persisten limitaciones relacionadas con la actualización constante del conocimiento y su cobertura. Por ello, una línea futura de trabajo es ampliar la base de datos, optimizar la recuperación de información y mejorar la integración con los registros médicos electrónicos.

El estudio [57]**:** tuvo como objetivo desarrollar VASC.AI, una interfaz de chat basada en inteligencia artificial, específica para la cirugía vascular, mediante la integración de un sistema RAG. Para su desarrollo, se utilizó una base de datos con más de 200,000 resúmenes clínicos, guías de práctica y ensayos clínicos relevantes, los cuales fueron vectorizados y procesados para mejorar la precisión de un modelo LLM (Large Language Model).

En las pruebas con el programa VESAP-5 de cirugía vascular, VASC.AI alcanzó un 93.8% de respuestas correctas. Esto lo ubicó claramente por encima de modelos generales como ChatGPT-3.5, ChatGPT-4 y ChatGPT-4o, cuyo mejor desempeño fue del 77.7%. Si bien cometió errores de razonamiento, no presentó equivocaciones de contenido, algo que se explica por su base de datos especializada. Entre sus limitaciones destacan la persistencia de fallos lógicos y la necesidad de actualizarse con nuevas guías clínicas. Hacia el futuro, se propone fortalecer su capacidad de razonamiento e incluir funciones que faciliten su aplicación directa en contextos médicos.

El estudio [21]**:** En este estudio se diseñó un tutor computarizado que combina un modelo de lenguaje con un sistema RAG para apoyar el aprendizaje de métodos de análisis cuantitativo. La idea principal fue evitar las alucinaciones típicas de los modelos generativos y ofrecer respuestas basadas en notas de curso previamente validadas.

El desarrollo se apoyó en tres piezas clave: la API de ChatGPT para comprender las preguntas, el modelo de embeddings text-embedding-ada-003 para la recuperación de fragmentos y la biblioteca LlamaIndex para estructurar la búsqueda. El tutor fue probado con 20 preguntas tomadas del curso, y las respuestas fueron revisadas por dos instructores mediante una rúbrica.

Los resultados fueron positivos: todas las preguntas recibieron respuestas correctas y, cuando se solicitó, el tutor incluso generó código en R. Aun así, se observó que en ocasiones las respuestas eran demasiado largas y podían distraer al estudiante. Entre las limitaciones se señalan esa variabilidad y la falta de una evaluación directa con usuarios reales. A futuro se propone ajustar la extensión de las respuestas y explorar el uso de modelos de código abierto que faciliten la adopción de este tipo de tutores en otros contextos educativos.

El estudio [52]**:** tuvo como objetivo desarrollar y evaluar un chatbot llamado "Tiny" para la gestión de inquietudes con el aula virtual Moodle, utilizando métricas de calidad en uso del estándar ISO/IEC 25022 en la PUCE Esmeraldas. Para ello, se emplearon tecnologías de inteligencia artificial como Microsoft Bot Framework, LUIS y QnA Maker, junto con herramientas de observación y análisis proporcionadas por Azure. La metodología combinó enfoques cualitativos y cuantitativos, recogiendo datos mediante observación y pruebas prácticas realizadas con estudiantes y docentes.

Durante la prueba, la mayoría de usuarios valoraron positivamente la experiencia con el chatbot: lo consideraron sencillo, eficaz y de ayuda para sus necesidades. Estos resultados confirman que la herramienta puede mejorar la manera en que se brinda apoyo en el ámbito educativo. Al mismo tiempo, se detectaron algunos aspectos por reforzar, como la integración de más funciones y la preparación de los usuarios para aprovecharlas al máximo. Esto abre la puerta a futuros ajustes que amplíen el alcance del chatbot y lo hagan más versátil en distintos contextos.

CAPÍTULO III
MATERIALES Y MÉTODOS

3.1 Tipo y diseño de investigación

Se realizó una investigación aplicada de enfoque mixto.

Aplicada: se persiguió resolver un problema práctico como lo es la dispersión de información acerca de los procesos universitarios de interés adjuntos al portafolio para estudiantes en la universidad mediante la construcción de un chatbot.

Cuantitativa–descriptiva: La evaluación del chatbot incluyó la eficacia, la eficiencia y la satisfacción de quienes lo usaron, aspectos que se midieron conforme a lo establecido en la norma ISO/IEC 25022.

Cualitativa–exploratoria: Los documentos normativos se seleccionaron y adecuaron tomando en cuenta las necesidades expresadas por el personal, con el propósito de integrarlos en el chatbot.

En la construcción del sistema se trabajó con dos marcos de referencia, SCRUM, que encaminó el desarrollo paso a paso, y CRISP-DM, que se aplicó para organizar el flujo de datos y limpiar los documentos de la institución.

3.2 Materiales

Tabla 2 Materiales utilizados para el desarrollo

Fuente: Autoría propia

Categoría	Descripción
Hardware	Computadora Asus Tuf dash F15
	Teléfono móvil con sistema operativo Android
Software base	LangGraph y Python 3.11.
	ChromaDB (base vectorial).
	Oracle
	FastAPI
	Ollama - modelos Llama 3.1
	Docker - Docker Compose
	CUDA
Servicios auxiliares	Para la gestión de la configuración se desarrolló un microservicio con FastAPI, accesible desde Oracle APEX. Como complemento, se diseñó la aplicación móvil “UTN Móvil” en Flutter, pensada para que los usuarios puedan consultar la información desde sus teléfonos.

	Servicio API de OpenAI para el uso de sus LLMs.
Datos y fuentes	Documentos PDF/DOCX institucionales (reglamentos, guías y formatos) y esquema de la base de datos universitaria. La información documental fue provista y validada por el Coordinador de Carrera de Software, Ing. MacArthur Ortega Bustamante.

3.3 Procedimiento

Tabla 3 Fases del desarrollo del proyecto

Fuente: Autoría propia

Fase	Actividades principales	Evidencia/artefacto
Recolección y curación	• Se entrevistó al coordinador de carrera y al líder del macroproyecto "UTN Móvil" para definir el alcance documental. • Se recolectó la información de acuerdo con los requisitos obtenidos, esto a través de entrevistas, solicitudes y descargas de documentación de canales oficiales de la universidad. • Primero se transcribió y resumió la información, después se organizó y se le aplicó el formato correspondiente.	Archivos de documentación transcritos en texto plano.
Ingesta RAG	• Se optó por dividir los textos en fragmentos definidos por parámetros, lo que permitió mantener la coherencia del contenido. • Cada bloque se embebió mediante modelos free de hugin-face y se almacenó en ChromaDB con metadatos de procedencia.	Base de datos en ChromaDB con información estructurada de la documentación de la Universidad Técnica del Norte.
Modelado de agentes	• Agente RAG: El enfoque utilizado integra una búsqueda semántica que recupera hasta cinco resultados, a los	Agentes de inteligencia artificial realizados en LangGraph.

	que se aplican módulos de corrección. Esta lógica se apoya en la arquitectura conocida como self-RAG (Ya mencionada en el marco teórico). • Agente SQL: Combina las peticiones de consulta en una bdd conjunto con módulos correctivos para las peticiones. • Agente Supervisor: Para la clasificación de las preguntas se empleó un LLM con temperatura 0.0, que distingue los tipos de consulta a partir de indicadores como palabras clave y referencias a tablas. • Nodo Chitchat: Funciona con un LLM que recibe de manera directa un prompt construido para transmitir la identidad institucional y poder responder preguntas fuera del corpus pero del contexto universitario.	
Despliegue	• Para organizar la arquitectura, los servicios se encapsularon en contenedores Docker, asignando una imagen distinta a chatbot_api, config_service, Ollama y ChromaDB. • Se definió la orquestación en el archivo docker-compose.yml y variables de entorno.	Paquete desplegado en infraestructura en la nube propuesta por el departamento de tecnologías de la Universidad.
Evaluación ISO/IEC 25022	• En el proceso de evaluación se consideraron tres aspectos: la eficacia, evaluada con la tasa de éxito; la eficiencia, calculada mediante el tiempo medio de respuesta; y la	Sección de evaluación en el presente documento de grado.

	satisfacción, medida con una versión adaptada de la escala SUS.

3.4 Métodos y técnicas de investigación

Tabla 4 Métodos empleados en el desarrollo del proyecto

Fuente: Autoría propia

Método	**Aplicación en el proyecto**
Analítico–sintético	Descomponer requisitos en flujos de agentes y sintetizar la arquitectura final.
Desarrollo iterativo SCRUM	Sprints con duración de 2 semanas - 5 sprints totales.
Pruebas de caja negra y validación por parte de usuarios	Pruebas de funcionalidad y usabilidad aplicadas a estudiantes, documentación de resultados a partir de encuestas.
Técnica documental	Para conformar el corpus del sistema RAG, se revisaron reglamentos y manuales, asegurando que también formaran parte de la base de conocimiento.
Encuesta SUS	Se midió la satisfacción después de un uso controlado del chatbot con estudiantes universitarios.

3.5 Participantes

Tabla 5 Personas externas involucradas en el proyecto

Fuente: Autoría propia

Rol	**Nº**	**Criterios de inclusión**
Estudiantes de Software (UTN)	46	Estudiantes de primero a octavo semestre debidamente matriculados.
Personal administrativo	2	Experiencia en trámites de matrícula, becas, y demás procesos estudiantiles.
Expertos de contenido	2	Haber trabajado en proyectos de inteligencia artificial y en el macroproyecto “UTN Móvil”.

3.6 Instrumentación

3.6.1 Herramientas de medición

- Taller de evaluación.
- Cuestionario SUS (10 ítems, escala Likert 1-5).

- ISO 25022.

3.6.2 Procedimiento de análisis

Las herramientas de medición fueron aplicadas a un total de 46 estudiantes universitarios, todos ellos pertenecientes a los últimos niveles de la carrera de Ingeniería en Software. Esta selección obedeció a dos razones principales: en primer lugar, los estudiantes de niveles avanzados poseen un mayor grado de familiaridad con procesos académicos y con herramientas tecnológicas, lo que garantiza que sus evaluaciones se realicen desde una perspectiva informada; en segundo lugar, se buscó mantener la confidencialidad y el control metodológico del macroproyecto, evitando la inclusión de participantes externos o de niveles iniciales que pudieran introducir sesgos en la percepción del sistema. El proceso se llevó a cabo en 4 fases específicas:

- Se instruyó a los estudiantes sobre la finalidad del proyecto y las limitaciones del mismo.
- Se entregó a cada estudiante los instrumentos necesarios (Taller y encuesta sus en Excel) para registrar las interacciones con el chatbot.
- Se recopiló la información obtenida y se aplicaron pruebas de normalidad en los datos.
- Se aplicaron las mediciones de usabilidad a partir de los datos recolectados y debidamente verificados, para obtener el resultado final de calidad en uso.

3.7 Ética y confidencialidad

En el proceso de recopilación no se incluyó información personal sensible. Además, el manejo de los datos se efectuó siguiendo la política de privacidad de la UTN y su Reglamento de Protección de Datos Personales [58], asegurando así anonimato en la publicación de resultados de este trabajo de grado.

3.8 Limitaciones del estudio

La selección del conjunto de documentos utilizados al inicio del proyecto se basó en archivos proporcionados por el personal administrativo de la universidad en diferentes departamentos de interés estudiantil, dichos documentos están alineados a los requisitos proveídos por el director del macroproyecto de la aplicación “UTN Móvil” para la ingesta de información del chatbot. Esto aseguró que la información con la que se entrenó el sistema fuera confiable y alineada con los procesos institucionales.

Para las pruebas de rendimiento se utilizó únicamente un servidor, de modo que el comportamiento del chatbot se analizó en un escenario limitado y sin repartir la carga

en varios equipos. La opción de escalar horizontalmente la infraestructura quedó pendiente como un posible ajuste en fases siguientes.

En la evaluación de la calidad se tomó como referencia la norma ISO/IEC 25022, que mide la utilidad y efectividad de un sistema. De sus dimensiones, se consideraron tres: efectividad, eficiencia y satisfacción.

CAPITULO IV
DESARROLLO

En el capítulo anterior se describieron, con detalle, los materiales, métodos y el marco metodológico que sustentaron la construcción del chatbot institucional para la Universidad Técnica del Norte (UTN). Habiendo definido el tipo de investigación, los instrumentos y los entornos técnicos empleados, este capítulo expone, de forma cronológica y basada en la gestión ágil de proyectos, el proceso de desarrollo que dio lugar al producto software final.

El trabajo se llevó a cabo con el marco Scrum, en ciclos de 14 días. A lo largo de cinco sprints se fueron desarrollando y probando las partes principales del sistema, lo que permitió recibir comentarios tempranos y documentar cada avance con sus objetivos y resultados.

La arquitectura se diseñó como un chatbot con varios agentes en LangGraph: uno de recuperación apoyado en ChromaDB, otro SQL que consulta la base de la UTN y un supervisor que decide a dónde enviar cada consulta. La solución se dividió en microservicios (config_service y chatbot_api) y se conectó con la aplicación móvil en Flutter que forma parte del macroproyecto "UTN Móvil".

4.1 Análisis

4.1.1 Equipo participante en la metodología SCRUM durante el desarrollo

En la Tabla se describe el equipo involucrado en el desarrollo del proyecto, adjunto al tipo de rol que desempeñó bajo el framework de desarrollo ágil usado.

Tabla 6 Roles del equipo Scrum

Fuente: Autoría propia

Nombre	Rol	Cargo
PhD. Iván García	Product Owner	Director del presente Trabajo de Grado y Docente de la Universidad Técnica del Norte
PhD. Antonio Quiña	Product Owner	Director del Macroproyecto "UTN Móvil" y Docente de la Universidad Técnica del Norte

Brayan de la Cruz	Scrum Máster Equipo de desarrollo	Tesista
PhD. Jorge Caraguay	Stakeholder	Director del DDTI de la Universidad Técnica del Norte

4.1.2 Elicitación de Requisitos

La definición de requisitos se realizó de acorde a la metodología Scrum a partir de historias de usuario redactadas entre el Scrum Máster, Product Owners y StakeHolder. La redacción de estas historias se basó en la plantilla extraída de Scrum Manager Bok [59].

La estimación de complejidad en cada historia se basó en puntos de historia de usuario, definiendo el nivel de complejidad en torno a la serie de Fibonacci, todo según la guía de Scrum Manager [60].

Tabla 7 Historia de usuario #1

Fuente: Autoría propia

Historias de usuario		
Identificador: CTB-01		
Usuario: Estudiante		
Nombre historia: Chatbot inteligente basado en un LLM		
Prioridad: Alta	**Dependencia:** N/A	**Estimación:** 21
Descripción: Como Estudiante, deseo tener acceso a un chatbot moderno impulsado por un modelo de lenguaje (LLM), que no funcione únicamente bajo reglas clave–valor, sino que brinde respuestas más inteligentes y contextuales, para poder obtener información puntual acerca de procesos adjuntos al portafolio estudiantil.		
Prueba de aceptación: • El chatbot debe tener un módulo que le permita recuperar fragmentos de la documentación universitaria. • El chatbot debe poder tomar decisiones basándose en el LLM. • El chatbot podrá responder a saludos y preguntas introductorias informales. • El chatbot podrá informar al usuario cuando este no tenga los medios para responder a la pregunta.		

- El chatbot debe ser escalable, permitiendo el cambio de modelo LLM de forma dinámica según las necesidades institucionales.
- El chatbot debe evitar respuestas incorrectas o inventadas (alucinaciones). Si no tiene información, debe notificarlo explícitamente.
- El chatbot debe responder basándose en documentos y configuraciones controladas por la universidad, evitando opiniones o respuestas fuera de contexto.

Tabla 8 Historia de usuario #2

Fuente: Autoría propia

Historias de usuario
Identificador: CTB-02 **Usuario:** Administrador **Nombre historia:** Sistema de configuración para el chatbot **Prioridad:** Alta **Dependencia:** CTB-01 **Estimación:** 21 **Descripción:** Como administrador, quiero gestionar el chatbot mediante un sistema desarrollado en Oracle APEX, para modificar y mantener su configuración de manera centralizada desde el entorno institucional. **Prueba de aceptación:** • El sistema de configuración debe desarrollarse en Oracle APEX, dentro del entorno tecnológico de la universidad. • Debe contar con una interfaz que permita visualizar, editar y guardar parámetros de configuración del chatbot. • El sistema debe validar los parámetros antes de enviarlos a la API correspondiente del chatbot.

Tabla 9 Historia de usuario #3

Fuente: Autoría propia

Historias de usuario
Identificador: CTB-03 **Usuario:** Administrador **Nombre historia:** Ingestar documento institucional

Prioridad: Alta **Dependencia:** CTB-01, CTB-02 **Estimación:** 13

Descripción: Como administrador, quiero subir un documento institucional relacionado con procesos universitarios al sistema de administración del chatbot, para ampliar su base de conocimiento y mejorar sus respuestas.

Prueba de aceptación:

- El sistema debe mostrar los formatos de archivo compatibles (por ejemplo, PDF, DOCX, TXT).
- Si el archivo no es compatible, el chatbot debe mostrar un mensaje de advertencia.
- Si el documento ya ha sido subido anteriormente, el chatbot debe notificar que se trata de un archivo duplicado.
- Si el archivo excede el tamaño máximo permitido, el chatbot debe rechazar la carga con un mensaje claro.
- Después de una carga exitosa, el chatbot debe mostrar una notificación de confirmación al administrador.

Tabla 10 Historia de usuario #4

Fuente: Autoría propia

Historias de usuario
Identificador: CTB-04
Usuario: Administrador
Nombre historia: Acceso del chatbot a la base de datos Universitaria
Prioridad: Alta **Dependencia:** CTB-01 **Estimación:** 13
Descripción: Como administrador, quiero que el chatbot tenga acceso a la base de datos universitaria, para que pueda responder automáticamente preguntas relacionadas con fechas, convocatorias u otros eventos académicos de manera precisa y actualizada.
Prueba de aceptación:
• El chatbot debe ser capaz de realizar consultas automatizadas a la base de datos universitaria según el requerimiento del estudiante.
• Las respuestas del chatbot deben estar basadas únicamente en la información recuperada desde la base de datos.

- En caso de que no exista información relevante o actual, el chatbot debe evitar alucinaciones y responder con mensajes como: “Lo siento, no poseo información relacionada a tu petición”

Tabla 11 Historia de usuario #5

Fuente: Autoría propia

Historias de usuario		
Identificador: CTB-05		
Usuario: Administrador		
Nombre historia: Administración dinámica de la base de datos utilizada por el chatbot		
Prioridad: Alta	**Dependencia:** CTB-01, CTB-02, CTB-04	**Estimación:** 8
Descripción: Como administrador, quiero gestionar a qué base de datos se conecta el chatbot, pudiendo cambiar la cadena de conexión desde el sistema de configuración, para así adaptarlo fácilmente a diferentes entornos o necesidades institucionales.		
Prueba de aceptación: • El sistema debe permitir al administrador cambiar la base de datos SQL que utiliza el chatbot mediante la interfaz del sistema de configuración. • El chatbot debe utilizar la base de datos actualmente configurada, y dejar de acceder automáticamente a la anterior.		

Tabla 12 Historia de usuario #6

Fuente: Autoría propia

Historias de usuario		
Identificador: CTB-06		
Usuario: Administrador		
Nombre historia: Administración del acceso del chatbot a las tablas de la base de datos		
Prioridad: Alta	**Dependencia:** CTB-01, CTB-02, CTB-04, CTB-05	**Estimación:** 5
Descripción: Como administrador, deseo gestionar a qué tablas de la base de datos puede acceder el chatbot, permitiendo especificar por nombre de tabla y una breve		

descripción, para garantizar un control seguro y específico sobre la información expuesta al sistema.

Prueba de aceptación:

- El chatbot solo podrá consultar las tablas que hayan sido explícitamente descritas por el administrador.
- El chatbot debe impedir el acceso a cualquier tabla que no haya sido registrada previamente.
- El formulario de autorización debe permitir ingresar:
  Nombre de la tabla.
  Descripción breve del contenido la misma.
- El sistema debe mostrar una lista de las tablas registradas, con opción de edición o eliminación.

Tabla 13 Historia de usuario #7:

Fuente: Autoría propia

Historias de usuario
Identificador: CTB-07
Usuario: Administrador
Nombre historia: Configurar modelo LLM
Prioridad: Alta **Dependencia:** CTB-01 **Estimación:** 3
Descripción: Como administrador, quiero seleccionar la IA "grande" que usará el chatbot, desde el sistema, para poder adaptar el chatbot según mi infraestructura y necesidades.
Prueba de aceptación: • Cuando el administrador envíe una configuración válida, el chatbot debe actualizarse y usar el modelo de IA indicado (por ejemplo: GPT-4, LLaMA 3.1, etc.). • Si los parámetros enviados están incompletos o en un formato equivocado, el chatbot debe mostrar un mensaje de advertencia indicando el error específico. • El administrador debe recibir una confirmación visual de que el cambio de IA se ha realizado correctamente.

Tabla 14 Historia de usuario #8

Fuente: Autoría propia

Historias de usuario
Identificador: CTB-08 **Usuario:** Estudiante **Nombre historia:** Aplicativo móvil para el chatbot **Prioridad:** Alta **Dependencia:** CTB-01 **Estimación:** 8 **Descripción:** Como estudiante, quiero hablar con el chatbot desde mi celular, para pedir ayuda desde cualquier lugar y en cualquier momento. **Prueba de aceptación:** • El aplicativo del chatbot debe estar incluido como módulo dentro de la app "UTN Móvil". • El usuario podrá enviar un mensaje por turno, y el aplicativo deshabilitará el input hasta recibir la respuesta del chatbot. • La interfaz debe ser limpia, agradable e intuitiva, siguiendo los lineamientos visuales institucionales. • El chatbot responderá al usuario cuando no esté disponible dando un mensaje de error claro.

Tabla 15 Historia de usuario #9

Fuente: Autoría propia

Historias de usuario
Identificador: CTB-09 **Usuario:** Estudiante **Nombre historia:** Exportar el historial del chat a PDF **Prioridad:** Alta **Dependencia:** CTB-08 **Estimación:** 3 **Descripción:** Como estudiante, quiero exportar el historial visible del chat con el chatbot a un archivo PDF, mediante un botón en la interfaz, para tener un respaldo descargable de mi conversación. **Prueba de aceptación:** • Debe existir un botón claramente identificado e intuitivo para exportar el historial de conversación. • Al hacer clic, el sistema debe generar una previsualización del contenido del chat en formato PDF antes de su descarga.

- El historial exportado debe reflejar únicamente los mensajes visibles en pantalla.
- El PDF generado debe mostrar quién escribió cada mensaje.

Tabla 16 Historia de usuario #10

Fuente: Autoría propia

Historias de usuario
Identificador: CTB-10 **Usuario:** Estudiante **Nombre historia:** Borrar conversación en pantalla **Prioridad:** Alta **Dependencia:** CTB-08 **Estimación:** 2 **Descripción:** Como estudiante, quiero borrar la conversación visible en pantalla con el chatbot mediante un botón, para mantener la interfaz limpia y enfocarme en nuevas consultas sin distracciones. **Prueba de aceptación:** • Debe existir un botón visible e intuitivo para borrar la conversación. • Al presionar el botón, el aplicativo debe eliminar todos los mensajes visibles del historial en pantalla. • Debe mantenerse únicamente el mensaje de bienvenida o saludo del chatbot (si aplica).

Tabla 17 Historia de usuario #11

Fuente: Autoría propia

Historias de usuario
Identificador: CTB-11 **Usuario:** Director del Departamento de Tecnologías **Nombre historia:** Control de redundancia en almacenamiento de mensajes **Prioridad:** Alta **Dependencia:** CTB-01, CTB-08 **Estimación:** 3 **Descripción:** Como Director de Tecnologías de la universidad, deseo que el chatbot no almacene las conversaciones en bases de datos institucionales, sino que únicamente

se guarden localmente las últimas 10 interacciones en la aplicación, para preservar la confidencialidad, reducir el uso de recursos y evitar sobrecarga en la infraestructura.

Prueba de aceptación:

- No se deben almacenar las conversaciones del chatbot en ninguna base de datos remota o centralizada.
- Solo las últimas 10 interacciones deben conservarse localmente en la aplicación (memoria o almacenamiento interno del dispositivo).
- En caso de desinstalación o reinicio de la aplicación, las interacciones anteriores deben desaparecer automáticamente, sin posibilidad de recuperación.

Tabla 18 Historia de usuario #12

Fuente: Autoría propia

Historias de usuario			
Identificador: CTB-12			
Usuario: Director del Departamento de Tecnologías			
Nombre historia: Despliegue del chatbot			
Prioridad: Alta	**Dependencia:** CTB-01, CTB-02, CTB-04	**Estimación:** 8	

Descripción: Como Director de Tecnologías de la UTN quiero que el chatbot esté desplegado sobre la infraestructura estandarizada de la universidad para mantener todas sus dependencias controladas internamente y facilitar el soporte técnico futuro.

Prueba de aceptación:

- Todo el código del chatbot debe estar dockerizado, con sus archivos importantes bien documentados.
- El sistema debe estar dividido en dos APIs separadas:
  Uno para la configuración y administración (por ejemplo, selección de modelo, conexión a BD).
  Otro para el consumo del chatbot por parte de las aplicaciones (como UTN Móvil u otro consumo).
- No debe haber dependencia directa con servicios de terceros externos sin autorización institucional.

Tabla 19 Historia de usuario #13

Fuente: Autoría propia

Historias de usuario		
Identificador: CTB-13		
Usuario: Product Owner		
Nombre historia: Ingesta de documentación al chatbot sobre procesos del portafolio estudiantil		
Prioridad: Alta	**Dependencia:** CTB-01, CTB-02, CTB-03	**Estimación:** 13
Descripción: Como Product Owner, deseo que el chatbot incluya documentación clave relacionada con los procesos gestionados en el portafolio estudiantil, para que pueda responder adecuadamente desde una base de conocimientos inicial bien estructurada.		
Prueba de aceptación: • La documentación que se ingrese al chatbot debe seguir la metodología CRISP-DM para la limpieza y formato de su contenido. • Se debe aplicar un formato estandarizado a todos los documentos. • La documentación debe ser visible a través del sistema de configuración.		

Tabla 20 Historia de usuario #14

Fuente: Autoría propia

Historias de usuario		
Identificador: CTB-14		
Usuario: Administrador		
Nombre historia: Creación del manual de uso del chatbot		
Prioridad: Alta	**Dependencia:** CTB-01-CTB-13	**Estimación:** 3
Descripción: Como administrador, deseo contar con un manual sencillo y práctico sobre el uso del chatbot, para poder orientarme en su operación, configuración y funcionalidades principales.		
Prueba de aceptación: • El manual debe estar disponible en formato PDF y alojado dentro del sistema o portal institucional. • El lenguaje debe ser claro, sin tecnicismos innecesarios, entendible para personas que no tengan conocimientos en programación.		

4.1.3 Creación del Product Backlog

En la Tabla 21 se presenta la organización de las historias de usuario, establecida según su prioridad por el Product Owner, en función de los objetivos y necesidades definidas para el proyecto.

Tabla 21 Product Backlog

Fuente: Autoría propia

Id	Nombre Historia	Estimación	Orden
CTB-01	Chatbot inteligente basado en un LLM.	21	01
CTB-03	Ingestar documento institucional.	13	02
CTB-04	Acceso del chatbot a la base de datos Universitaria.	13	03
CTB-02	Sistema de configuración para el chatbot.	21	04
CTB-05	Administración dinámica de la base de datos utilizada por el chatbot.	2	05
CTB-06	Administración del acceso del chatbot a las tablas de la base de datos.	3	06
CTB-07	Configurar modelo LLM.	3	07
CTB-08	Aplicativo móvil para el chatbot.	13	08
CTB-09	Exportar el historial del chat a PDF.	3	09
CTB-10	Borrar conversación en pantalla.	2	10
CTB-11	Control de redundancia en almacenamiento de mensajes.	3	11
CTB-13	Ingesta de documentación al chatbot sobre procesos del portafolio estudiantil.	21	12
CTB-12	Despliegue del chatbot.	13	13
CTB-14	Creación del manual de uso del chatbot.	3	14

4.2 Sprint 1: Diseño de la arquitectura del chatbot

En este primer sprint se establecieron los componentes fundamentales del sistema, incluyendo la arquitectura del chatbot, la definición de sus agentes principales, la integración con inteligencia artificial y una base de datos vectorial, así como la configuración inicial del entorno de desarrollo.

4.2.1 Resumen correspondiente al Sprint 1

Durante la reunión de planificación llevada a cabo el 07/04/2025, con la participación del Product Owner, Scrum Master y el Equipo de Desarrollo, se definieron las tareas correspondientes al Sprint 1, centradas en el diseño de la arquitectura del chatbot, la estimación de cada tarea se rige a la metodología scrum, dividiendo el esfuerzo total de las historias de usuario en cada una de estas (ver Tabla 22).

Tabla 22 Tareas del Sprint 1

Fuente: Autoría propia

Historia	Tarea	Estimación
CTB-01	Definición de arquitectura del agente de conversación del chatbot.	5
CTB-04	Definición de arquitectura del agente de conexión a la bdd universitaria del chatbot.	5
CTB-01	Creación de la estructura base del proyecto de acuerdo con las arquitecturas definidas.	1
CTB-01	Configuración de las dependencias del proyecto y creación de variables de entorno.	1
CTB-01	Instanciamiento de modelos de IA (LLM y modelo de IA para embeddings) y conexión de estos con el proyecto.	1
CTB-03	Instanciamiento de la base de datos a utilizarse para la base de conocimiento del chatbot y conexión de esta con el proyecto.	3
CTB-01	Configuración de herramientas para la depuración de los módulos del chatbot.	1

Posteriormente, en la reunión de revisión y retrospectiva realizada el 18/04/2025, con la presencia de los mismos actores, se obtuvo como resultado el incremento entregable correspondiente a la arquitectura tecnológica del sistema, su comunicación con la base de datos universitaria, su base de conocimientos y el diagrama de procesos vinculados a los componentes principales del chatbot.

4.2.2 Arquitectura de tecnologías del chatbot

El sistema fue construido con una arquitectura modular en Python, donde LangGraph cumple el papel de eje central. En el corazón del diseño se encuentra el enfoque RAG, que aprovecha LangChain para coordinar los modelos de lenguaje - ya sea

que se ejecuten de forma local con Ollama o en la nube - junto con la base vectorial ChromaDB. Esto le permite recuperar fragmentos de documentos previamente preparados y generar respuestas ajustadas al contexto de cada consulta.

Para complementar estas capacidades, el sistema se conecta de manera directa con la base de datos Oracle, que contiene información de la universidad en constante actualización. Los accesos y configuraciones no se manejan de forma manual, sino a través de un servicio especializado que administra credenciales y parámetros de conexión de manera independiente.

Todo este conjunto de funciones se hace accesible mediante un servicio web basado en FastAPI. Gracias a ello, es posible utilizar la solución tanto desde una interfaz en Oracle APEX, pensada para la gestión administrativa y la carga de documentos, como desde una aplicación móvil en Flutter, destinada a los estudiantes para realizar consultas.

En la Figura 3 ya se ha detallado la composición de dependencias, la Figura 6 a continuación sintetiza esta arquitectura, destacando los módulos principales, los flujos de datos y los puntos de interacción con el usuario final.

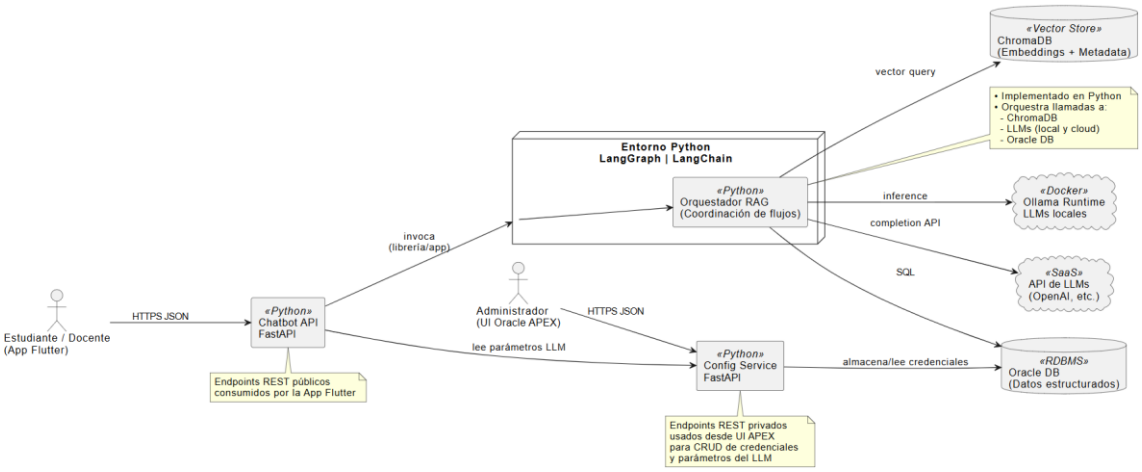


Fig. 6 Diagrama de arquitectura tecnológica

Fuente: Autoría propia

4.2.3 Diagrama del agente RAG

El chatbot se compone de varios agentes de inteligencia artificial, los cuales trabajan en conjunto para ofrecer respuestas precisas. El agente RAG se implementa como un grafo de estados en LangGraph, donde cada nodo encapsula una tarea especializada y las transiciones se controlan mediante módulos de decisión automáticos. La Figura 7 muestra la conexión de cada nodo, y en la tabla 23 se explica brevemente la finalidad de cada uno.

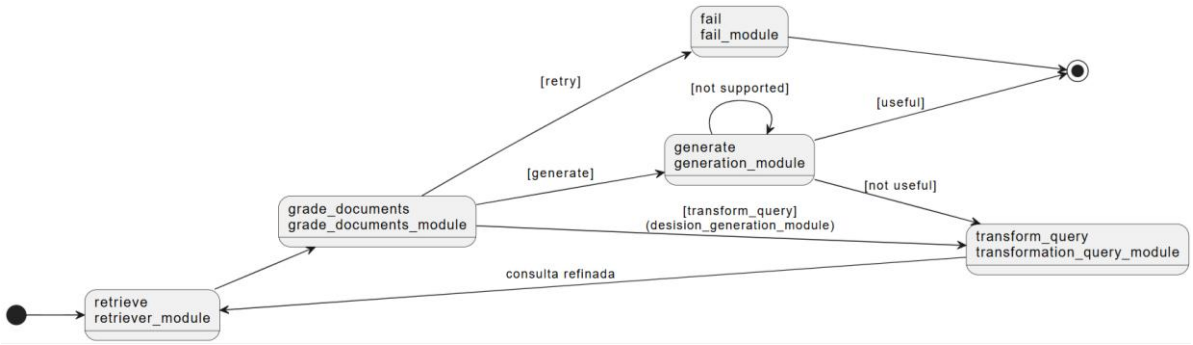


Fig. 7 Diagrama del agente de inteligencia Artificial RAG

Fuente: Autoría propia

Tabla 23 Explicación de la arquitectura del agente de inteligencia artificial RAG

Fuente: Autoría propia

Estado / Módulo	Función principal
retrieve (retriever_module)	Recupera los documentos más relevantes desde ChromaDB en función de la consulta del usuario.
grade_documents (grade_documents_module)	Evalúa la pertinencia de los documentos recuperados y decide el siguiente paso mediante desision_generation_module.
generate (generation_module)	Redacta una respuesta preliminar con un LLM (local u OpenAI). La salida se valida con grade_generation_v_documents_and_question_module.
transform_query (transformation_query_module)	Reformula la pregunta si la cobertura documental es insuficiente; el flujo regresa a retrieve.
fail (fail_module)	Devuelve un mensaje de error o reintento tras agotar alternativas.

4.2.4 Diagrama del agente SQL

El agente SQL se modela en LangGraph como un flujo de estados encargado de traducir preguntas en lenguaje natural a consultas SQL, ejecutarlas sobre Oracle DB y presentar los resultados al usuario. Cada nodo corresponde a un módulo especializado; las bifurcaciones se controlan mediante módulos supervisores que validan la información

disponible o la corrección de la consulta. La Figura 8 muestra la conexión de cada nodo, y la Tabla 24 muestra el funcionamiento de cada uno interconectado.

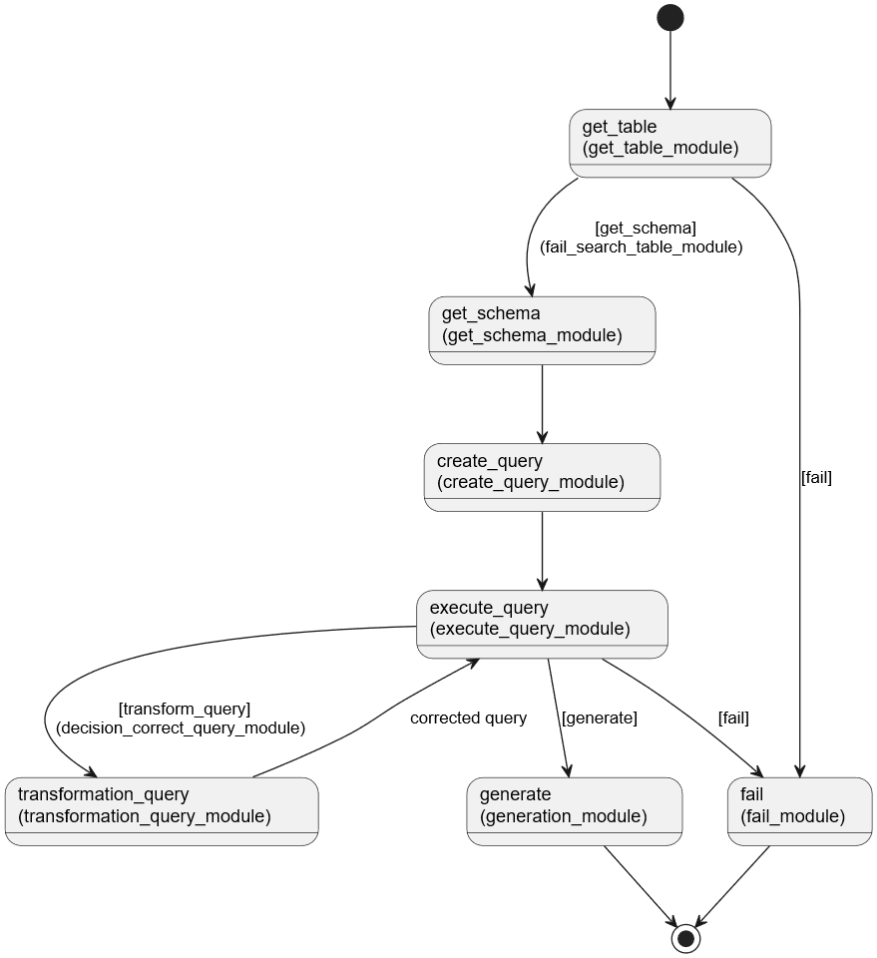


Fig. 8 Diagrama del agente de inteligencia Artificial SQL

Fuente: Autoría propia

Tabla 24 Explicación de la arquitectura del agente de inteligencia artificial RAG

Fuente: Autoría propia

Estado / Módulo	Función principal
get_table (get_table_module)	Encuentra las tablas relevantes para la pregunta del usuario.
get_schema (get_schema_module)	Recupera el esquema (columnas y tipos) de las tablas seleccionadas.
create_query (create_query_module)	Crea la consulta SQL inicial a partir de la pregunta y el esquema.

execute_query (execute_query_module)	Ejecuta la consulta en la base de datos y devuelve el resultado.
transformation_query (transformation_query_module)	Corrige o rehace la consulta si se detectan errores de sintaxis o de lógica.
generate (generation_module)	Crea la respuesta final, mostrándolos de manera no técnica al usuario.
fail (fail_module)	Devuelve un mensaje de error cuando no se encuentran tablas o la consulta no es ejecutable.

4.2.5 Diagrama del agente Supervisor

El Supervisor actúa como orquestador de la conversación. Mediante detección de intención decide cuál sub-agente (SQL, RAG o Chit-Chat) debe gestionar la pregunta del usuario. Cada sub-agente es un grafo independiente ya compilado, de modo que para el Supervisor son "cajas negras" que terminan en una respuesta o en un fallo controlado. En la Figura 9 se muestran los grafos interconectados a través de un super grafo, además de otros nodos complementarios que ayudan al funcionamiento del agente, la funcionalidad de cada nodo interconectado se explica en la Tabla 25.

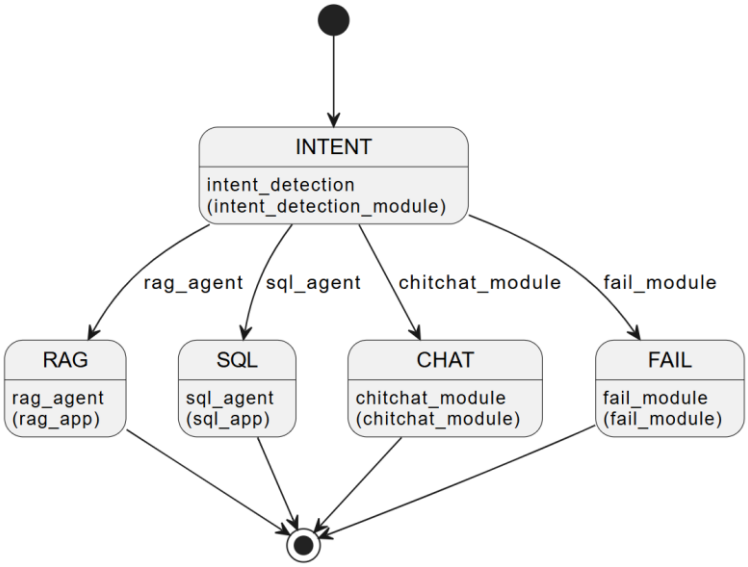


Fig. 9 Diagrama del agente Supervisor

Fuente: Autoría propia

Tabla 25 Explicación de la arquitectura del agente Supervisor

Fuente: Autoría propia

Estado / Módulo	Función principal
intent_detection (intent_detection_module)	Clasifica la intención de la pregunta y produce la etiqueta sql, rag o chitchat.
sql_agent (sql_app)	Encapsula el flujo SQL (consulta a Oracle) cuando la intención requiere de datos estructurados.
rag_agent (rag_app)	Es el agente RAG construido anteriormente y detallado en la Figura 7.
chitchat_module (chitchat_module)	Genera respuestas casuales, pero que mantienen el ambiente universitario. Sirve para mantener un ambiente "Humano".
fail_module (fail_module)	Devuelve un mensaje de error cuando no se puede identificar una intención válida.

4.2.6 Estructura y comunicación entre carpetas

Para garantizar que el chatbot evolucione sin fricciones y se integre con el macroproyecto móvil, se adoptó una estructura de carpetas estrictamente modular. Cada directorio aborda un único dominio (agentes, nodos, prompts, recursos, configuración, servicios), lo que aisla responsabilidades, evita efectos colaterales y permite agregar o reemplazar componentes - por ejemplo, nuevos LLMs o bases vectoriales - modificando solo la carpeta pertinente. Esta organización fomenta la reusabilidad (prompts y utilidades compartidas), facilita las pruebas unitarias al mantener nodos independientes, y habilita la portabilidad entre entornos. Además, los servicios FastAPI se despliegan como microservicios autónomos, favoreciendo la escalabilidad y la colaboración de equipos multidisciplinares; todo ello deja la puerta abierta a futuras extensiones (nuevos back-ends, dashboards o métricas) sin tocar el núcleo conversacional. La estructura se describe mejor en la Figura 10, y la descripción del contenido de cada componente en la Tabla 26.

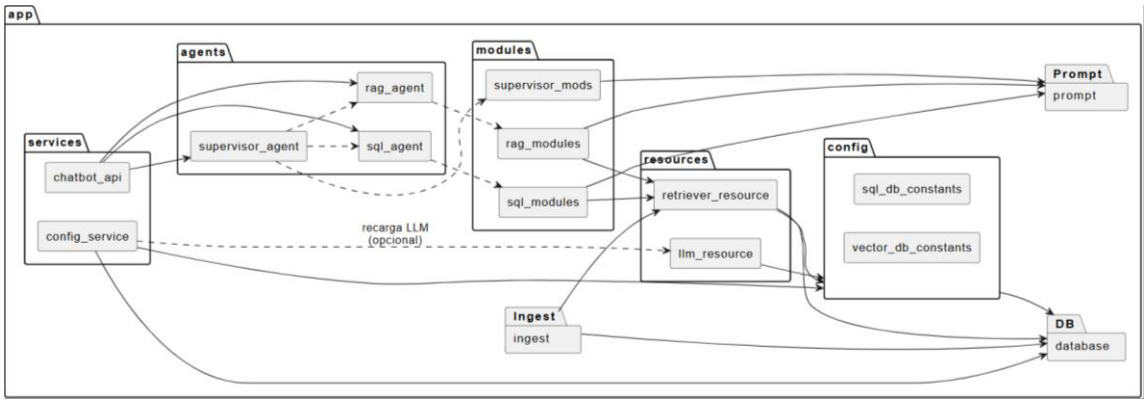


Fig. 10 Diagrama de comunicación entre carpetas y recursos

Fuente: Autoría propia

Tabla 26 Explicación del contenido de cada carpeta

Fuente: Autoría propia

Capa / Carpeta	**Propósito esencial**
agents/	Define los grafos LangGraph de alto nivel (rag_agent, sql_agent, supervisor_agent) que orquestan la conversación.
modules/	Contiene la lógica central de cada agente de langgraph; cada archivo es un nodo independiente (retrieval, grading, generación, etc.).
prompt/	Almacena plantillas de texto que guían al LLM; separa claramente contenido estático de la lógica.
resources/	Código de utilidades que se comparten (retriever de Chroma, instanciador de LLM, memoria); actúa como "caja de herramientas".
config/	Centraliza la inicialización de componentes de cada agente y le asigna un valor según se requiera (Oracle, Chroma).
database/	Contiene la lógica CRUD con Oracle/Chroma; Esto ayuda a aislar la capa de persistencia.
ingest/	Contiene la lógica para el ingreso de documentación al chatbot.
services/	Contiene la lógica de las API (microservicios) que se exponen del chatbot, tanto para ser consumido como configurado.

4.2.7 Herramientas de depuración para el chatbot

Gracias a las capacidades proporcionadas por LangGraph, es posible realizar la depuración individual de cada agente utilizando herramientas integradas, lo que permite validar el funcionamiento de sus nodos de forma controlada y eficiente.

4.3 Sprint 2: Desarrollo de módulos principales de conversación

Este sprint estuvo dedicado a darle forma a la lógica conversacional del chatbot. Se conectó con ChromaDB como base de conocimiento, se preparó el proceso de ingesta de documentos y se crearon los primeros componentes para recuperar información, decidir respuestas, generarlas y manejar posibles errores. Con ello, el sistema comenzó a adquirir la capacidad de responder en contexto, un punto de partida esencial para su funcionamiento completo.

4.3.1 Reunión de planificación y Sprint Backlog – Sprint 2

A partir del Sprint 2 se adopta un enfoque orientado al desarrollo progresivo, en el cual se hace necesario registrar los avances obtenidos en función del objetivo principal del chatbot.

Durante la reunión de planificación llevada a cabo el 21/04/2025, con la participación del Product Owner, Scrum Master y el Equipo de Desarrollo, se obtuvo como resultado el Sprint Backlog correspondiente al Sprint 2 el cual se detalla en la Tabla 27.

Tabla 27 Sprint Backlog – Sprint 2

Fuente: Autoría propia

Historia de usuario	Nombre	Tarea	Estimación
CTB-03	Ingestar documento institucional.	Desarrollo de interacción del proyecto con la base de conocimientos (cofiguración de Chromadb).	5
CTB-03	Ingestar documento institucional.	Desarrollo de funcionalidad para la ingesta de documentación a la base de datos de conocimiento.	5

CTB-01	Chatbot inteligente basado en un LLM.	Desarrollo del módulo recuperador de información de la base de conocimiento (retriever).	1
CTB-01	Chatbot inteligente basado en un LLM.	Desarrollo del módulo de toma de desiciones del chatbot	1
CTB-01	Chatbot inteligente basado en un LLM.	Desarrollo del módulo de conversación informal del chatbot (chitchat)	1
CTB-01	Chatbot inteligente basado en un LLM.	Desarrollo del módulo de generación de respuesta del chatbot	1
CTB-01	Chatbot inteligente basado en un LLM.	Desarrollo del módulo de control de fallos de respuestas	1

4.3.2 Reunión de revisión– Sprint 2

Durante la reunión de revisión realizada el 02/05/2025, con la participación del Product Owner, Scrum Máster y el Equipo de Desarrollo, se llevó a cabo la evaluación del Sprint Backlog y la validación de las pruebas de aceptación, con el fin de comprobar el cumplimiento de los requisitos asociados a las historias de usuario definidas (Véase en la Tabla 28).

Tabla 28 Pruebas de aceptación - Sprint 2

Fuente: Autoría propia

Historia de usuario	Nombre	Funcionalidad	Aceptación
CTB-03	Ingestar documento institucional.	Comunicación con la base de datos en ChromaDB	Si
		Ingesta de documentación a la base de datos de conocimiento.	Si
CTB-01	Chatbot inteligente basado en un LLM.	Módulo de recuperación para seguir el enfoque RAG.	Si

	Módulo de toma de decisiones basándose en el LLM	Si
	Módulo de conversación informal del chatbot (chitchat)	Si
	Módulo de generación de respuesta del chatbot	Si
	Módulo de control de fallos de respuestas	Si

El incremento relacionado al proyecto es explicado a continuación en relación con cada funcionalidad implementada y validada:

- Se implementó la lógica CRUD que permite la comunicación entre el código en Python y la base de datos vectorial ChromaDB. La Figura 11 muestra la instancia de ChromaDB ejecutándose en Docker; la Figura 12 presenta el código correspondiente a la conexión con la base de datos vectorial; y la Figura 13 expone los métodos CRUD desarrollados para su gestión.

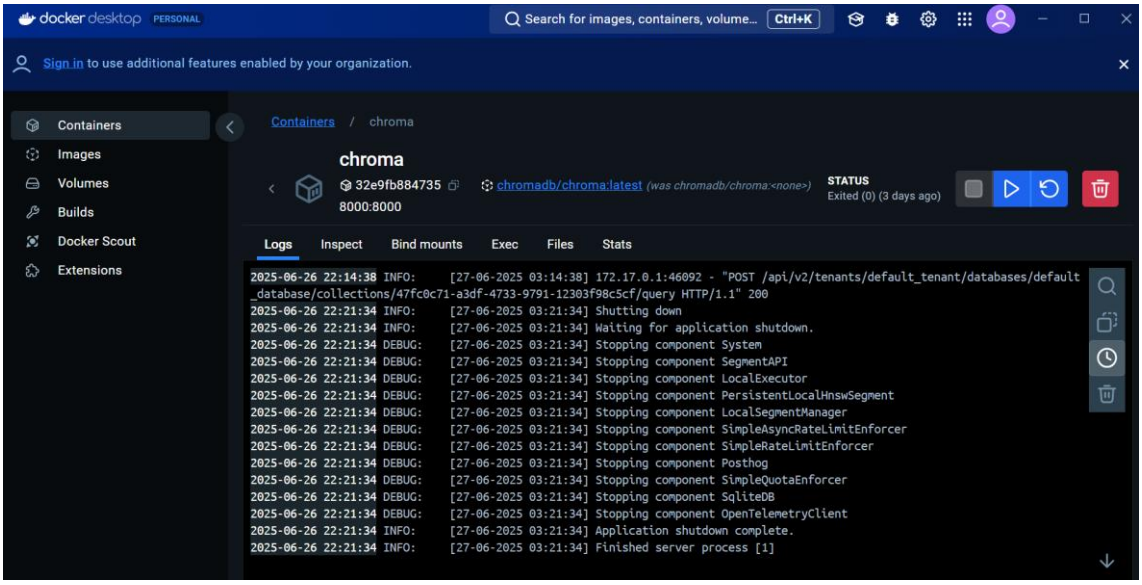


Fig. 11 Imagen de ChromaDB local en Docker

Fuente Autoría Propia

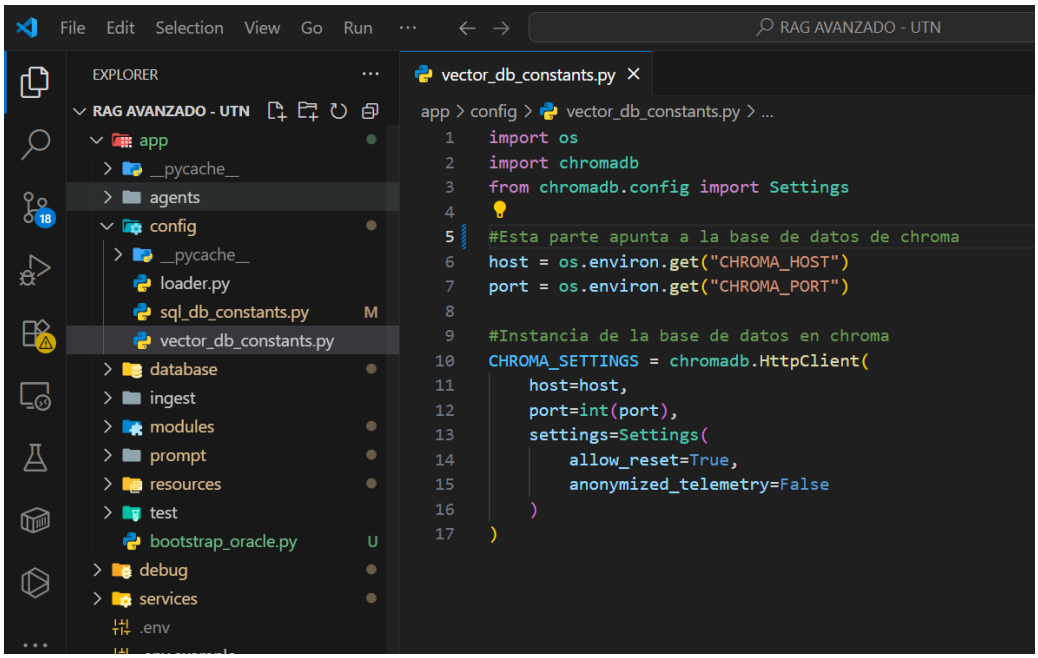


Fig. 12 Código de conexión del proyecto con la base de datos en ChromaDB

Fuente: Autoría propia

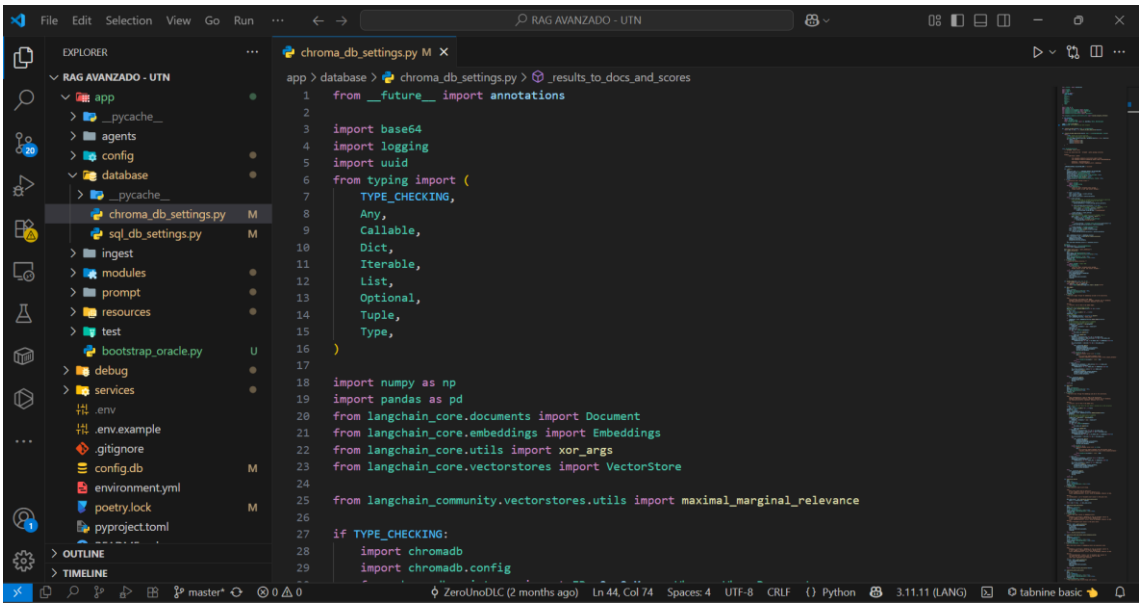


Fig. 13 Código con la lógica CRUD para el manejo de ChromaDB

Fuente: Autoría propia

- La lógica de ingesta de documentación en texto plano permite incorporar información al chatbot mediante la conexión con la base de datos vectorial. Para ello, se utilizan métodos que procesan los textos a través de un modelo de embeddings, transformando su contenido semántico en representaciones

vectoriales que luego son almacenadas en ChromaDB. La Figura 14 muestra el código correspondiente a este proceso de ingestión.

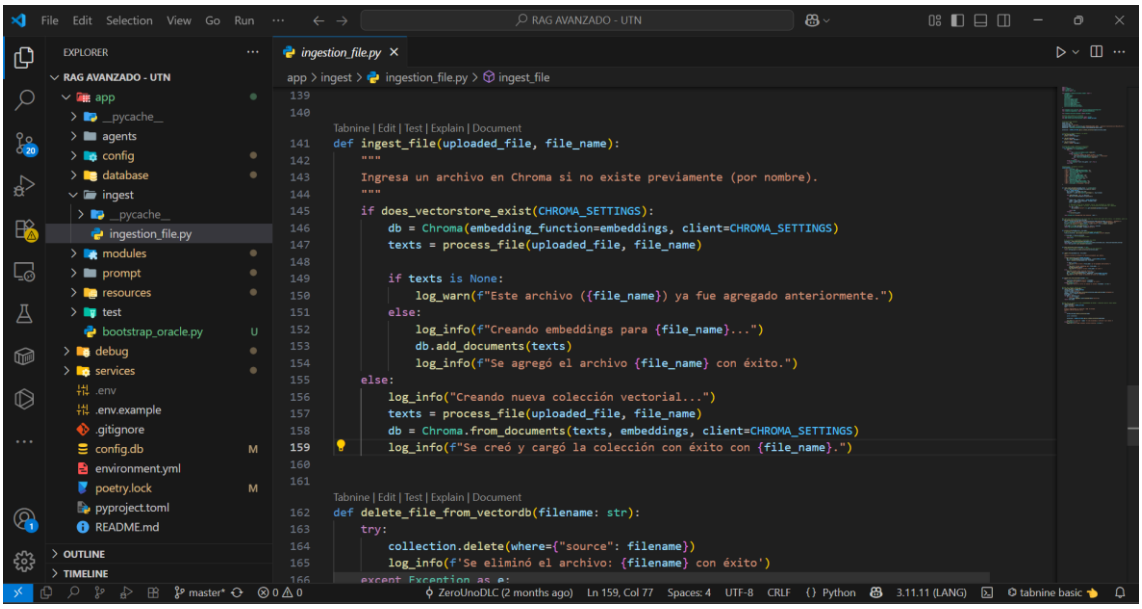


Fig. 14 Código con la lógica de ingestión a la base de datos en ChromaDB

Fuente: Autoría propia

- El módulo de recuperación del chatbot transforma la pregunta ingresada en una representación vectorial y la compara con los vectores almacenados en la base de datos, recuperando los k resultados más relevantes. El valor de k es configurable; en este proyecto se ha definido 5 como el valor óptimo. La Figura 15 muestra el retriever encargado de esta conversión y búsqueda, mientras que la Figura 16 indica el nodo correspondiente dentro del diagrama general del Agente RAG al que este módulo pertenece.

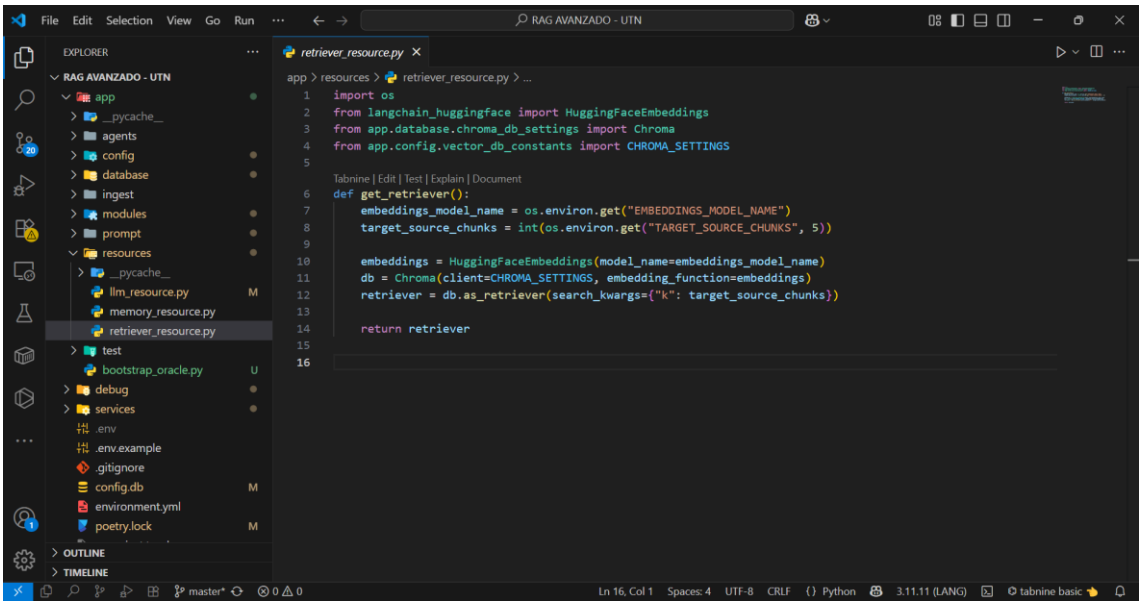


Fig. 15 Código del retriever para el agente RAG del chatbot

Fuente: Autoría propia

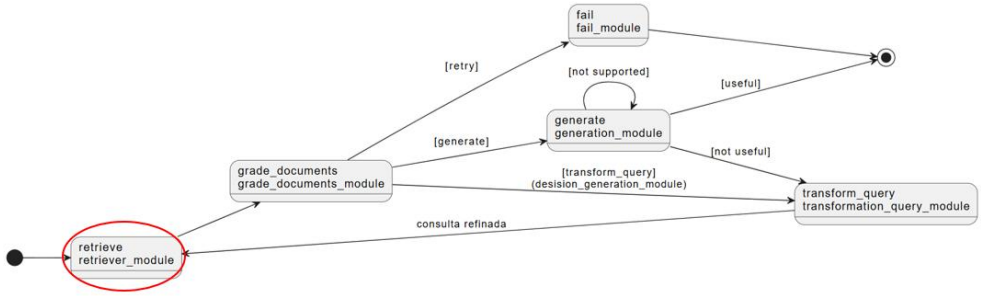


Fig. 16 Nodo desarrollado correspondiente al agente RAG

Fuente: Autoría propia

- El módulo de toma de decisiones utiliza al LLM para elegir entre dos rutas posibles a través del contexto proveído, centraliza las rutas que puede tomar el chatbot dependiendo de cada agente. En la Figura 17 se muestra el código de este módulo.

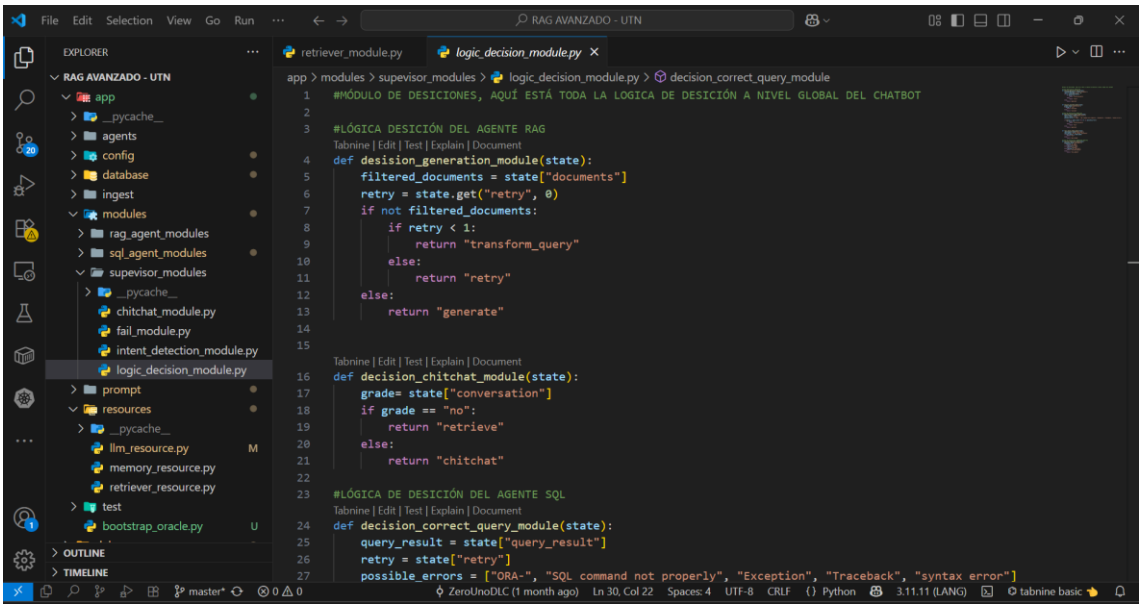


Fig. 17 Código del módulo de toma de decisiones del chatbot

Fuente: Autoría propia

- El módulo de conversación informal es un componente sencillo que proporciona contexto de rol al LLM, permitiéndole responder al estudiante como un asistente universitario ante saludos, preguntas introductorias e interacciones informales. La Figura 18 muestra el prompt básico utilizado para asignar este rol al modelo, mientras que la Figura 19 presenta el módulo correspondiente dentro del agente Supervisor.

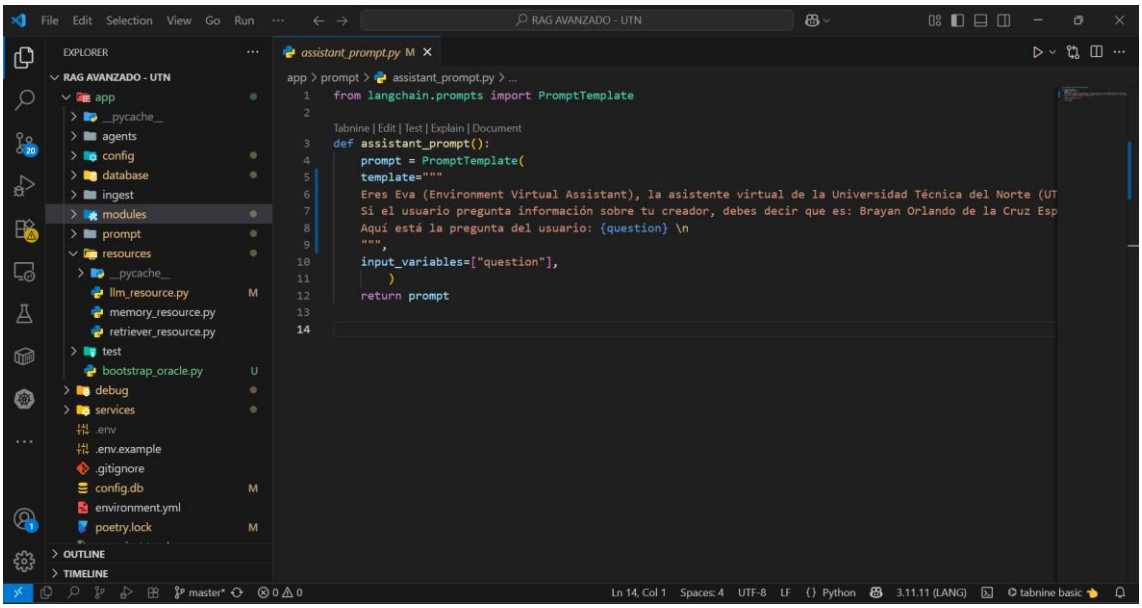


Fig. 18 Prompt para el módulo de conversación informal

Fuente: Autoría propia

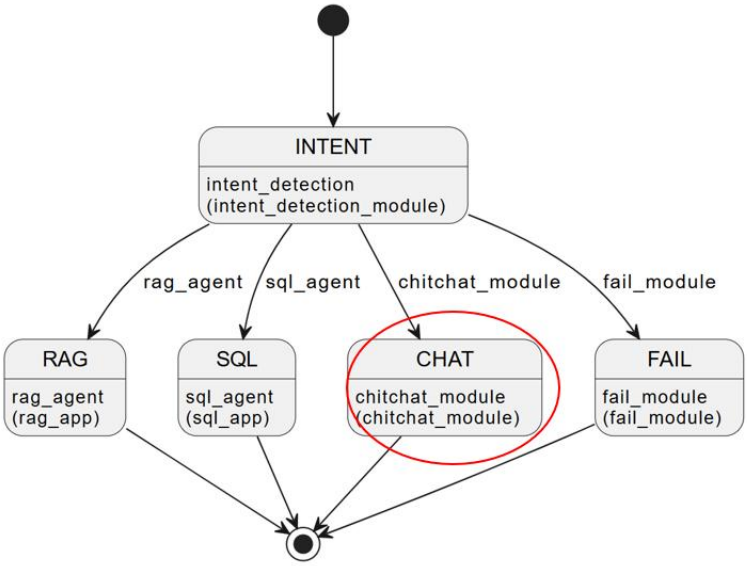


Fig. 19 Nodo desarrollado correspondiente al agente Supervisor

Fuente: Autoría propia

- El módulo de generación de respuestas utiliza el contexto proporcionado por los módulos previos, junto con el rol de asistente universitario, para generar una respuesta adecuada al usuario. La Figura 20 muestra el código de este módulo, incluyendo la construcción de su cadena mediante LangChain, mientras que la Figura 21 presenta el nodo correspondiente dentro del agente RAG.

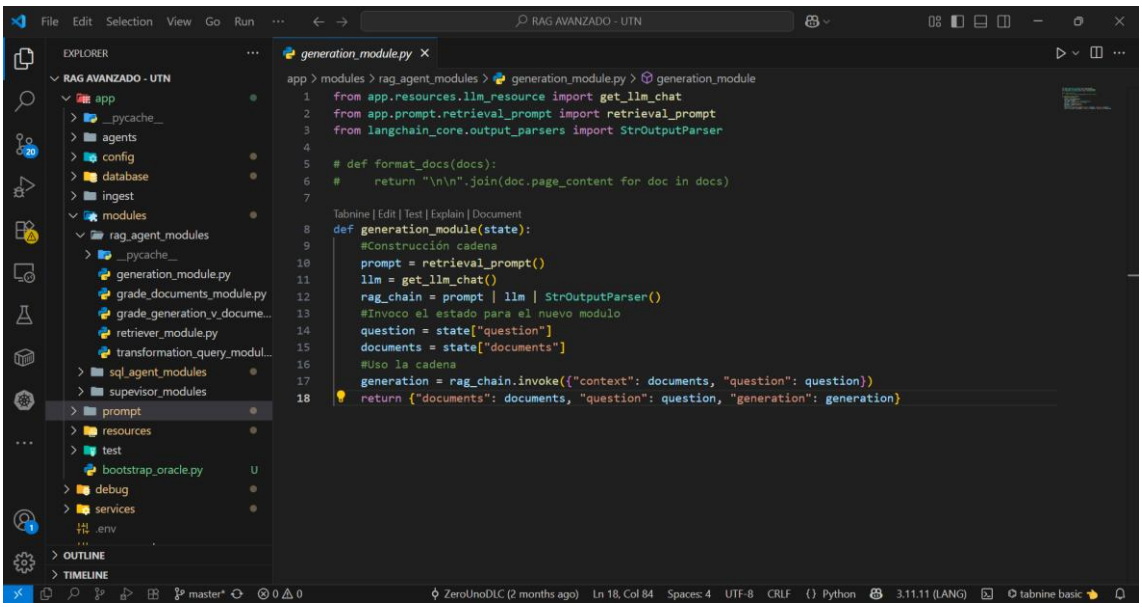


Fig. 20 Código del módulo de generación de respuestas del agente RAG

Fuente: Autoría propia

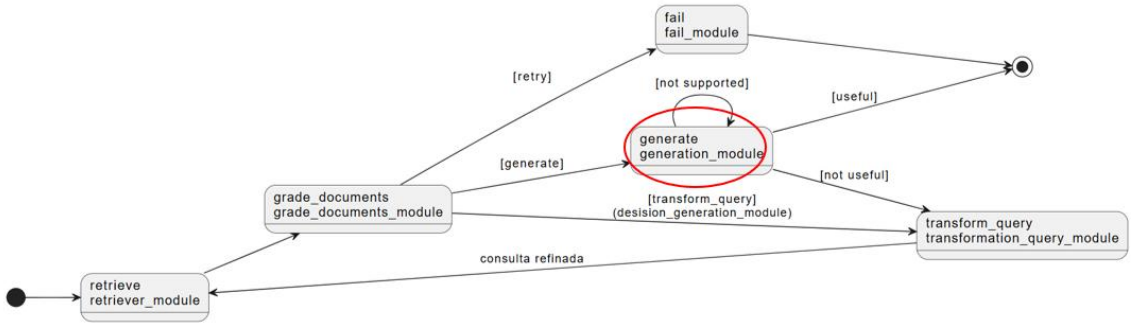


Fig. 21 Nodo desarrollado correspondiente al agente RAG

Fuente: Autoría propia

- El módulo de control de fallos es un componente sencillo que emite una respuesta directa al usuario cuando, durante el proceso de decisión, se determina que no es posible continuar con el flujo conversacional. La Figura 22 muestra el módulo desarrollado correspondiente al agente Supervisor.

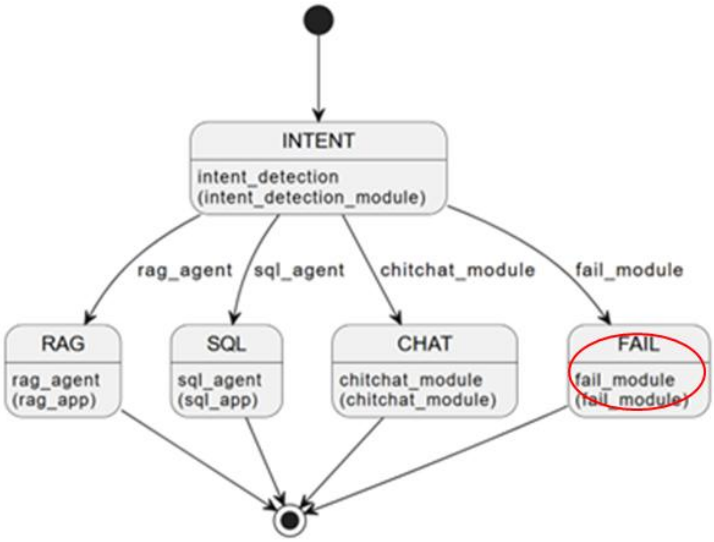


Fig. 22 Nodo desarrollado correspondiente al agente Supervisor

Fuente: Autoría propia

4.3.3 Reunión de retrospectiva – Sprint 2

Durante la reunión de retrospectiva realizada el 02/05/2025, con la participación del Product Owner, Scrum Máster y el Equipo de Desarrollo, se obtuvo como resultado el plan de mejora, en el cual se definen tres puntos importantes:

- **Aciertos:** lo que se ha alcanzado satisfactoriamente durante el desarrollo del Sprint.

- **Problemas:** los inconvenientes que se han presentado durante el Sprint.
- **Mejoras:** las mejoras que se implementarán.

El plan de mejora correspondiente al Sprint 2 es evidenciado en la Tabla 29:

Tabla 29 Plan de mejora - Sprint 2

Fuente: Autoría propia

Plan de mejora	
Aciertos	• Conexión del proyecto a la base de datos vectorial correctamente. • Construcción exitosa de los nodos de langgraph para la creación de los agentes: RAG, Supervisor.
Problemas	• Dificultades iniciales en la configuración y ejecución de ChromaDB en entorno Docker local. • Complejidad en la comunicación entre nodos desarrollados.
Mejoras	• Capacitación en arquitecturas recomendadas por LangGraph para construcción de agentes óptimos. • Métodos de recuperación en la base de datos de ChromaDB óptimos. • Agentes del chatbot construidos y funcionales.

4.4 Sprint 3: Construcción de agentes y módulos auxiliares

En este sprint el trabajo se centró construir los agentes conversacionales y en añadir los módulos que los hacen más completos. Los diseños previos se transformaron en nodos funcionales usando LangGraph, y se sumaron recursos como evaluadores de relevancia, control de alucinaciones, reformulación de preguntas y manejo de errores en SQL. También se crearon tres agentes principales: el RAG, el SQL y el Supervisor, que se reparten las tareas de recuperación, generación de consultas y coordinación de la conversación. Para que todo funcionara, se conectó el LLM con la base de datos universitaria y se definieron las estructuras que permiten esa comunicación. Con estas mejoras, el chatbot comenzó a responder de manera más sólida, incluso frente a preguntas más difíciles o contextos variados.

4.4.1 Reunión de planificación y Sprint Backlog – Sprint 3

Durante la reunión de planificación realizada el 05/05/2025, con la participación del Product Owner, Scrum Master y el Equipo de Desarrollo, se definió el Sprint Backlog correspondiente al Sprint 3, cuyos elementos se detallan en la Tabla 30.

Tabla 30 Sprint Backlog – Sprint 3

Fuente: Autoría propia

Historia de usuario	Nombre	Tarea	Estimación
CTB-01	Chatbot inteligente basado en un LLM.	Desarrollo del módulo evaluador de relevancia en la información recuperada (grader).	1
CTB-01	Chatbot inteligente basado en un LLM.	Desarrollo del módulo evaluador de alucinación en la respuesta del chatbot.	1
CTB-01	Chatbot inteligente basado en un LLM.	Desarrollo del módulo reformulador de preguntas del chatbot.	1
CTB-01	Chatbot inteligente basado en un LLM.	Creación del agente de conversación del chatbot a partir de la conversión de los módulos desarrollados a nodos (langgraph).	1
CTB-04	Acceso del chatbot a la base de datos Universitaria.	Desarrollo del módulo de generación de consultas a partir de lenguaje natural.	5
CTB-04	Acceso del chatbot a la base de datos Universitaria.	Desarrollo del módulo de corrección de consultas sql.	1
CTB-04	Acceso del chatbot a la base de datos Universitaria.	Desarrollo del módulo de control de fallos en consulta a la bdd universitaria.	1

CTB-04	Acceso del chatbot a la base de datos Universitaria.	Creación del agente sql del chatbot a partir de la conversión de los módulos desarrollados a nodos (langgraph).	1
CTB-01	Chatbot inteligente basado en un LLM.	Creación del agente Supervisor para la comunicación entre agentes dentro del chatbot.	3

4.4.2 Reunión de revisión– Sprint 3

Durante la reunión de revisión efectuada el 16/05/2025, con la participación del Product Owner, Scrum Master y el Equipo de Desarrollo, se realizó la evaluación del Sprint Backlog y la verificación de las pruebas de aceptación, con el objetivo de confirmar el cumplimiento de los requisitos establecidos en las historias de usuario definidas (véase Tabla 31).

Tabla 31 Pruebas de aceptación - Sprint 3

Fuente: Autoría propia

Historia de usuario	Nombre	Funcionalidad	Aceptación
CTB-01	Chatbot inteligente basado en un LLM.	Módulo evaluador de relevancia en la información recuperada (grader).	Si
CTB-01	Chatbot inteligente basado en un LLM.	Módulo evaluador de alucinación en la respuesta del chatbot.	Si
CTB-01	Chatbot inteligente basado en un LLM.	Módulo reformulador de preguntas del chatbot.	Si
CTB-01	Chatbot inteligente basado en un LLM.	Agente de conversación del chatbot a partir de la conversión de los módulos desarrollados a nodos (langgraph).	Si

CTB-04	Acceso del chatbot a la base de datos Universitaria.	Módulo de generación de consultas a partir de lenguaje natural.	Si
CTB-04	Acceso del chatbot a la base de datos Universitaria.	Módulo de corrección de consultas sql.	Si
CTB-04	Acceso del chatbot a la base de datos Universitaria.	Módulo de control de fallos en consulta a la bdd universitaria.	Si
CTB-04	Acceso del chatbot a la base de datos Universitaria.	Agente sql del chatbot a partir de la conversión de los módulos desarrollados a nodos (langgraph).	Si
CTB-01	Chatbot inteligente basado en un LLM.	Agente Supervisor para la comunicación entre agentes dentro del chatbot.	Si

El incremento del proyecto que se ha logrado en este sprint se describe a continuación:

- Se desarrolló el módulo encargado de evaluar la relevancia de los documentos recuperados. Este módulo analiza específicamente el fragmento obtenido y utiliza el modelo de lenguaje (LLM) para determinar si dicho contenido está efectivamente relacionado con la consulta del usuario. En caso de que el fragmento recuperado no sea pertinente, se realiza un nuevo proceso de recuperación. Si tras esta segunda búsqueda no se obtienen resultados satisfactorios, se activa el módulo de gestión de fallos, el cual genera un mensaje informando al usuario que no se dispone de información relacionada. La Figura 23 muestra una captura del código principal de este módulo.

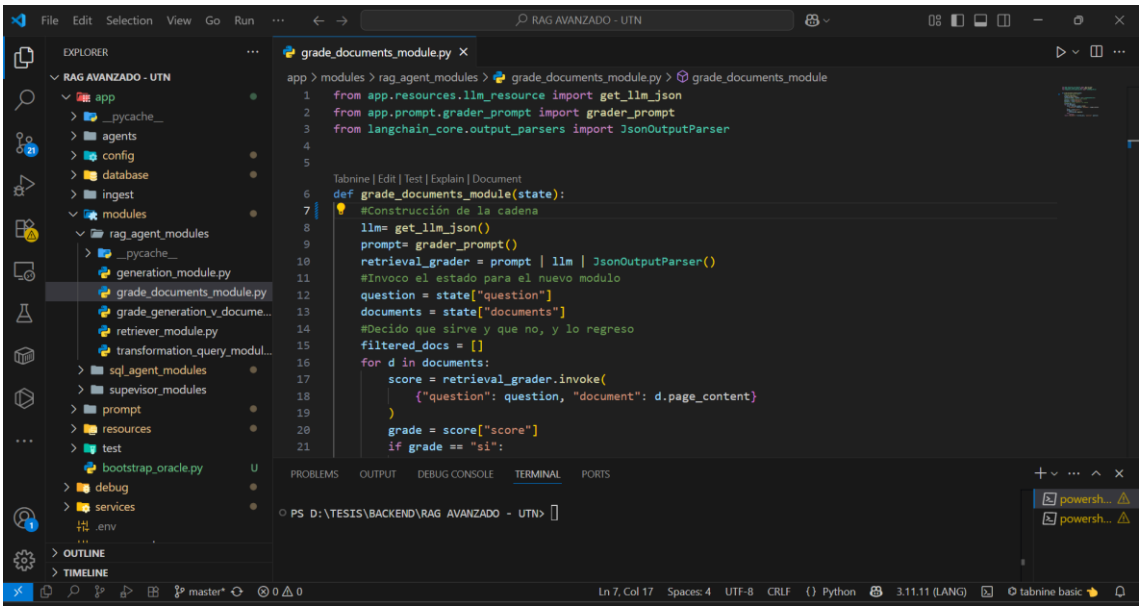


Fig. 23 Código del evaluador de relevancia en información recuperada

Fuente: Autoría propia

- El módulo evaluador de alucinaciones funciona de manera similar al módulo de evaluación de relevancia. Sin embargo, su propósito es comparar la respuesta final generada con los fragmentos documentales recuperados. Si se detecta que la respuesta no guarda relación con la información recuperada, el sistema procede a generar una nueva respuesta. En caso de que esta segunda respuesta continúe siendo insatisfactoria, se activa el módulo de gestión de fallos para informar al usuario que no se dispone de información válida. La Figura 24 muestra una captura del código principal de este módulo.

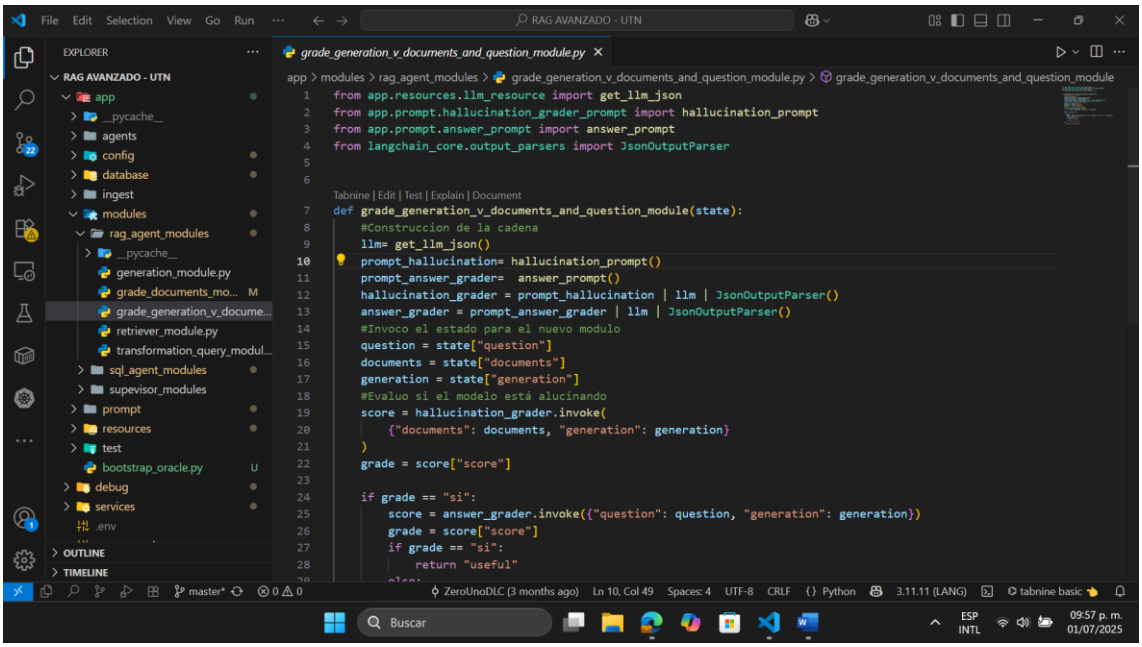


Fig. 24 Código del evaluador de alucinaciones del chatbot

Fuente: Autoría propia

- El módulo de reformulación de preguntas fue implementado con el objetivo de optimizar el proceso de recuperación de información. En muchos casos, el usuario puede formular preguntas poco precisas o adecuadas para el sistema de recuperación; por ello, este módulo se encarga de reinterpretar y reescribir dichas consultas en un formato más claro y efectivo. Por ejemplo, una entrada como "¿Podrías darme info de becas?" se reformula como "Becas disponibles en la Universidad Técnica del Norte". La Figura 25 presenta una captura del código principal de este módulo.

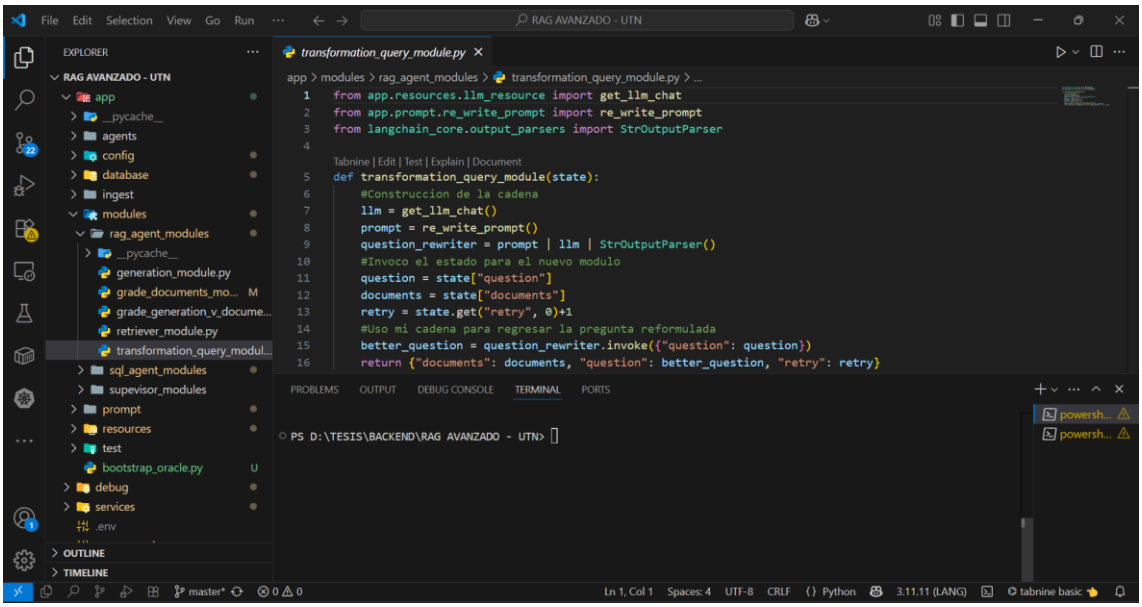


Fig. 25 Código del módulo que reformula preguntas

Fuente: Autoría propia

- El agente RAG se construyó integrando los módulos desarrollados hasta el momento, permitiéndole responder consultas con base en la documentación previamente ingestada. Este agente es capaz de recuperar, evaluar y generar respuestas coherentes y contextualizadas según la información disponible. La Figura 26 muestra el código correspondiente a la construcción de este agente.

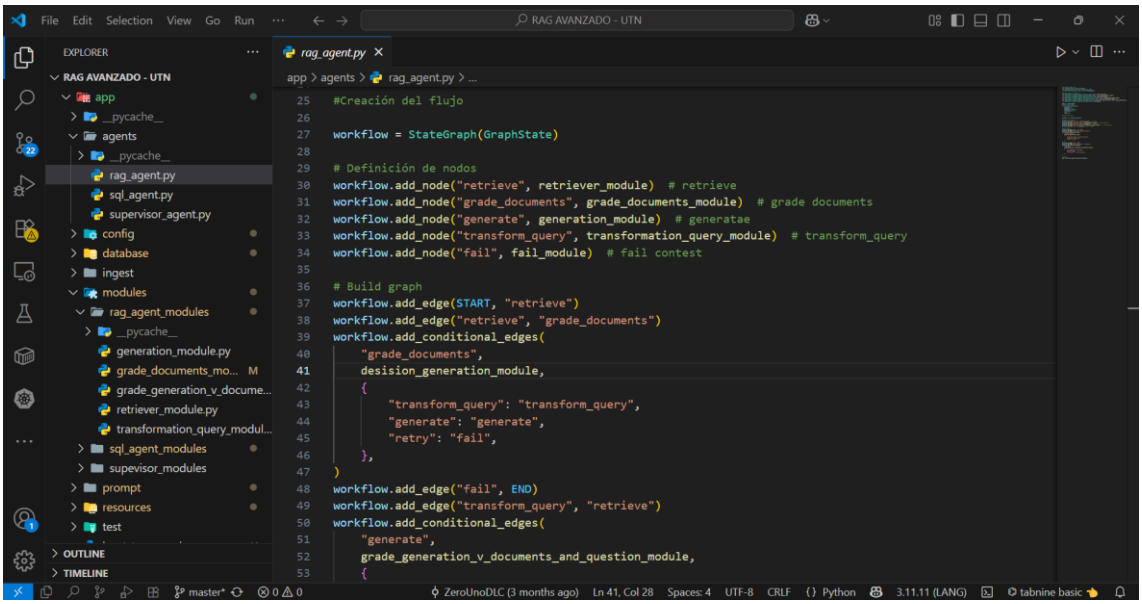


Fig. 26 Construcción del agente RAG

Fuente: Autoría propia

- El módulo de generación de consultas en lenguaje natural se compone de tres nodos principales. En primer lugar, identifica la tabla más adecuada para responder a la consulta del usuario, considerando el contexto proporcionado (Figura 27). A continuación, recupera el esquema de dicha tabla, el cual es incorporado como parte del contexto para el modelo de lenguaje (Figura 28). Finalmente, utilizando tanto la pregunta del usuario como la estructura de la tabla seleccionada, el módulo genera una consulta SQL precisa que será ejecutada sobre la base de datos (Figura 29).

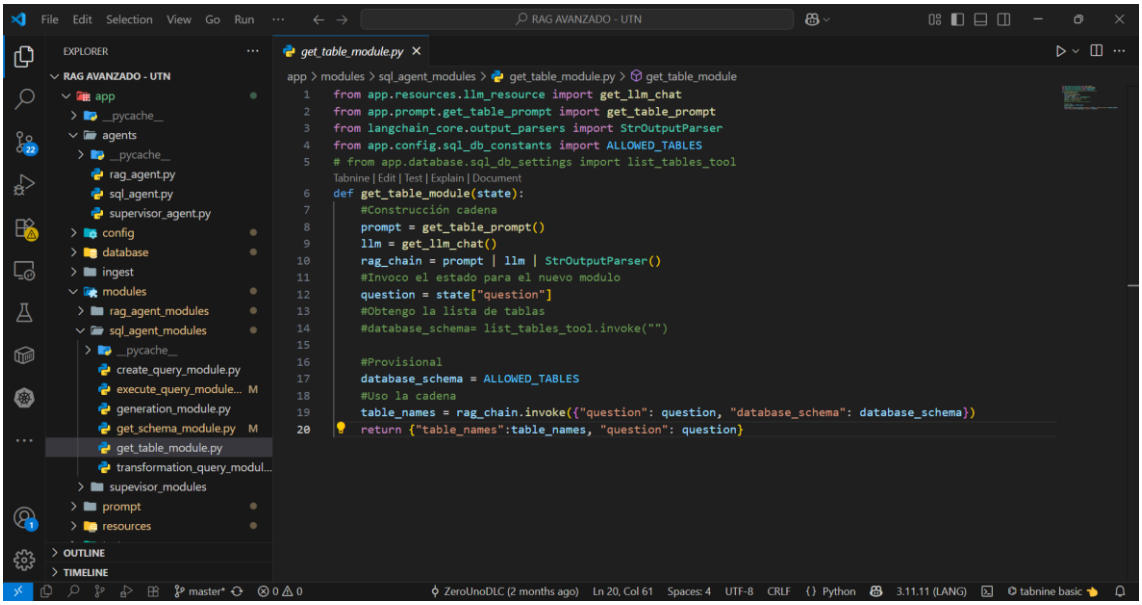


Fig. 27 Código encargado de obtener la tabla relevante a la consulta del usuario

Fuente: Autoría propia

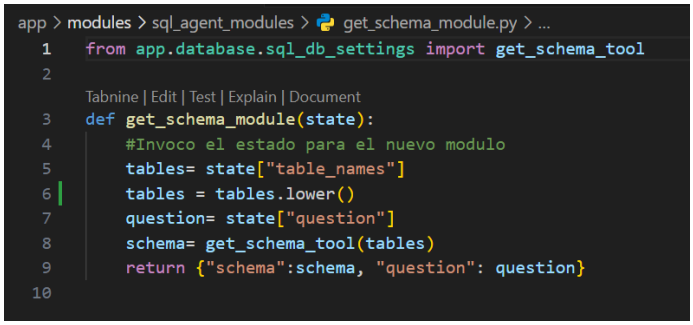


Fig. 28 Código encargado de obtener el esquema de una tabla

Fuente: Autoría propia

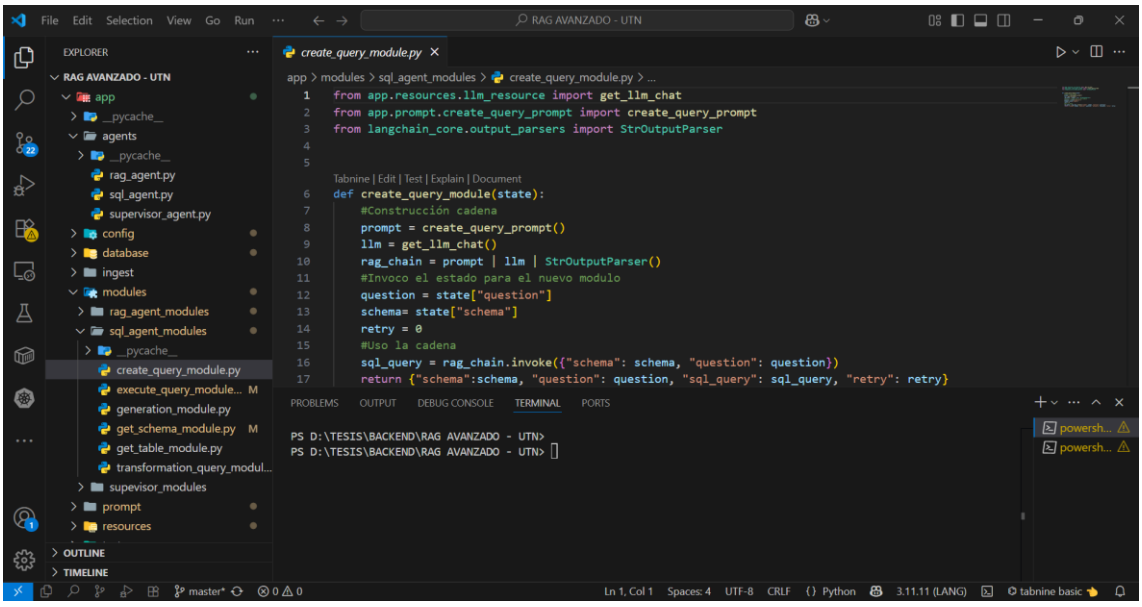


Fig. 29 Código que crea la consulta para la base de datos

Autoría propia

- El módulo corrector de consultas se encarga de tomar la query generada en el nodo anterior y realizar un análisis para identificar y corregir posibles incoherencias sintácticas o errores que puedan impedir su ejecución. Su objetivo es garantizar que la consulta resultante sea válida y ejecutable sobre la base de datos. La Figura 30 muestra una captura del código principal de este módulo.

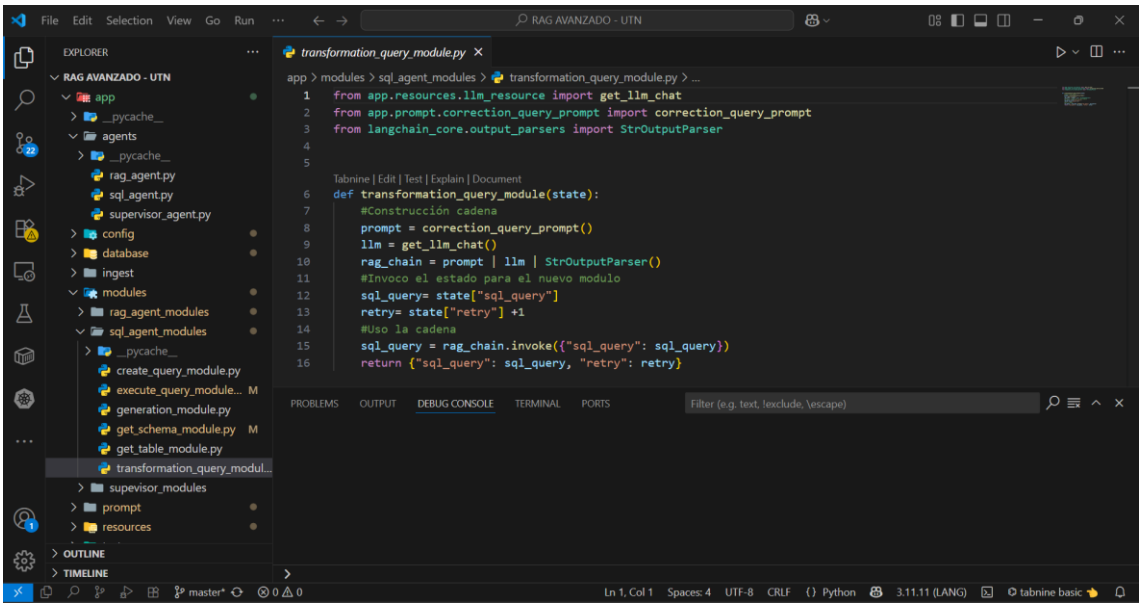


Fig. 30 Código que corrige consultas SQL erróneas

Fuente: Autoría propia

- El módulo de control de fallos fue reutilizado y reestructurado dentro del agente supervisor para integrarse con el módulo de toma de decisiones. En este contexto, si la consulta SQL generada no produce un resultado válido o presenta errores durante su ejecución, el chatbot activa este módulo para notificar al usuario que no dispone de información relacionada con la consulta realizada. La Figura 31 presenta una captura del código principal de este módulo.

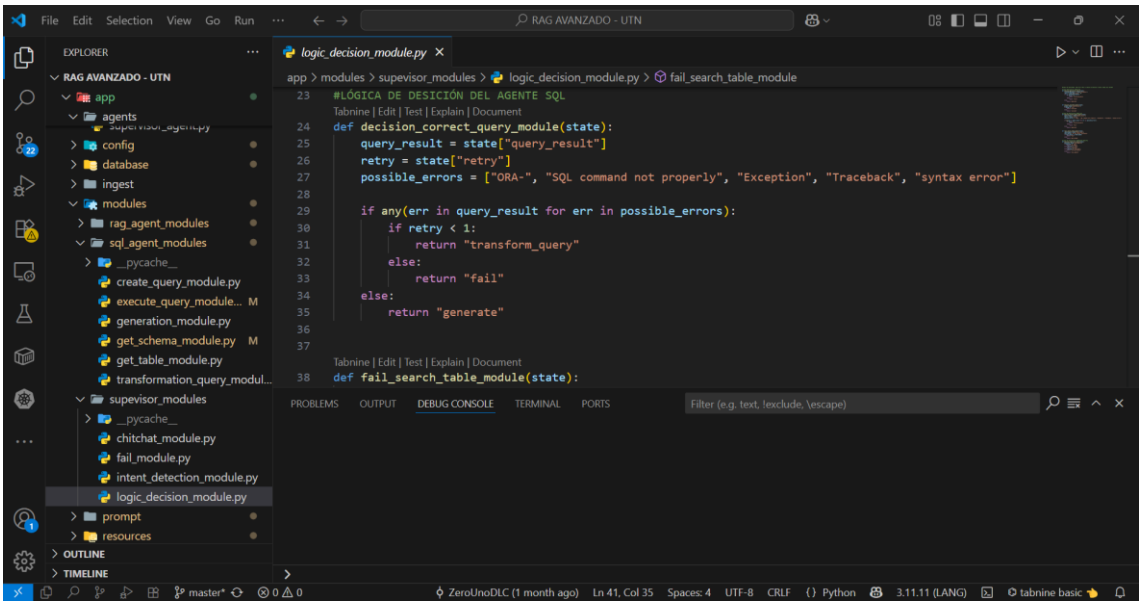


Fig. 31 Código que controla fallos en el agente SQL

Fuente: Autoría propia

- El agente SQL se diseñó a partir de los nodos ya construidos y tiene la capacidad de generar y ejecutar consultas en la base de datos sin intervención externa. A diferencia del RAG, funciona por su cuenta y se centra en responder preguntas que requieren información organizada en tablas. La Figura 32 presenta cómo se implementó este agente dentro del sistema.

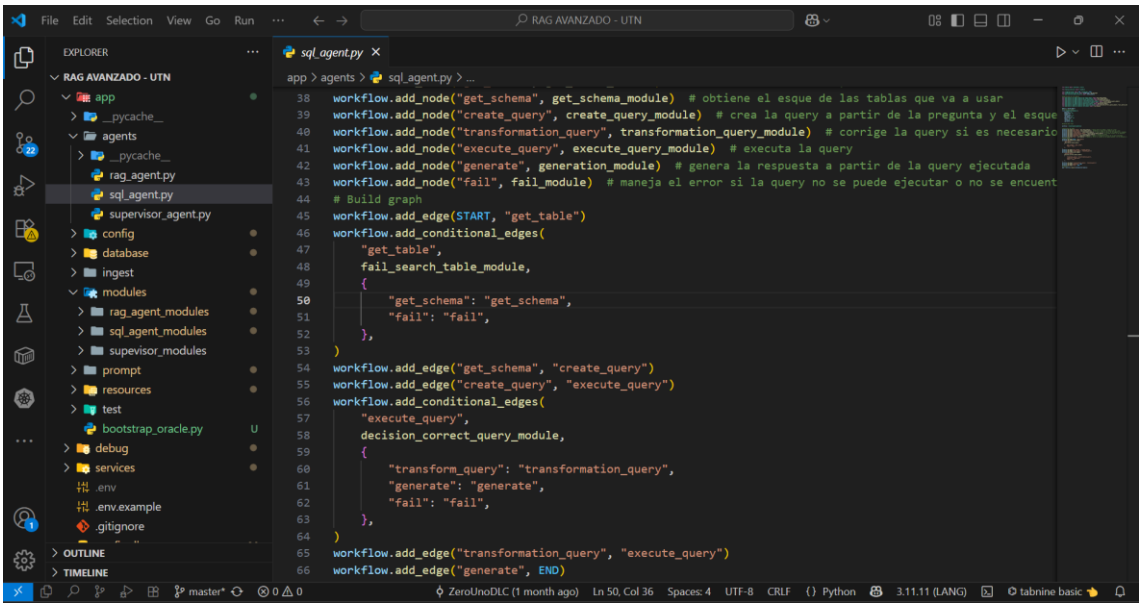


Fig. 32 Construcción del agente SQL

Fuente: Autoría propia

- El agente supervisor fue diseñado como una capa central de coordinación que integra los agentes previamente desarrollados. Su función principal es determinar, en función de la intención de la consulta del usuario, qué agente debe ser activado para generar una respuesta adecuada. Para ello, incorpora como nodos a los agentes RAG y SQL, así como un nodo adicional destinado a la gestión de conversaciones informales.

  El agente supervisor prioriza el uso del agente RAG cuando se trata de preguntas relacionadas con procesos universitarios; recurre al agente SQL en casos donde la consulta involucra procedimientos con referencias temporales o estructuradas; y utiliza el módulo de conversación informal para gestionar saludos u otros mensajes introductorios. Además, está programado para rechazar preguntas que no estén relacionadas con el ámbito universitario o que sean de carácter bromista, con el fin de evitar un uso innecesario del modelo de lenguaje. La Figura 33 muestra la implementación principal de este agente.

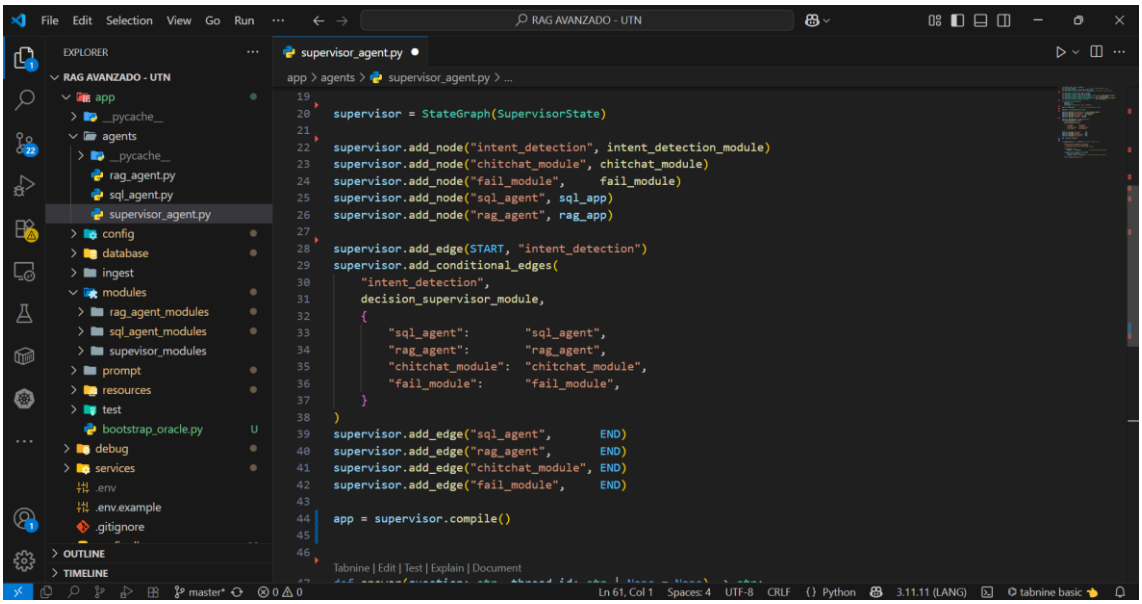


Fig. 33 Construcción del agente Supervisor

Fuente: Autoría propia

4.4.3 Reunión de retrospectiva – Sprint 3

Durante la reunión de retrospectiva realizada el 16/05/2025, con la participación del Product Owner, Scrum Máster y el Equipo de Desarrollo, se obtuvo como resultado el plan de mejora, el cual es evidenciado en la Tabla 32.

Tabla 32 Plan de mejora - Sprint 3

Fuente: Autoría propia

Plan de mejora	
Aciertos	• Creación de los módulos principales de los agentes de inteligencia artificial del chatbot. • Creación de los agentes que componen el chatbot. • Comunicación entre agentes de inteligencia artificial de manera óptima.
Problemas	• Incompatibilidad del uso de todas las herramientas que ofrece LangGraph con Ollama para el uso de LLMs locales. • Mayor latencia de respuesta en modelos de LLMs Saas, como GPT.
Mejoras	• Interfaces gráficas para la administración y consumo del chatbot.

4.5 Sprint 4: Desarrollo de interfaces gráficas para el chatbot

Durante este sprint, los esfuerzos se enfocaron en la construcción de la capa de presentación y de servicios que permite conectar la lógica conversacional del sistema con sus distintos tipos de usuarios. Se implementaron dos APIs REST utilizando FastAPI: una dedicada a la configuración, responsable de exponer de forma segura los parámetros del modelo LLM, las rutas de ingestión y las credenciales de la base de datos; y otra orientada al consumo, optimizada para ofrecer respuestas del chatbot con baja latencia y trazabilidad de fuentes. Además, se diseñó y desarrolló un panel de administración en Oracle APEX, desde el cual los administradores pueden ajustar configuraciones de los modelos, monitorear el proceso de ingestión de conocimiento y consultar métricas de uso. También se creó el aplicativo móvil en Flutter para el consumo del chatbot por parte de los estudiantes.

4.5.1 Reunión de planificación y Sprint Backlog – Sprint 4

Durante la reunión de planificación realizada el 19/05/2025, con la participación del Product Owner, Scrum Máster y el Equipo de Desarrollo, se definió el Sprint Backlog correspondiente al Sprint 4, cuyos elementos se detallan en la Tabla 33.

Tabla 33 Sprint Backlog – Sprint 4

Fuente: Autoría propia

Historia de usuario	Nombre	Tarea	Estimación
CTB-02	Sistema de configuración para el chatbot.	Desarrollo del API de administración del chatbot.	5
CTB-02	Sistema de configuración para el chatbot.	Creación del diseño del frontal de administración del chatbot	3
CTB-02	Sistema de configuración para el chatbot.	Desarrollo del frontal de administración del chatbot	13
CTB-05	Administración dinámica de la base de datos utilizada por el chatbot.	Creación del endpoint de configuración dónde el chatbot hace sus consultas.	2

		Reestructuración del proyecto.	
CTB-06	Administración del acceso del chatbot a las tablas de la base de datos.	Creación del endpoint para escribir sub-esquemas a los que el chatbot tendrá acceso.	3
CTB-07	Configurar modelo LLM.	Creación del endpoint para configurar dinámicamente el LLM a utilizarse.	3
CTB-08	Aplicativo móvil para el chatbot.	Desarrollo del API para consumo del chatbot	2
CTB-08	Aplicativo móvil para el chatbot.	Creación del diseño del frontal para el chatbot con el usuario final (estudiante)	3
CTB-08	Aplicativo móvil para el chatbot.	Desarrollo del frontal para el chatbot para el usuario final (estudiante)	8
CTB-09	Exportar el historial del chat a PDF.	Desarrollo de la funcionalidad de exportar la conversación en pantalla dentro del aplicativo del chatbot.	3
CTB-10	Borrar conversación en pantalla.	Desarrollo de la funcionalidad de limpiar la pantalla de chat del aplicativo del chatbot.	2
CTB-11	Control de redundancia en almacenamiento de mensajes.	Desarrollo de la funcionalidad de mantener las ultimas N interacciones en pantalla en la aplicación (sin base de datos)	3

4.5.2 Reunión de revisión Sprint 4

Durante la reunión de revisión realizada el 30/05/2025, con la participación del Product Owner, Scrum Máster y el Equipo de Desarrollo, se llevó a cabo la evaluación del Sprint Backlog y la validación de las pruebas de aceptación, con el fin de comprobar

el cumplimiento de los requisitos asociados a las historias de usuario definidas (Véase en la Tabla 34).

Tabla 34 Pruebas de aceptación – Sprint 4

Fuente: Autoría propia

Historia de usuario	Nombre	Funcionalidad	Aceptación
CTB-02	Sistema de configuración para el chatbot.	API para consumo del chatbot.	Si
CTB-02	Sistema de configuración para el chatbot.	Diseño del frontal para el chatbot con el usuario final (estudiante).	Si
CTB-02	Sistema de configuración para el chatbot.	Frontal para el chatbot para el usuario final (estudiante).	Si
CTB-05	Administración dinámica de la base de datos utilizada por el chatbot.	Funcionalidad de configuración dónde el chatbot hace sus consultas.	Si
CTB-06	Administración del acceso del chatbot a las tablas de la base de datos.	Funcionalidad para escribir sub-esquemas a los que el chatbot tendrá acceso.	Si
CTB-07	Configurar modelo LLM.	Funcionalidad para configurar dinámicamente el LLM a utilizarse.	Si
CTB-08	Aplicativo móvil para el chatbot.	API de administración del chatbot.	Si
CTB-08	Aplicativo móvil para el chatbot.	Diseño del frontal de administración del chatbot.	Si
CTB-08	Aplicativo móvil para el chatbot.	Frontal de administración del chatbot.	Si

CTB-09	Exportar el historial del chat a PDF.	Funcionalidad de exportar la conversación en pantalla dentro del aplicativo del chatbot.	Si
CTB-10	Borrar conversación en pantalla.	Funcionalidad de limpiar la pantalla de chat del aplicativo del chatbot.	Si
CTB-11	Control de redundancia en almacenamiento de mensajes.	Funcionalidad de mantener las ultimas N interacciones en pantalla en la aplicación (sin base de datos)	Si

El incremento que se realizó en el proyecto relacionado con todas las tareas realizadas en este Sprint se describe a continuación:

- Para la gestión centralizada de los parámetros del chatbot, se implementó una API con FastAPI para la configuración de parámetros del chatbot, esto aprovechando su compatibilidad con el ecosistema Python; en la Figura 34 se muestra el Swagger de esta.

Fig. 34 Swagger del API de configuración del chatbot

Fuente: Autoría propia

- Con el fin de facilitar la administración del sistema del chatbot, se diseñó la interfaz web del chatbot a desarrollarse en Oracle Apex, visible en la Figura 35.

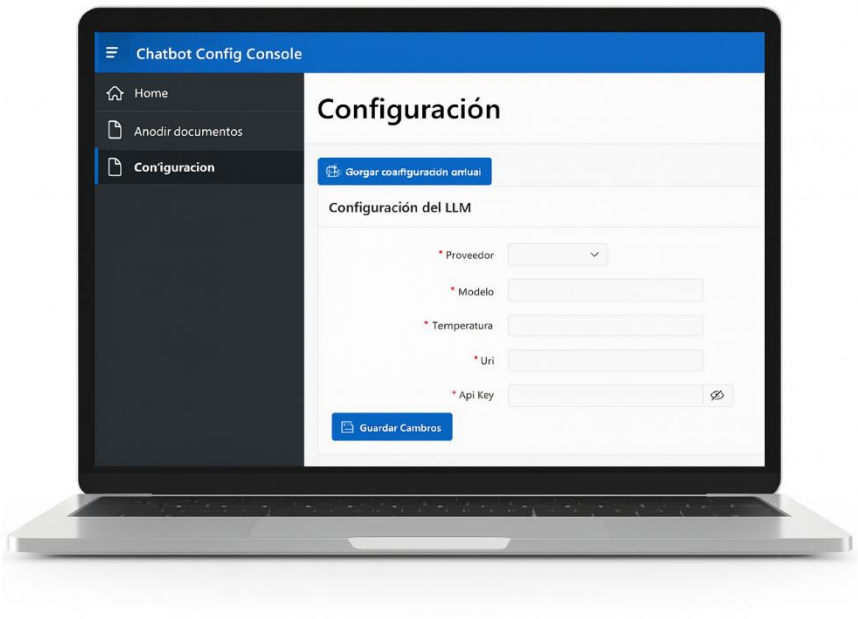


Fig. 35 Diseño de la interfaz de administración del chatbot

Fuente: Autoría propia

- Dentro del entorno universitario, se habilitó un panel de control en Oracle APEX para la administración del chatbot; la Figura 36 ilustra su disposición.

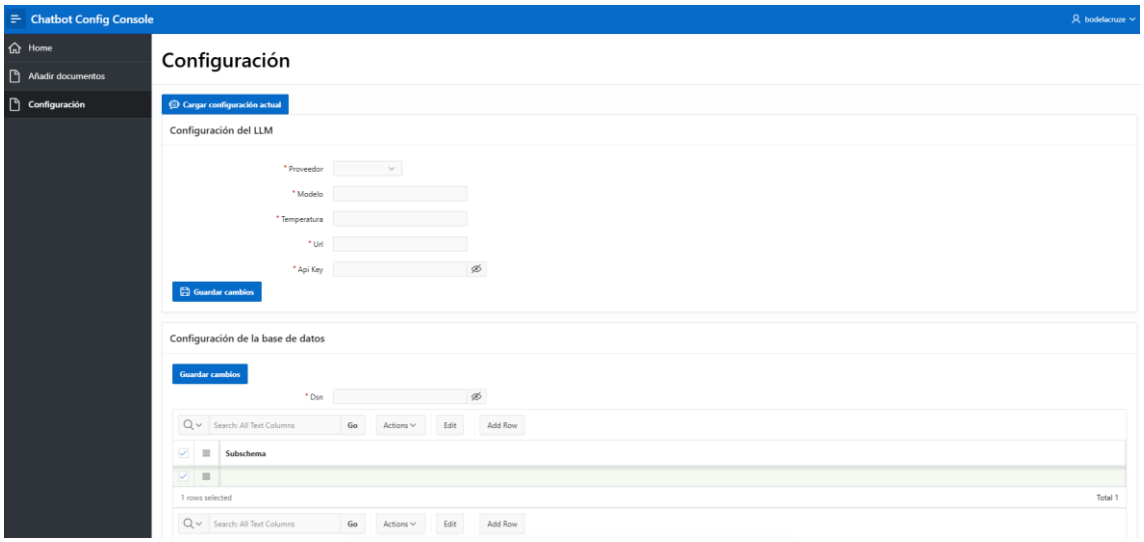


Fig. 36 Interfaz de administración del chatbot en Oracle APEX

Fuente: Autoría propia

- Para permitir la configuración dinámica de la base de datos SQL que consulta el chatbot, se implementó la lógica en el código fuente como se muestra en la figura 37, y se añadió dicha funcionalidad en el panel de administración como se muestra en la figura 38.

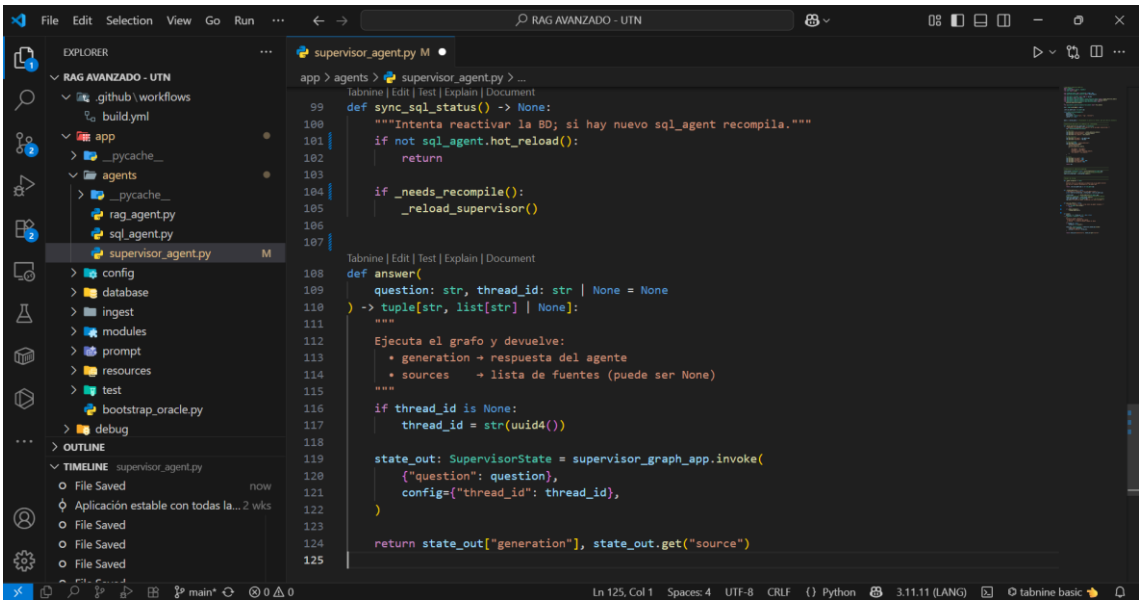


Fig. 37 Implementación de la lógica para cambio dinámico de la base de datos del chatbot

Fuente: Autoría propia

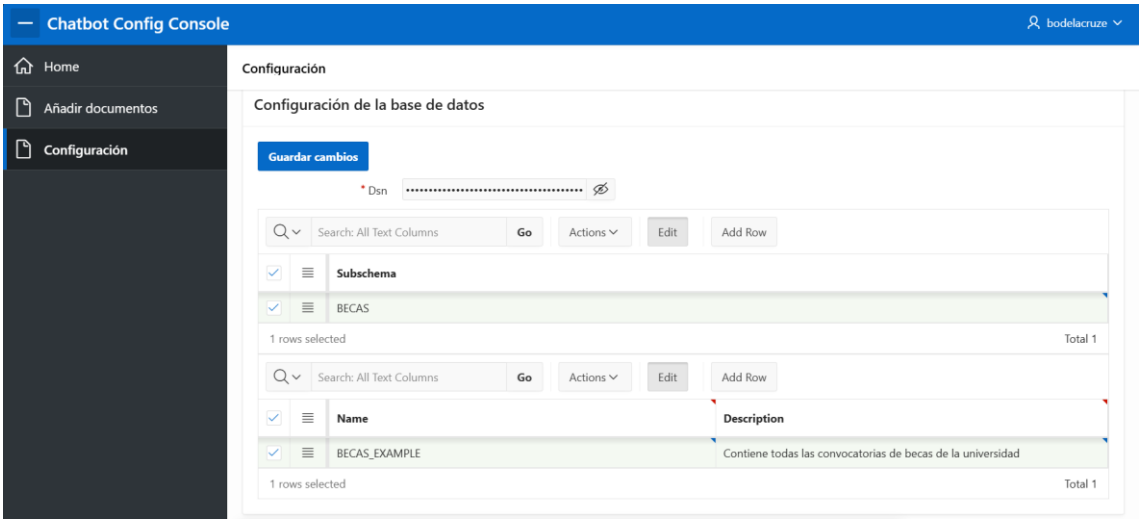


Fig. 38 Implementación del cambio dinámico de la base de datos en la interfaz de administración

Fuente: Autoría propia

- Con el propósito de delimitar el conocimiento accesible al modelo, se incorporó la opción de definir subesquemas y describir tablas específicas dentro de este; la Figura 39 detalla esta funcionalidad.

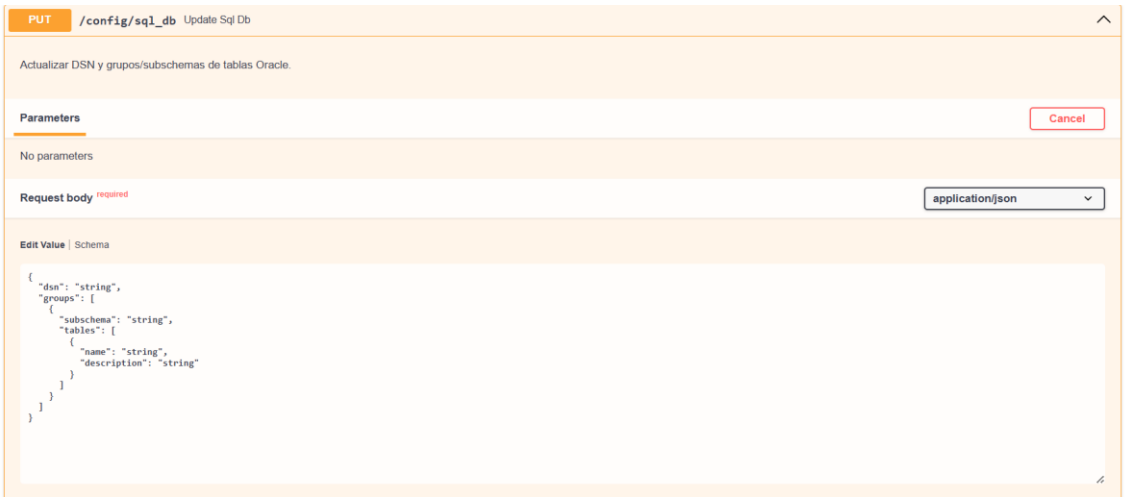


Fig. 39 Estructura de configuración de subesquemas en el sistema del chatbot

Fuente: Autoría propia

- A fin de cambiar en caliente el modelo de lenguaje usado por el chatbot, se implementó la selección dinámica de LLM, se puede omitir el campo de "api_key" si se opta por utilizar un LLM local, el uso de esta funcionalidad es mostrada en la Figura 40.

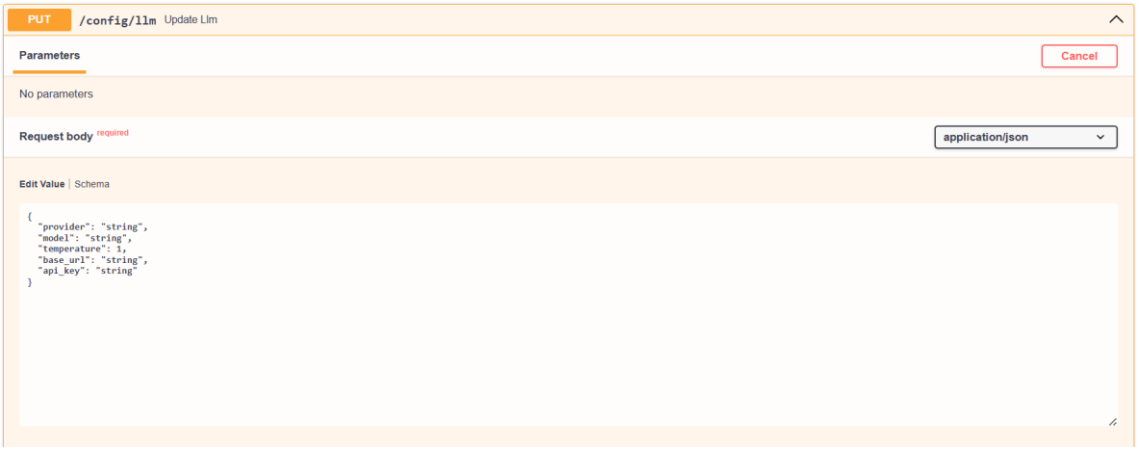


Fig. 40 Estructura de configuración de LLM en el sistema del chatbot

Fuente: Autoría propia

- Para exponer el agente supervisor a los clientes externos, se desarrolló una segunda API de consumo en FastAPI; su estructura se refleja en la Figura 41.

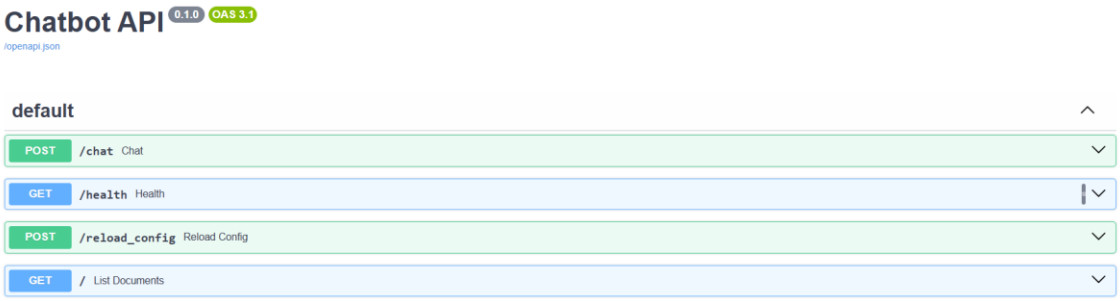


Fig. 41 Swagger del API de consumo del chatbot

Fuente: Autoría propia

- Para el frontal móvil, se diseñó la experiencia de usuario respetando la paleta cromática del macroproyecto UTN Móvil; la Figura 42 presenta dicha interfaz.

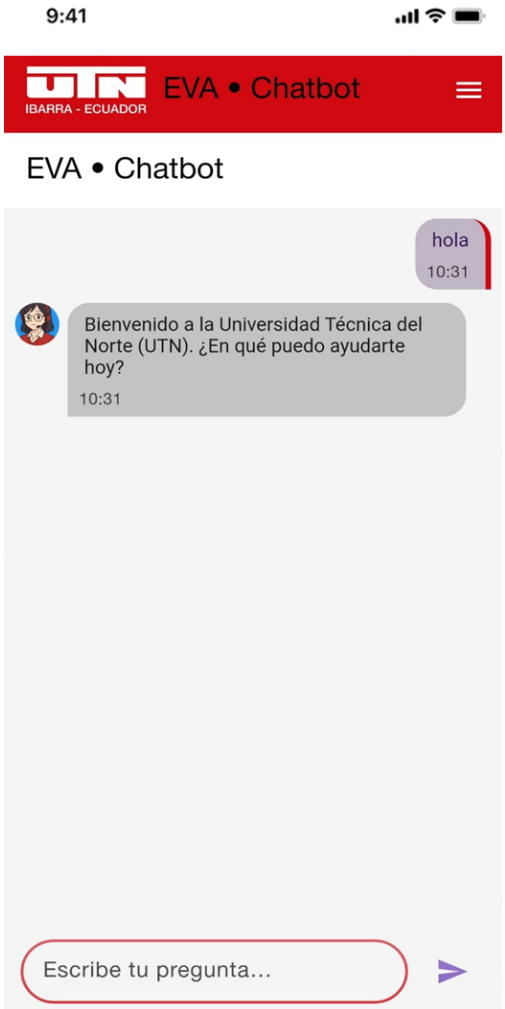


Fig. 42 Diseño de la interfaz móvil para el consumo del chatbot

Fuente: Autoría propia

- Con base en la arquitectura exigida por el macroproyecto, se integró el front-end móvil que invoca al chatbot, como se observa en la Figura 43, y se desarrolló el frontal de acorde al diseño como se muestra en la figura 44.

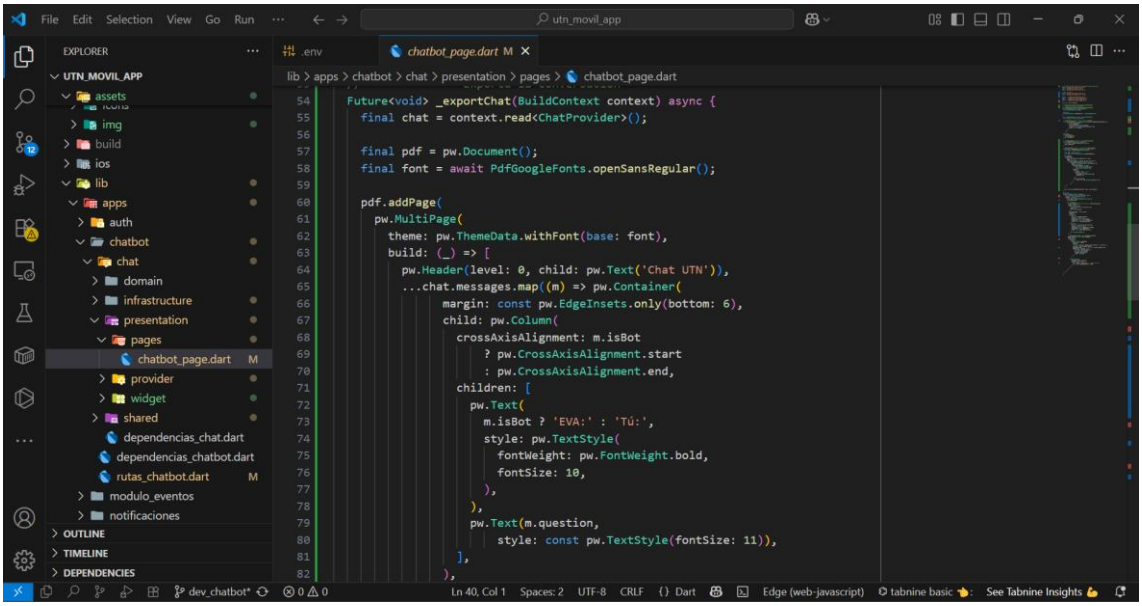


Fig. 43 Desarrollo del frontal móvil en la arquitectura del macroproyecto UTN móvil

Fuente: Autoría propia

Fig. 44 Frontal móvil del chatbot

Fuente: Autoría propia

- Para permitir la exportación de la conversación visible, se añadió la funcionalidad de generar un archivo PDF de acorde a la conversación en pantalla, la implementación técnica se muestra en la figura 45, y la evidencia práctica se muestra en la figura 46.

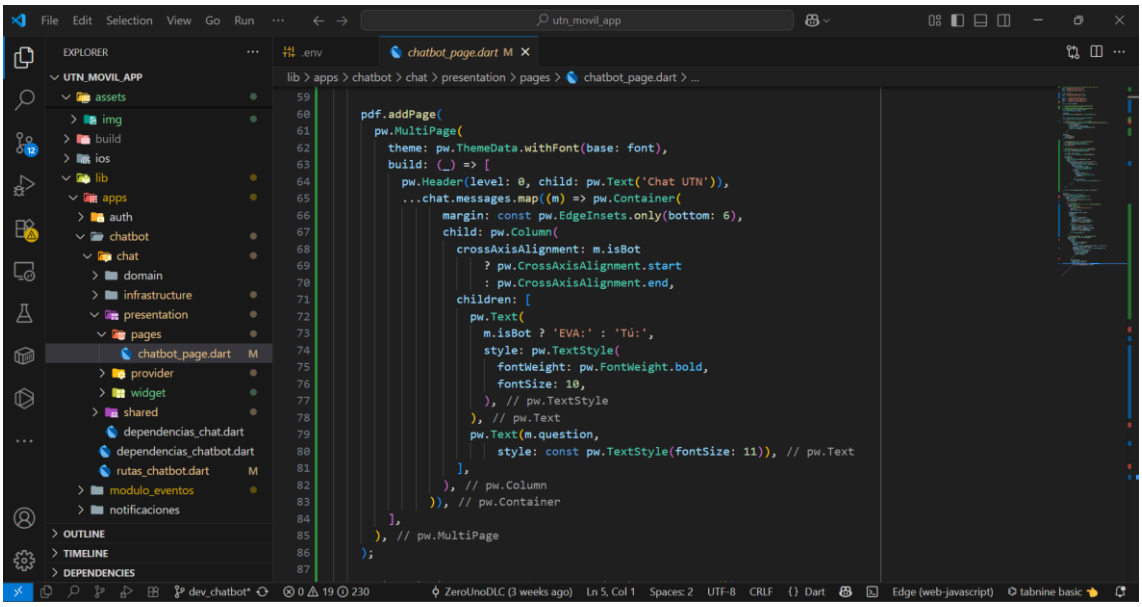


Fig. 45 Implementación técnica de la funcionalidad de exportación de conversación de pantalla a PDF

Fuente: Autoría propia

Fig. 46 PDF de conversación en pantalla exportada del chatbot

Fuente: Autoría propia

- Con el objetivo de mejorar la usabilidad, se implementó la opción de limpiar el historial mostrado en el front-end de consumo, tal como se indica en la Figura 47.

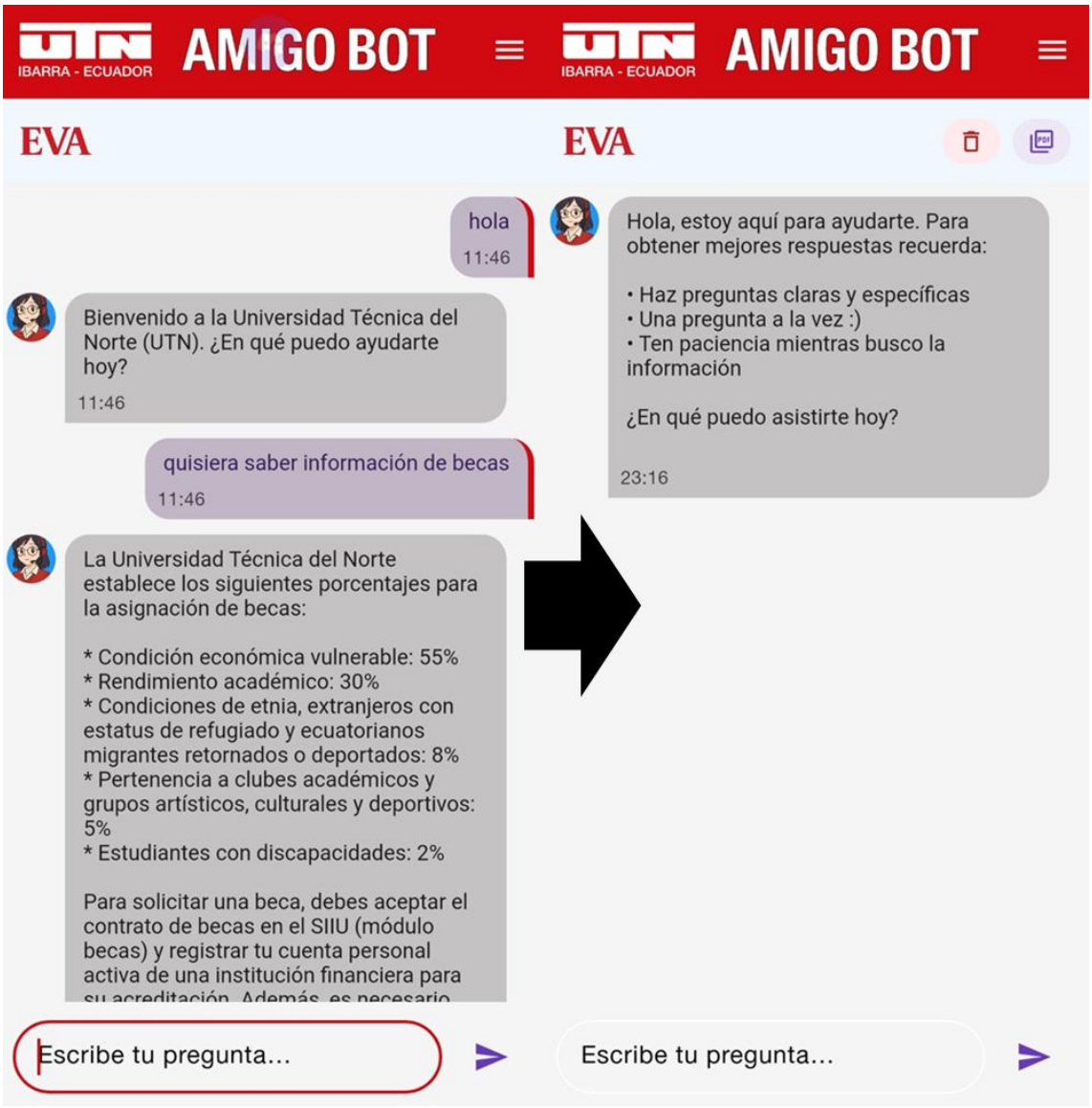


Fig. 47 Funcionalidad de limpieza de pantalla del frontal móvil del chatbot

Fuente: Autoría propia

- Para la persistencia local de sesiones, se almacenaron las últimas N interacciones directamente en la aplicación sin emplear una base de datos externa; la Figura 48 muestra esta implementación.

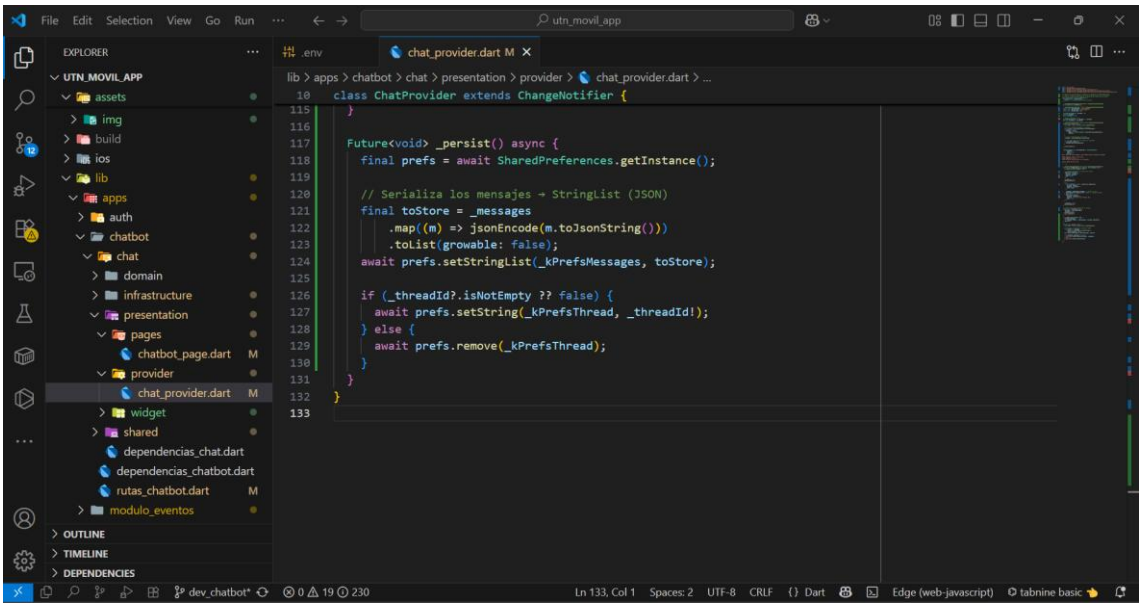


Fig. 48 Implementación de persistencia local de mensajes en el frontal móvil del chatbot

Fuente: Autoría propia

- Se optó por crear una pantalla en la cual se indique la información que se encuentra dentro del chatbot junto a un aviso importante sobre respuestas generadas por IA, esto para delegar responsabilidad al usuario en cuanto al uso responsable de esta, esta pantalla se muestra en la Figura 49.

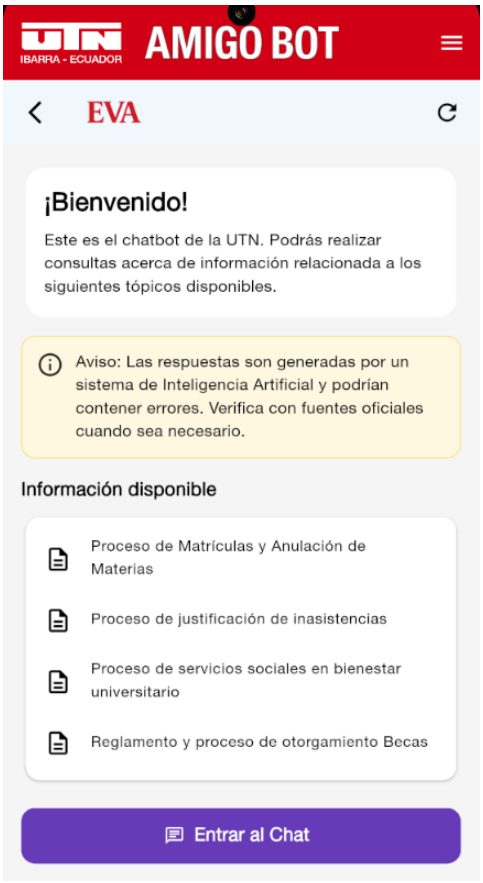


Fig. 49 Pantalla que muestra la información que contiene el chatbot

Fuente: Autoría propia

4.5.3 Reunión de retrospectiva – Sprint 4

Durante la reunión de retrospectiva realizada el 30/05/2025, con la participación del Product Owner, Scrum Máster y el Equipo de Desarrollo, se obtuvo como resultado el plan de mejora, el cual es evidenciado en la Tabla 35.

Tabla 35 Plan de mejora – Sprint 4

Fuente: Autoría propia

Plan de mejora	
Aciertos	• Creación de la aplicación para el consumo para el chatbot. • Creación del panel en APEX para la administración del chatbot. • Diseños adaptados a las especificaciones de la universidad.
Problemas	• Dificultad del consumo del API de administración desde APEX. • Conocimiento limitado de Oracle APEX.

Mejoras	• Base de conocimiento del chatbot alimentada con la información de los procesos universitarios relacionados al portafolio estudiantil.

4.6 Sprint 5: Ingesta de documentación a la base de conocimiento y despliegue del chatbot

En este Sprint 5 se consolidó la base de conocimiento y se llevó el chatbot a producción. El trabajo comenzó con la recopilación de información y su clasificación por temas, de manera que pudiera usarse en búsquedas contextuales. Luego, con la metodología CRISP-DM, los datos se limpiaron y se les dio el formato necesario para integrarlos en ChromaDB. Después se probaron distintos modelos y se eligieron los LLM que mejor se adaptaban al chatbot. Una vez definidos, se hicieron pruebas de los diálogos y de la trazabilidad de las fuentes. El proyecto concluyó con el despliegue del chatbot en el entorno universitario y con la elaboración de un manual práctico, pensado para orientar a los usuarios y facilitar el entendimiento al departamento de tecnologías de la universidad.

4.6.1 Reunión de planificación y Sprint Backlog – Sprint 5

Durante la reunión de planificación realizada el 02/06/2025, con la participación del Product Owner, Scrum Máster y el Equipo de Desarrollo, se definió el Sprint Backlog correspondiente al Sprint 5, cuyos elementos se detallan en la Tabla 36.

Tabla 36 Sprint backlog – Sprint 5

Fuente: Autoría propia

Historia de usuario	Nombre	Tarea	Estimación
CTB-13	Ingesta de documentación al chatbot sobre procesos del portafolio estudiantil.	Selección de la información a ser ingestada en el chatbot.	1
CTB-13	Ingesta de documentación al chatbot sobre procesos	Recopilación de información a ser indexada a la base de conocimiento del chatbot.	13

	del portafolio estudiantil.		
CTB-13	Ingesta de documentación al chatbot sobre procesos del portafolio estudiantil.	Formalización y clasificación de la información según su contenido.	1
CTB-13	Ingesta de documentación al chatbot sobre procesos del portafolio estudiantil.	Aplicación de la metodología CRISP - DM para la limpieza y formato de la información.	5
CTB-13	Ingesta de documentación al chatbot sobre procesos del portafolio estudiantil.	Ingesta de la documentación en el chatbot y prueba de funcionalidades.	1
CTB-12	Aplicativo móvil para el chatbot.	Elección del LLM a utilizarse en el chatbot.	2
CTB-12	Despliegue del chatbot.	Dockerización de la aplicación.	3
CTB-12	Despliegue del chatbot.	Despliegue del chatbot.	8
CTB-14	Creación del manual de uso del chatbot.	Creación del manual de uso del chatbot	3

4.6.2 Reunión de revisión Sprint 5

Durante la reunión de revisión realizada el 13/06/2025, con la participación del Product Owner, Scrum Máster y el Equipo de Desarrollo, se llevó a cabo la evaluación del Sprint Backlog y la validación de las pruebas de aceptación, con el fin de comprobar el cumplimiento de los requisitos asociados a las historias de usuario definidas (Véase en la Tabla 37).

Tabla 37 Pruebas de aceptación - Sprint 5

Fuente: Autoría propia

Historia de usuario	Nombre	Funcionalidad	Aceptación
CTB-13	Ingesta de documentación al chatbot sobre procesos del portafolio estudiantil.	Información a ser ingestada en el chatbot seleccionada.	Si
CTB-13	Ingesta de documentación al chatbot sobre procesos del portafolio estudiantil.	Información para indexar a la base de conocimiento del chatbot recolectada.	Si
CTB-13	Ingesta de documentación al chatbot sobre procesos del portafolio estudiantil.	Información formalizada y clasificada según su contenido.	Si
CTB-13	Ingesta de documentación al chatbot sobre procesos del portafolio estudiantil.	Información formateada con ayuda de la metodología CRISP-DM.	Si
CTB-13	Ingesta de documentación al chatbot sobre procesos del portafolio estudiantil.	Base de conocimiento del chatbot con la información de los procesos del portafolio estudiantil.	Si
CTB-12	Aplicativo móvil para el chatbot.	LLM seleccionado para la utilización del chatbot.	Si
CTB-12	Despliegue del chatbot.	Backend del chatbot Dockerizado.	Si
CTB-12	Despliegue del chatbot.	Chatbot desplegado.	Si

CTB-14	Creación del manual de uso del chatbot.	Manual de utilización del chatbot.	Si

El incremento funcional obtenido en el proyecto durante este sprint, derivado de las tareas ejecutadas conforme a las historias de usuario, se describe a continuación:

- Con el apoyo de los Product Owners se identificó y seleccionó la información que conforma la base de conocimientos inicial del chatbot. La Tabla 38 detalla los tipos de contenido incorporados, junto con el documento institucional de referencia que describe los procesos universitarios correspondientes.

Tabla 38 Información seleccionada para la base inicial de conocimientos del chatbot

Fuente: Autoría propia

Proceso	**Documento**
Proceso de atención del centro de educación inicial	Proceso de atención – CEI [61]
Proceso de atención medica – enfermería	Flujograma de Atenciones Médicas y de Enfermería [62]
Proceso de atención medica - laboratorio clínico	Proceso de atención medica - laboratorio clínico [63]
Proceso de atención medica - odontología	Proceso de atención - Odontológica [64]
Proceso de defensa de Tesis y prórrogas	Reglamento de la Unidad de Integración curricular de grado de la UTN [65]
Proceso de distribución de actividades del personal académico	Instructivo para la distribución de actividades de enseñanza-docencia, investigación, vinculación con la sociedad y gestión educativa para el personal académico titular y ocasional de la UTN [66]
Proceso de justificación de inasistencias	Reglamento de Régimen Académico de Grado y Posgrado [67]
Proceso de Matrículas y Anulación de Materias	Reglamento de Régimen Académico de Grado y Posgrado [67]

Proceso de prácticas preprofesionales	Lineamiento transitorio para el cumplimiento del requisito de prácticas preprofesionales y vinculación con la sociedad [68]
Proceso de servicios sociales en bienestar universitario	Proceso de atención – Psicológica [69] Proceso de atención - Orientación Profesional [70] Proceso de atención Trabajo Social [71]
Proceso del curso de actualización de conocimientos (CAC)	Reglamento para curso de actualización de conocimientos en carreras de grado [72]
Reglamento de trabajos de grado (Tesis)	Reglamento de la Unidad de Integración curricular de grado de la UTN [65]
Reglamento y proceso de otorgamiento Becas	Reglamento de Becas Reformado para Estudiantes de Carreras de Grado [73]

- Posterior a la selección de la información descrita anteriormente, se procedió a su recolección y formalización, clasificando dicha información según su contenido en un documento con extensión .docx.
- Se aplicó un formato a la información siguiendo la metodología CRISP-DM, en la fase de preparación de datos. Los pasos ejecutados fueron:
  - Corrección ortográfica y gramatical.
  - Normalización de títulos, subtítulos y secciones.
  - Eliminación de redundancias, saltos innecesarios o símbolos sueltos.
  - Formateo en bloques de información para que el chatbot pueda dividirlos de manera más eficiente.
  - Inclusión de términos clave en cada párrafo para facilitar el ingreso ordenado en la base de conocimiento.
  - Conversión a PDF de cada texto después de su limpieza.
- La información formateada fue indexada a la base de conocimientos del chatbot, siendo capaz este de responder a preguntas basadas en dicha información.

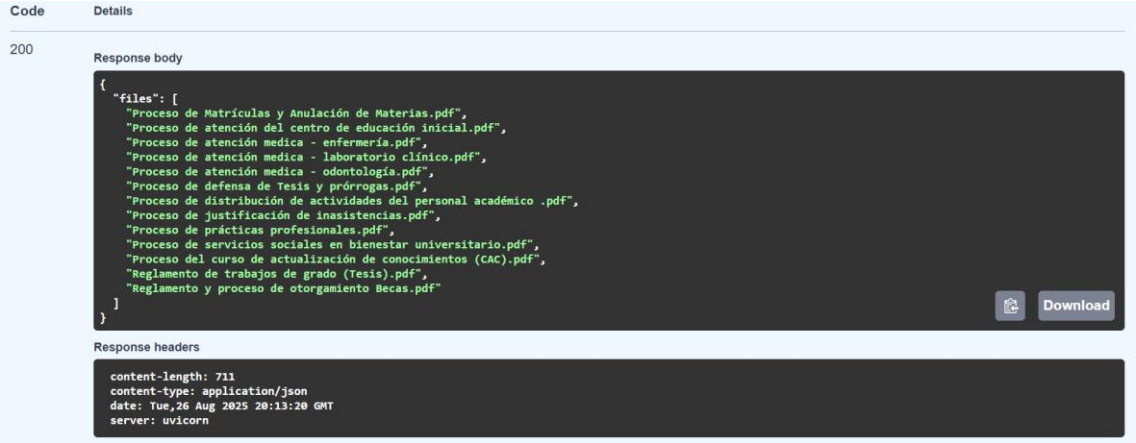


Fig. 50 Ingesta de documentación a la base de conocimiento del chatbot

Fuente: Autoría propia

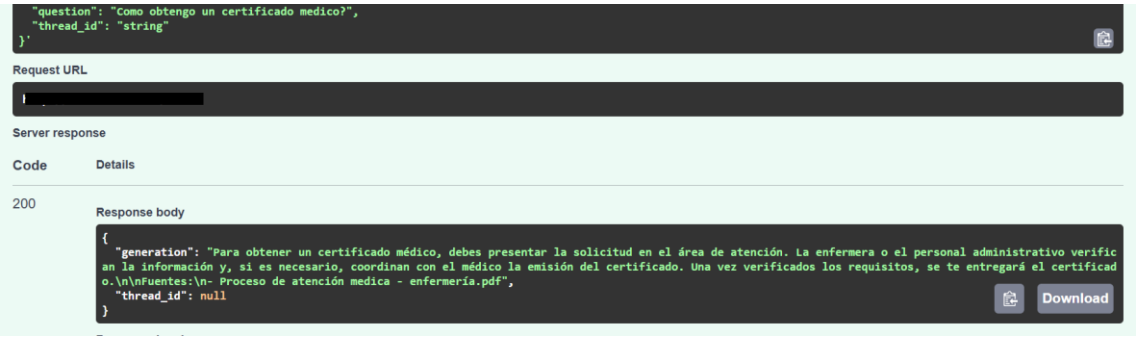


Fig. 51 Prueba de funcionamiento del chatbot posterior a la ingesta

Fuente: Autoría propia

- Se optó por utilizar la API de GPT, específicamente el modelo gpt-4.1-nano, debido a que representa una alternativa de bajo costo y alta velocidad. En términos presupuestarios, resulta más eficiente y rápido de implementar, ofreciendo además un razonamiento y capacidad de procesamiento superiores en comparación con el mantenimiento de un modelo desplegado de manera local. Los beneficios de utilizar el API en conjunto con este modelo se describen en la Tabla 39.

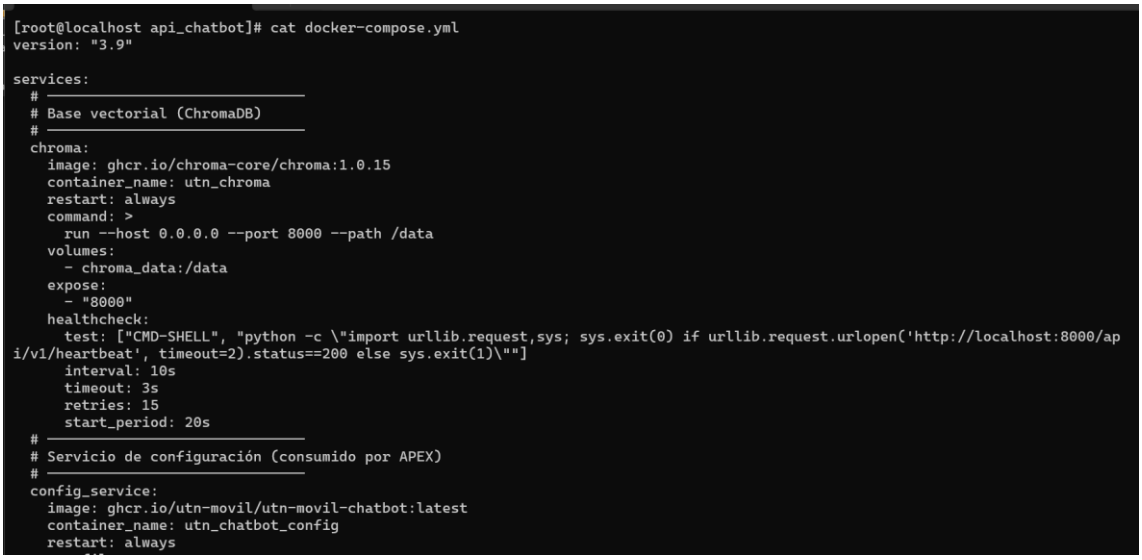
Tabla 39 Beneficios del modelo gpt-4.1-nano de OpenAI

Fuente: Autoría propia

Beneficio	Descripción breve
Bajo costo	Es el modelo más económico de la familia GPT-4.1, con tarifas muy reducidas por token.

Alta velocidad / baja latencia	Ideal para casos en tiempo real, como autocompletar y clasificación rápida.
Contexto extenso	Admite hasta 1 millón de tokens, facilitando el manejo de documentos muy largos sin recortes.
Procesamiento eficiente	Equilibra velocidad, precisión y eficiencia, útil para tareas de extracción y análisis rápido.
Fácil implementación presupuestaria	Más económico y rápido de implementar que mantener un modelo localmente.

- Se dockerizo el chatbot para continuar con el proceso de despliegue dentro de un servidor dedicado proveído por la universidad.



```
[root@localhost api_chatbot]# cat docker-compose.yml
version: "3.9"

services:
  # ──────────────────────────────
  # Base vectorial (ChromaDB)
  # ──────────────────────────────
  chroma:
    image: ghcr.io/chroma-core/chroma:1.0.15
    container_name: utn_chroma
    restart: always
    command: >
      run --host 0.0.0.0 --port 8000 --path /data
    volumes:
      - chroma_data:/data
    expose:
      - "8000"
    healthcheck:
      test: ["CMD-SHELL", "python -c \"import urllib.request,sys; sys.exit(0) if urllib.request.urlopen('http://localhost:8000/ap
i/v1/heartbeat', timeout=2).status==200 else sys.exit(1)\""]
      interval: 10s
      timeout: 3s
      retries: 15
      start_period: 20s
  # ──────────────────────────────
  # Servicio de configuración (consumido por APEX)
  # ──────────────────────────────
  config_service:
    image: ghcr.io/utn-movil/utn-movil-chatbot:latest
    container_name: utn_chatbot_config
    restart: always
```

Fig. 52 Docker - Compose del chatbot dentro del servidor universitario

Fuente: Autoría propia

- El chatbot se desplegó en el ambiente universitario, siendo capaz de ser consumido sin problema dentro de este entorno.

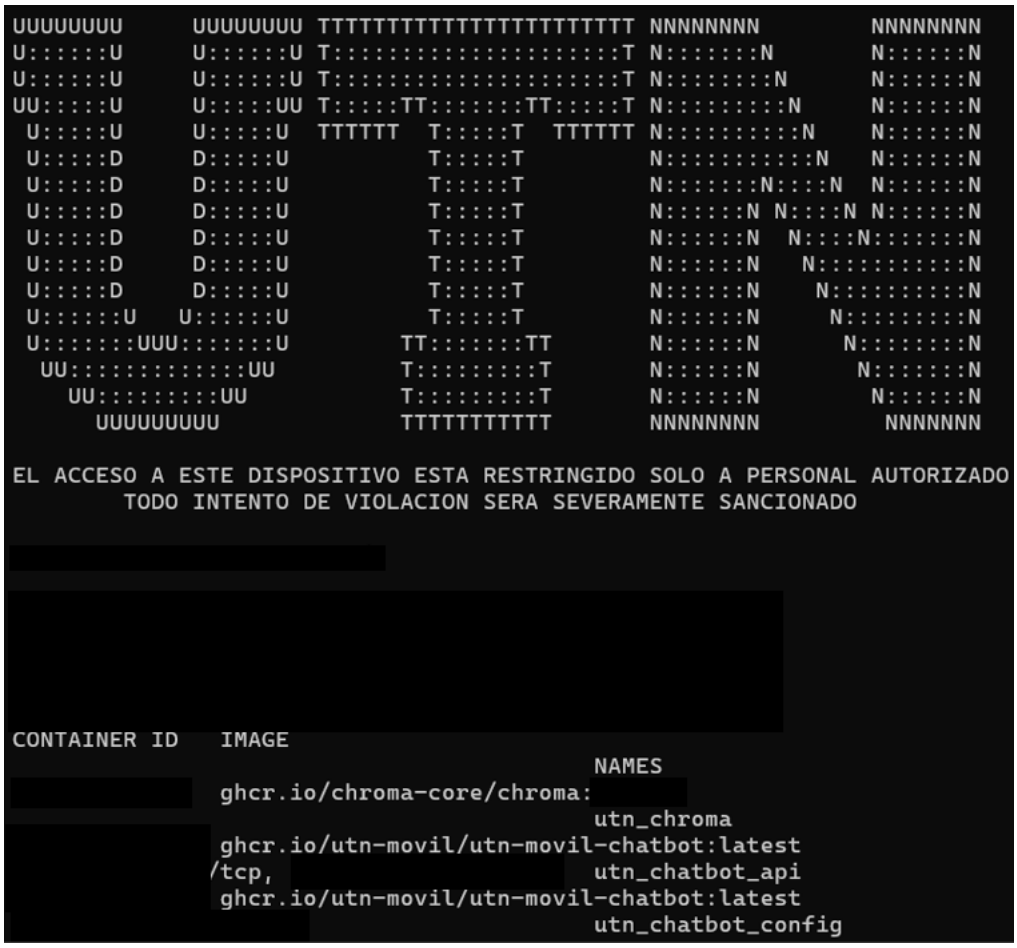


Fig. 53 Contenedores necesarios del chatbot corriendo en el ambiente universitario

Fuente: Autoría propia

- Para el entendimiento de cualquier usuario que desee administrar el chatbot, se desarrolló un manual de usuario y un manual técnico.

4.6.3 Reunión de restrospectiva – Sprint 5

Durante la reunión de retrospectiva realizada el 13/06/2025, con la participación del Product Owner, Scrum Máster y el Equipo de Desarrollo, se obtuvo como resultado el plan de mejora, el cual es evidenciado en la Tabla 39.

Tabla 40 Plan de mejora - Sprint 5

Fuente: Autoría propia

Plan de mejora	
Aciertos	• Información recolectada, formalizada, formateada e indexada a la base de conocimientos del chatbot. • Chatbot desplegado en el ambiente universitario.
Problemas	• Procesos universitarios ambiguos en su definición.

	• Dificultades al momento de desplegar el chatbot en el ambiente universitario.
Mejoras	• Reporte de uso del chatbot.

4.7 Acta de entrega y recepción

Tras culminar el desarrollo de todas las historias de usuario priorizadas en el Product Backlog, se generó la versión definitiva y validada del chatbot para la Gestión del Portafolio Estudiantil. Este artefacto fue formalmente transferido a los Product Owners mediante el acta de entrega-recepción correspondiente. Véase el **Anexo E**.

CAPITULO V
RESULTADOS Y ANALISIS

5.1 Definición del modelo para la evaluación en calidad en uso del chatbot

En conjunto con el Product Owner y el Scrum Master, se definieron las métricas de calidad en uso conforme a la Norma ISO/IEC 25022, perteneciente a la familia ISO/IEC 25000. Se eligieron indicadores principales la Eficacia, la Eficiencia y la Satisfacción [52]. Asimismo, se identificaron los pesos correspondientes a cada una, considerando su impacto en los objetivos del negocio. La experiencia previa del Product Owner sirvió como base para la determinación de estas prioridades. El modelo aplicado se evidencia en la Tabla 41.

Tabla 41 Definición del modelo en calidad en uso para el chatbot

Fuente: Autoría propia

Modelo de Calidad en Uso			
Característica	Métrica	Peso de característica	Peso de métrica
Eficacia	Tareas completas	36	10
	Objetivos logrados		10
	Tareas sin errores		16
Eficiencia	Tiempo de tareas	28	12
	Eficiencia del tiempo		16
Satisfacción	Utilidad	36	10
	Confianza		16
	Comodidad		10

5.2 Medición de calidad en uso en el chatbot

La evaluación de cada característica y sus respectivas métricas se llevó a cabo siguiendo los lineamientos de la norma ISO/IEC 25022. La eficacia y la eficiencia fueron medidas mediante la realización de un taller práctico, mientras que la satisfacción se valoró a través de la aplicación del cuestionario SUS.

5.2.1 Definición de la muestra poblacional para la evaluación

La muestra se determinó mediante un muestreo no probabilístico, debido a dos factores principales: la necesidad de mantener la confidencialidad del proyecto y la

dificultad operativa debido a la gran cantidad de estudiantes dentro de la Universidad. En consecuencia, se seleccionó un grupo de 46 estudiantes de los niveles superiores de la carrera de Ingeniería en Software, perteneciente a la Facultad de Ingeniería en Ciencias Aplicadas, por considerarse representativo de la población objetivo en función de su avance académico y la cercanía con el perfil profesional de egreso.

Con el fin de llevar a cabo la evaluación, se organizaron talleres prácticos en los que se aplicaron las métricas definidas anteriormente. La evidencia fotográfica del taller aplicado se encuentra en el **Anexo A**.

Dentro de la muestra poblacional, se identificó 2 usuarios expertos, con id 17, 38 en relación con el **Anexo C** y el **Anexo D**

5.2.2 Taller práctico

El taller se estructuró en seis objetivos, de los cuales los cuatro primeros incluyeron dos tareas específicas cada uno, mientras que los objetivos cinco y seis se conformaron con una sola tarea respectivamente, los detalles se pueden observar en la Tabla 42. La cantidad de objetivos se definió cuidadosamente con el propósito de evitar una sobrecarga o malestar en los participantes. La descripción detallada del taller se presenta en el **Anexo B**.

Para la ejecución de la evaluación, el aplicativo fue configurado de manera que la pantalla inicial correspondiera directamente a la interfaz de interacción con el chatbot. A partir de este punto se procedió a la realización de las tareas asignadas por parte de los participantes.

Tabla 42 Objetivos y tareas del taller práctico para evaluación del chatbot

Fuente: Autoría propia

Nro.	Objetivo	Tarea
1	Escribir y recibir un saludo del chatbot	Escribir un saludo al chatbot universitario Verificar que el chatbot haya respondido de manera cordial
2	Charlar informalmente con el chatbot	Escribir una pregunta informal como: ¿Cómo te encuentras hoy?

		Verificar que el chatbot responda a la pregunta de forma natural
3	Plantear preguntas fuera de contexto al chatbot	Escribir un mensaje como: Casate conmigo Verificar que el chatbot recomiende al usuario volver al contexto universitario para sus solicitudes
4	Consultar al chatbot sobre información universitaria	Escribir una pregunta al chatbot relacionada con algún proceso universitario adjunto al portafolio estudiantil, ejemplo: Quiero información sobre becas - Quiero información sobre prorrogas - Como justificar una falta Verificar que el chatbot haya respondido satisfactoriamente la consulta realizada
5	Exportar la conversación en pantalla	En la pantalla de conversación del chatbot, presione el botón con el símbolo de "PDF" para exportar la conversación en pantalla a este formato.
6	Eliminar conversación en pantalla	En la pantalla de conversación del chatbot, presione el botón con el símbolo de basurero para

	limpiar la conversación en pantalla, dejando únicamente el saludo inicial del chatbot.

5.2.3 Encuesta SUS - Evaluación de satisfacción del chatbot

La encuesta SUS (System Usability Scale) consta de diez ítems orientados a evaluar el grado de usabilidad de un sistema, tal como se detalla en la Tabla 43. Para su aplicación se emplea una escala tipo Likert de cinco puntos, en la cual el valor 1 corresponde a Totalmente en desacuerdo y el valor 5 a Totalmente de acuerdo [74].

Tabla 43 Encuesta SUS aplicada en la evaluación del chatbot

Fuente: Autoría propia

Código	Pregunta
P1	¿Considera que utilizaría esta aplicación con frecuencia?
P2	¿Considera que la aplicación es compleja?
P3	¿Considera que la aplicación es fácil de utilizar?
P4	¿Considera que necesitaría apoyo de un experto para poder utilizar esta aplicación?
P5	¿Considera que esta aplicación es fácil de aprender para la mayoría de las personas?
P6	¿Considera que la aplicación presenta inconsistencias?
P7	¿Considera que la mayoría de personas aprendería a usar este sistema rápidamente?
P8	¿Encontré el sistema muy complicado de usar?
P9	¿Se sintió seguro/a al utilizar la aplicación?
P10	¿Considera que se requieren conocimientos técnicos previos para utilizar correctamente la aplicación?

Las preguntas P1, P6 y P9 se consideran medidores de utilidad, mientras que las preguntas P2, P3, P4, P5, P7, P8 y P10 se utilizaron para la medición de la comodidad.

5.3 Validaciones estadísticas de los instrumentos de medición

5.3.1 Test de normalidad en el taller de usabilidad

Para el análisis de los resultados obtenidos en los talleres aplicados a 46 estudiantes (véase **Anexo C**), fue necesario verificar si las variables recolectadas (Errores, Tareas y Objetivos logrados) seguían una distribución normal. Esta comprobación es esencial antes de aplicar pruebas estadísticas adicionales para evaluar la usabilidad del chatbot, ya que la elección entre métodos paramétricos o no paramétricos depende de la distribución de los datos.

Con este propósito se aplicaron dos pruebas de normalidad: Kolmogorov-Smirnov y Shapiro-Wilk. La prueba de Kolmogorov-Smirnov permite contrastar la distribución empírica de los datos con una distribución normal teórica, mientras que la prueba de Shapiro-Wilk se reconoce por su mayor sensibilidad en muestras pequeñas o medianas, como en este caso (n = 46). El uso conjunto de ambos tests aporta mayor robustez en la validación de los resultados, evitando conclusiones erróneas derivadas de asumir normalidad sin comprobarla, dichas métricas son evidenciadas en la Tabla 44.

Tabla 44 Test de normalidad en taller de usabilidad

Fuente: Autoría propia

Variable	KS Estadístico	Df (KS)	KS P-value	Shapiro Estadístico	Df (Shapiro)	Shapiro P-value
Errores	0.442	46	0.000	0.398	46	0.000
Tareas logradas	0.442	46	0.000	0.398	46	0.000
Objetivos logrados	0.440	46	0.000	0.392	46	0.000

Los resultados de los tests de normalidad arrojaron valores de p menores a 0.05 en todas las variables, lo que confirma que los datos no siguen una distribución normal. Esta situación se explica porque en Tareas logradas y Objetivos logrados la mayoría alcanzó el valor máximo (efecto techo), mientras que en Errores predominó el valor cero, generando una distribución asimétrica.

En términos de usabilidad, se observó que la mayoría de los usuarios completó las tareas sin dificultad y que apenas se registraron errores. Esto sugiere que el chatbot se comporta de manera eficiente y cumple con las expectativas de interacción.

5.3.2 Test de fiabilidad en los datos obtenidos del taller de usabilidad

Para evaluar la fiabilidad del taller práctico se aplicó el coeficiente alfa de Cronbach, considerando como ítems los resultados de tareas y objetivos logrados

obtenidos por los 46 participantes. Previamente, ambas variables fueron transformadas a proporciones (0–1) con el fin de homogeneizar las escalas (tareas con un máximo de 10 y objetivos con un máximo de 6).

El análisis estadístico dio como resultado un α=0.998, lo que muestra que existe una consistencia interna muy alta. Generalmente, valores por encima de 0.70 ya se aceptan como adecuados, y al acercarse a 1.0 nos dice que los ítems están muy relacionados entre sí. Esto indica que el taller funcionó como un instrumento confiable para medir la usabilidad del chatbot.

5.3.4 Test de fiabilidad en los datos obtenidos de la encuesta SUS

Para evaluar la fiabilidad de la encuesta System Usability Scale (SUS) (véase **Anexo D**) aplicada a los 46 participantes, se calculó el coeficiente alfa de Cronbach, considerando previamente la transformación de las preguntas pares al formato necesario para la evaluación de fiabilidad. El valor obtenido fue α=0.770, lo que significa una consistencia interna aceptable.

Con el fin de profundizar en el análisis, se calcularon los coeficientes de Cronbach por dimensiones, estos se evidencian en la Tabla 44. Ambos valores se encuentran próximos al umbral de aceptación (0.70), indicando que cada dimensión presenta una fiabilidad moderada de forma independiente.

Tabla 44 Comparación de coeficientes por dimensión en la encuesta SUS

Fuente: Autoría propia

Dimensión	Ítems considerados	Alfa de Cronbach (α)	Interpretación
SUS	Q1–Q10	0.770	Consistencia aceptable
Utilidad	Q1, Q3, Q5, Q7, Q9	0.675	Fiabilidad moderada
Comodidad	Q2, Q4, Q6, Q8, Q10	0.695	Fiabilidad moderada

5.4 Evaluación del modelo de calidad en uso

Con los datos del taller y la encuesta SUS, se procedió a obtener y evidenciar los resultados de cada métrica definidos en el modelo de calidad en uso detallado en la Tabla 41.

5.4.1 Eficacia

Guiándonos del modelo definido en la Tabla 41, se puede definir cada métrica de la siguiente manera:

- Tareas completas: Se utiliza la fórmula: X=A/B, en dónde A es el número de tareas completadas por los usuarios (425), y B es el número de tareas totales (460). Como resultado obtenemos X (0.923), que indica que el 92% de las tareas se ha realizado con éxito.
- Objetivos logrados: Se utiliza la fórmula: X=A/B, en dónde A es el número de objetivos logrados por los usuarios (255), y B el número de objetivos totales (276). Como resultado obtenemos X (0.923), que indica el 92% de los objetivos se cumplieron con éxito.
- Errores en tareas: Se utiliza la fórmula: X= 1- A/B, en dónde A es el número de errores en tareas (35), y B el número de tareas totales (460). Como resultado obtenemos X (0.923), lo que indica que el 92% de las tareas fueron completadas sin errores.

5.4.2 Eficiencia

Para el cálculo de las métricas correspondientes a esta característica, se consideraron los tiempos registrados por los participantes en general y se compararon con los tiempos obtenidos por los usuarios expertos. Las métricas se definieron de la siguiente manera:

- Tiempo de tareas: Se utiliza la fórmula: X=A/B, en dónde A es el tiempo empleado por usuarios expertos para completar una tarea, y B es el tiempo empleado por los usuarios nuevos en completar una tarea. Es necesario encontrar el promedio de cada grupo, y aplicar la formula por cada una de las 10 tareas totales detalladas en el taller del **Anexo B**, para lo cual se obtiene los resultados de la Tabla 45.

Tabla 45 Comparación de tiempos en tareas entre usuarios expertos y usuarios nuevos

Fuente: Autoría propia

Tarea	Tiempos promedio de usuarios expertos (mm:ss.0)	Tiempos promedio de usuarios nuevos (mm:ss.0)	Fórmula
A1.1	00:03.3	00:11.8	0.279
A1.2	00:02.7	00:10.3	0.262
A2.1	00:04.0	00:12.3	0.325
A2.2	00:11.3	00:06.0	1.883

A3.1	00:02.0	00:15.4	0.129
A3.2	00:06.8	00:07.7	0.883
A4.1	00:13.1	00:14.0	0.935
A4.2	00:11.7	00:13.8	0.847
A5.1	00:12.4	00:13.3	0.932
A6.1	00:07.1	00:08.9	0.797

Después de obtener estos tiempos, se realiza el promedio de las fórmulas aplicadas a cada tarea, se procede a obtener el promedio (0.727), en dónde se nos dice que tenemos un 72% de eficiencia en tareas.

- Eficiencia del tiempo: Se utiliza la fórmula: X=A/B, en dónde A es el tiempo empleado por usuarios expertos en completar un objetivo, y B es el tiempo empleado por usuarios novatos en completar un objetivo. Como primera instancia es necesario obtener los tiempos por objetivo de cada usuario, para posteriormente calcular el promedio de cada grupo, y aplicar la formula por cada uno de los 6 objetivos totales detallados en el taller del **Anexo B**, para lo cual se obtienen los resultados detallados en la Tabla 46.

Tabla 46 Comparación de tiempos en objetivos entre usuarios expertos y usuarios nuevos

Fuente: Autoría propia

Objetivo	Tiempos promedio de usuarios expertos (mm:ss.0)	Tiempos promedio de usuarios nuevos (mm:ss.0)	Fórmula
O1	00:05.9	00:22.2	0.265
O2	00:15.3	00:18.3	0.836
O3	00:09.8	00:23.1	0.424
O4	00:24.8	00:27.8	0.892
O5	00:12.4	00:13.3	0.932
O6	00:07.1	00:08.9	0.797

Después de obtener estos tiempos, se realiza el promedio de las fórmulas aplicadas a cada objetivo, se procede a obtener el promedio (0.691), en dónde se nos dice que tenemos un 69% de eficiencia en objetivos.

5.4.3 Satisfacción

Para calcular las métricas correspondientes a esta característica se usó los resultados obtenidos en la encuesta SUS de la siguiente manera:

- Utilidad: Para el cálculo de esta métrica se procedió a asignar el peso según la escala de Likert dentro de las respuestas de la encuesta (véase en la Tabla 47).

Tabla 47 Peso de respuestas en la encuesta SUS aplicada

Fuente: Autoría propia

Escala	Respuesta	Peso
1	Totalmente en desacuerdo	0.2
2	En desacuerdo	0.4
3	Neutro	0.6
4	De acuerdo	0.8
5	Totalmente de acuerdo	1

Posterior a la definición de los pesos, se usa las preguntas que se relacionan directamente con la utilidad: p1, p6, p9, siendo la pregunta 6 de orientación positiva, se procedió a invertirla para que mantenga el mismo sentido. Para el procedimiento se cuantifica la elección de cada pregunta para ser multiplicada por su peso, y sumada en total, esto se evidencia en la Tabla 48.

Tabla 48 Resultados de las preguntas relacionadas a la utilidad en la encuesta

Fuente: Autoría propia

Pregunta	R=1	R=2	R=3	R=4	R=5	Total
p1	3	5	15	12	11	32.2
p6	1	2	6	12	25	39.2
p9	4	1	6	10	25	37.8

Ahora, obtenemos el promedio de los totales para obtener el número de usuarios que consideran útil el chatbot (36.4), correspondiente a la variable A en la fórmula de la forma X=A/B, en dónde B será el total de encuestados (46). Como resultado

obtenemos 0.79, lo que quiere decir que el 79% de los usuarios encuestados consideran que el chatbot es útil.

- Confianza: Para calcular esta métrica se hace uso de la fórmula X= 1- A/B, en dónde A es el número de usuarios que tuvieron reclamos dentro del taller aplicado (36), y B el número total de encuestados (46). Como resultado obtenemos 0.78, refiriéndose a un nivel de confianza del 78% de los usuarios en el chatbot.
- Comodidad: Para el cálculo de esta métrica se identificó las preguntas relacionadas a la comodidad dentro de la encuesta SUS aplicada, las cuales son: p2, p3, p4, p5, p7, p8, p10, en dónde se invirtieron las preguntas pares para conservar el mismo sentido. Para el procedimiento se cuantifica la elección de cada pregunta para ser multiplicada por su peso, y sumada en total, esto se evidencia en la Tabla 49.

  Tabla 49 Resultados de las preguntas relacionadas a la comodidad en la encuesta

  Fuente: Autoría propia

Pregunta	R=1	R=2	R=3	R=4	R=5	Total
p2	0	2	7	5	32	41
p3	1	1	4	14	26	40.2
p4	1	1	3	7	34	42
p5	0	1	5	12	28	41
p7	1	2	5	9	29	40.2
p8	0	2	0	7	37	43.4
p10	0	1	6	9	30	41.2

  Ahora, obtenemos el promedio de los totales para obtener el número de usuarios que consideran cómodo el uso del chatbot (41.2), correspondiente a la variable A en la fórmula de la forma X=A/B, en dónde B será el total de encuestados (46). Como resultado obtenemos 0.89, lo que quiere decir que el 89% de los usuarios encuestados consideran cómodo el uso del chatbot.

5.5 Resultado de la evaluación del modelo de Calidad en Uso

Después de obtener los resultados individuales por cada métrica, se procede a obtener el resultado total del modelo en calidad en uso, esto se muestra en la Tabla 50:

Tabla 50 Resultados del modelo de Calidad en Uso

Fuente: Autoría propia

Resultados del Modelo de Calidad en Uso					
Característica	Métrica	Peso de característica	Peso de métrica	Equivalente (Peso x medición obtenida)	Característica
Eficacia	Tareas completas	36	10%	9.2%	33.1%
	Objetivos logrados		10%	9.2%	
	Tareas sin errores		16%	14.7%	
Eficiencia	Tiempo de tareas	32	12%	8.7%	19.7%
	Eficiencia del tiempo		16%	11.0%	
Satisfacción	Utilidad	42	10%	7.9%	29.3%
	Confianza		16%	12.5%	
	Comodidad		10%	8.9%	

Sumando los totales de cada característica, obtenemos como resultado un 82.1% de calidad en uso respecto al chatbot. Al final, tenemos que el chatbot se encuentra en un rango satisfactorio en cuanto a calidad en uso de acuerdo con las evaluaciones, esto guiándonos a partir de lo que se establece en la ISO 25022 [47], en la Figura 54 se ilustra de mejor manera como el chatbot se encuentra dentro del rango objetivo de satisfacción.

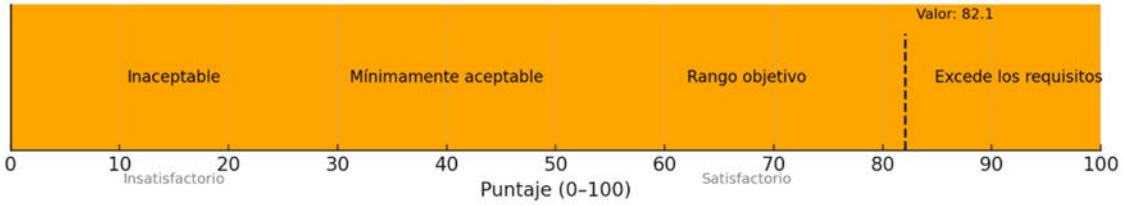


Fig. 54 Calidad en uso del chatbot basada en la ISO 25022

Fuente: Autoría propia

5.6 Discusión

5.6.1 Interpretación de los resultados

Al evaluar la calidad en uso del chatbot se obtuvo un resultado global del 82,1%. Según la norma ISO/IEC 25022 (Fig. 54), esta cifra se encuentra dentro del rango considerado satisfactorio. El puntaje refleja la contribución de las tres dimensiones medidas (Tabla 50): eficacia con 33,1%, eficiencia con 19,7% y satisfacción con 29,3%.

En eficacia, el mayor aporte provino de tareas sin errores (14,7%), seguido de tareas completas y objetivos logrados (9,2% cada una). Esto sugiere que, una vez que el usuario interactúa con el sistema, el flujo de resolución es estable y con baja tasa de fallos, lo que es coherente con la naturaleza orientada a consulta del portafolio estudiantil.

La eficiencia alcanzó un 19,7%, lo que representa la dimensión con menor aporte. Este valor se compone de un 11,0% por eficiencia del tiempo y un 8,7% por el tiempo invertido en tareas. Aunque los tiempos fueron aceptables para el ámbito académico, todavía hay espacio para mejorar la rapidez de respuesta y simplificar los pasos por tarea. Si se hacen ajustes en los mecanismos de recuperación y generación podrían dar un mejor desempeño en futuras pruebas.

La satisfacción llegó al 29,3%, sostenida por la confianza (12,5%), la comodidad (8,9%) y la utilidad (7,9%). Esto nos indica que los estudiantes consideran confiables las respuestas y encuentran en la interfaz un medio cómodo y claro para interactuar.

En conjunto, el sistema mostró un perfil característico: buena eficacia y satisfacción, pero con una eficiencia menor.

5.6.2 Relación con la ISO/IEC 25022

La ISO/IEC 25022 propone medir la calidad en uso a partir de métricas observables por el usuario en su entorno de operación. Los resultados obtenidos confirman que:

- El sistema logra el objetivo de uso: los estudiantes pudieron completar sus tareas con pocos errores, lo que refleja su eficacia.
- La percepción de los usuarios fue positiva: utilidad, confianza y comodidad se situaron en valores satisfactorios.
- Los tiempos de respuesta fueron adecuados para el uso académico, aunque aún pueden optimizarse para alcanzar un nivel superior de eficiencia.

En conjunto, los hallazgos respaldan la validez de la metodología usada y sirven como base para continuar con mejoras en etapas posteriores.

5.6.3 Comparación con trabajos previos

A diferencia de enfoques tradicionales - por ejemplo, desarrollos apoyados en Microsoft Bot Framework[54] o Dialogflow[55] - que suelen requerir fases explícitas de entrenamiento o construcción de intenciones y flujos rígidos, el chatbot propuesto se apoya en un LLM y en recuperación basada en documentos institucionales. Esta arquitectura presenta dos ventajas:

- Reducción de costos de mantenimiento: no se necesita reentrenar el modelo para ampliar el corpus; basta con gestionar el repositorio documental (ingesta/actualización), lo que disminuye costos operativos y tiempos de actualización frente a cambios normativos.
- Comportamiento más "humano": en lugar de respuestas definidas con clave - valor, el chatbot razona sobre su base de conocimiento, llegando a la altura expectativa actual de usuarios acostumbrados a IA generativa que produce respuestas robustas.

En cuanto a la usabilidad, este enfoque hace que la interacción sea más sencilla y natural. Los estudiantes no tienen que memorizar palabras clave ni recorrer menús extensos, sino que pueden preguntar directamente en su propio estilo y recibir respuestas apoyadas en la normativa oficial de la universidad.

5.6.4 Implicaciones prácticas para la gestión del portafolio estudiantil

Los resultados muestran beneficios claros para la comunidad universitaria. Por un lado, los estudiantes pueden acceder más rápido a normativas y procedimientos, lo que disminuye tiempos de espera y dependencia del personal administrativo. Además, las respuestas mantienen un nivel de consistencia que genera confianza frente a preguntas repetitivas, a esto se suma que la reducción de consultas manuales libera a las unidades responsables para atender casos más complejos.

Para llevar el servicio a un nivel superior en cuanto a usabilidad, se proponen acciones técnicas, como guardar en caché los resultados más frecuentes, optimizar el recuperador de fragmentos de la documentación con ajustes de parámetros y segmentación, emplear técnicas de compresión del contexto y administrar las sesiones de manera que aprovechen el historial previo de la conversación.

5.6.5 Limitaciones y amenazas a la validez

El chatbot es capaz de responder a una consulta de manera satisfactoria, citando de manera precisa la fuente de donde se basa su respuesta, no obstante, muestra dificultad,

o directamente falla cuando la consulta contiene 2 solicitudes de diferente contexto al mismo tiempo, por ejemplo: “Me gustaría que me des información sobre requisitos para aplicar a una beca, y también quiero que me digas como agendo una cita en odontología”. Por el momento, el chatbot está limitado a enfocarse en un solo contexto por pregunta para garantizar precisión, siendo capaz de responder 2 solicitudes al mismo tiempo, siempre y cuando estas se encuentren dentro del mismo contexto.

El chatbot está compuesto de 2 agentes de inteligencia artificial principales, siendo capaz de responder preguntas basadas en la documentación oficial que se le haya cargado, así mismo como responder preguntas basándose en la base de datos a la que este tenga acceso, no obstante, el chatbot solo puede usar un agente a la vez por consulta, por lo tanto, en preguntas que necesiten el uso de los dos agentes en conjunto, el chatbot puede optar por priorizar la pregunta que necesite mas atención.

Por petición de los Stakeholders, y para reducir costos operativos (consumo de tokens), se ha optado que el chatbot no guarde memoria entre mensajes, es decir, no mantiene el hilo de conversación, sino, se enfoca en responder de manera satisfactoria la consulta que a este se le realice en cada petición. Es necesario que, para cada pregunta, el usuario introduzca todo el contexto de su requerimiento evitando citar consultas anteriores. Ademas, para evitar costos y saturación de la base de datos universitaria, el chatbot no guarda el historial de conversaciones de los usuarios ni sus interacciones dentro de la base de datos universitaria, por lo tanto, este desconoce el contexto anterior al usuario que precede a su solicitud.

El chatbot es agnóstico al usuario que hace la petición, no maneja la información personal para el contexto o personalización durante sus consultas, contestando de manera general que no personaliza su modo de respuesta dependiendo del usuario. Se priorizó el uso de este como un chatbot para consultas, más que un asistente personalizado. El chatbot no está conectado ni ligado a la información de cada usuario dentro de la base de datos, no recupera ni recolecta su información personal, ni es necesaria dicha información o conexión para su funcionamiento.

La carga de información documental está limitada solo a los usuarios administradores, los estudiantes o usuarios finales del chatbot no podrán cargar nueva información documental para el chatbot. Además, la carga de información documental se limita a documentos de texto plano, siendo incapaz de procesar información de imágenes, o audio.

El chatbot solo maneja 2 proveedores de LLMs para su funcionamiento de manera dinámica, siendo estos, Ollama para LLMs locales y OpenAI para LLMs en la nube. El sistema no soporta LLMs de otros proveedores que no sean los ya mencionados.

El agente SQL del sistema es capaz de ingresar a una base de datos para obtener información a la que se le haya dado permiso, sin embargo, las bases de datos a las que se desee conectar el chatbot deben estar basadas estrictamente en Oracle, de lo contrario, el chatbot no podrá conectarse.

En cuanto a la evaluación del chatbot, las pruebas se centraron en una visión general de la calidad en uso. Aunque sirvieron para medir satisfacción y éxito en las tareas, también dejaron ver algunas limitaciones. Por ejemplo, los escenarios evaluados fueron los más habituales, por lo que no está claro cómo respondería el sistema en casos poco comunes. También hay que considerar que la latencia y el rendimiento dependen de factores externos como la concurrencia de usuarios, el tamaño del corpus o los recursos disponibles. Finalmente, no se analizaron por separado los distintos componentes internos, lo que impide atribuir con precisión el aporte de cada uno al resultado final. Estas limitaciones no invalidan los hallazgos, pero sí recuerdan que todavía hay espacio para mejoras en futuros trabajos.

5.6.6 Líneas de trabajo futuro

Como primera línea de trabajo futuro, se plantea la mejora de los mecanismos de recuperación de información documental mediante la incorporación de criterios temporales asociados a la fecha de indexación de los documentos. Esto permitiría priorizar automáticamente versiones más recientes de normativas, reglamentos o procedimientos institucionales cuando existan múltiples documentos relacionados con un mismo tema. La inclusión de este filtro no solo contribuiría a reducir el riesgo de respuestas desactualizadas, sino que también fortalecería la confiabilidad del sistema frente a errores humanos en la gestión del repositorio documental, aspecto crítico en entornos universitarios donde la normativa puede cambiar con frecuencia.

Otra línea relevante de trabajo futuro está relacionada con la capacidad del chatbot para manejar consultas complejas que involucren múltiples contextos o fuentes de información. En este sentido, se podría explorar la incorporación de mecanismos de descomposición automática de preguntas, que permitan dividir una consulta compuesta en subconsultas más simples, cada una asignada al agente más adecuado. Esta mejora permitiría superar la limitación actual de utilizar un solo agente por petición y habilitaría respuestas más completas en escenarios donde sea necesario combinar información

documental con datos provenientes de la base de datos institucional, sin sacrificar precisión ni trazabilidad.

Se considera pertinente analizar la viabilidad de incorporar algún grado de memoria conversacional controlada. Aunque la decisión actual de no mantener contexto entre mensajes responde a criterios de reducción de costos y protección de recursos institucionales, futuras investigaciones podrían evaluar esquemas híbridos, como memoria de corto plazo o resúmenes temporales por sesión, que permitan mantener coherencia en interacciones prolongadas sin incurrir en un consumo elevado de tokens ni en el almacenamiento permanente de información sensible. Este enfoque podría mejorar significativamente la experiencia de uso del chatbot en escenarios donde el usuario necesita realizar consultas encadenadas.

En relación con la arquitectura del sistema, una línea de trabajo futuro importante consiste en ampliar la compatibilidad con otros proveedores de modelos de lenguaje y motores de bases de datos. La incorporación de LLMs adicionales, tanto locales como en la nube, permitiría comparar costos, rendimiento y calidad de respuesta en diferentes contextos operativos. De forma similar, extender el agente SQL para soportar otros sistemas gestores de bases de datos, más allá de Oracle, aumentaría la portabilidad y reutilización del chatbot en otras instituciones o escenarios organizacionales, reduciendo la dependencia tecnológica actual.

Respecto a la gestión de la información documental, futuras mejoras podrían enfocarse en ampliar los tipos de archivos soportados por el sistema. La capacidad de procesar documentos que incluyan imágenes, tablas complejas o contenido audiovisual permitiría enriquecer la base de conocimiento del chatbot y ampliar su alcance funcional. La integración de técnicas de reconocimiento óptico de caracteres o transcripción automática podría ser un primer paso hacia un sistema más robusto en términos de ingestión de información heterogénea.

Es necesario indagar en una mejora del chatbot para convertirse en un asistente personalizado para cada estudiante dentro de su propio entorno universitario, siendo capaz de acceder a información precisa de cada usuario para ofrecer consultas mas personales respecto a su desempeño académico y contexto individual, permitiendo a su vez, personalizar a los agentes conversacionales de acuerdo con cada estudiante.

En cuanto a la evaluación del chatbot, se identifican múltiples oportunidades de profundización metodológica. Además de las pruebas centradas en la calidad en uso, sería valioso realizar evaluaciones técnicas específicas sobre los distintos componentes

internos del sistema, tales como el desempeño de los modelos de embeddings, los mecanismos de recuperación y reordenamiento de información, y la latencia asociada a cada agente de manera individual y conjunta. Complementar estos análisis con pruebas de carga, mediciones de consumo de recursos y escenarios menos frecuentes permitiría obtener una visión más completa del comportamiento del chatbot bajo condiciones reales de uso, sentando bases sólidas para su escalabilidad y mejora continua. En el siguiente punto se indaga formas de evaluación del chatbot propuestas para trabajos futuros.

5.6.7 Puntos de vista para la evaluación del chatbot en trabajos futuros

La evaluación de sistemas conversacionales basados en inteligencia artificial generativa plantea nuevos desafíos que no pueden ser abordados únicamente con métricas tradicionales. En esta sección, se plantea diferentes líneas de evaluación para un chatbot basado en LLMs.

Una primera línea de evaluación futura corresponde a la medición de la calidad de las respuestas generadas por el chatbot. Si bien métricas clásicas de generación de texto como BLEU, ROUGE o METEOR han sido utilizadas históricamente para evaluar sistemas de lenguaje natural, diversos estudios han evidenciado que estas presentan una correlación limitada con la percepción humana de calidad en contextos conversacionales abiertos [75]. Como alternativa, investigaciones recientes proponen el uso de modelos de lenguaje como evaluadores automáticos, capaces de juzgar aspectos como coherencia, relevancia, alineación con la pregunta y corrección factual de las respuestas [76]. Este enfoque resulta particularmente pertinente para sistemas basados en recuperación aumentada por generación, donde la evaluación puede extenderse a verificar si la respuesta generada se encuentra correctamente sustentada en las fuentes recuperadas [77].

Otra forma relevante de evaluación se basa en el cumplimiento de tareas y objetivos por parte del chatbot. En sistemas orientados a la resolución de consultas concretas, como la obtención de información cargada en un corpus, o la ejecución de consultas sobre bases de datos, se pueden definir métricas de éxito que midan si el agente logra alcanzar el objetivo esperado de manera correcta [78]. Estas métricas, comúnmente denominadas tasas de éxito, permiten evaluar de forma directa la efectividad funcional del sistema. Adicionalmente, se han propuesto métricas que consideran no solo el resultado final, sino también el esfuerzo requerido por el agente para completar una tarea, medido en número de pasos, llamadas a herramientas o acciones intermedias [79].

Desde una perspectiva centrada en la interacción, futuras evaluaciones podrían incorporar enfoques dinámicos que consideren el comportamiento del chatbot en

escenarios conversacionales reales. A diferencia de evaluaciones estáticas basadas en conjuntos de preguntas predefinidas, la evaluación interactiva permite observar cómo el sistema responde a variaciones en el lenguaje del usuario, ambigüedades o consultas encadenadas [80]. En este contexto, métricas como la latencia de respuesta, el tiempo hasta el primer token generado y el tiempo total de generación adquieren especial relevancia, ya que influyen directamente en la percepción de usabilidad del sistema [81]. Asimismo, el análisis del consumo de recursos computacionales y de tokens procesados resulta fundamental para estimar la viabilidad operativa del chatbot en entornos institucionales con múltiples usuarios concurrentes [82].

La evaluación humana sigue siendo un componente clave en la validación de sistemas conversacionales. Estudios con usuarios finales, encuestas de satisfacción, escalas tipo Likert y evaluaciones permiten capturar dimensiones subjetivas como claridad, utilidad percibida y confianza en las respuestas, aspectos que difícilmente pueden ser evaluados de manera automática [83]. En trabajos futuros, la combinación de evaluaciones automáticas y humanas permitiría obtener una visión más completa del desempeño del chatbot, especialmente en interacciones prolongadas o escenarios multi-turno donde la coherencia del diálogo cobra mayor importancia [84].

Recientemente se ha propuesto el uso de marcos de evaluación integrales y benchmarks especializados para sistemas basados en agentes y modelos de lenguaje. Estos marcos establecen una relación explícita entre los objetivos del sistema, las tareas evaluadas, los conjuntos de datos utilizados y las métricas seleccionadas, permitiendo una evaluación sistemática y reproducible [85]. Asimismo, la adopción de entornos de prueba estandarizados y benchmarks diseñados para agentes conversacionales facilita la comparación objetiva entre distintas soluciones y arquitecturas [86].

Actualmente existen diferentes propuestas para marcos de evaluación de chatbots generativos, asimismo como agentes de inteligencia artificial, centrados principalmente en usuario como principal evaluador. En el presente trabajo, la aplicación de un marco de evaluación de este tipo permitiría analizar de manera aislada el desempeño de cada agente del chatbot, así como su contribución al resultado final, sentando las bases para mejoras estructurales y optimizaciones futuras. Es posible no descartar la propuesta de un propio marco de evaluación como trabajo futuro, centrado en evaluaciones técnicas, como en evaluaciones centradas en el usuario.

5.6.8 Síntesis

El chatbot alcanza calidad en uso satisfactoria (82,1%) con alta eficacia y elevada satisfacción, y muestra oportunidades de mejora en eficiencia. Su arquitectura basada en LLM con recuperación documental reduce costos de mantenimiento y eleva la naturalidad de la interacción frente a enfoques de intención/flujo preconfigurado. El siguiente salto cualitativo requiere gobernanza de la vigencia documental y una caracterización técnica de los componentes que permita optimizar latencia y desempeño bajo escala. Con estas acciones, el sistema tiene potencial para exceder los requisitos de calidad en uso definidos por la ISO/IEC 25022 y consolidarse como soporte confiable para el estudiante.

CONCLUSIONES

La adopción de Ollama demostró la viabilidad de ejecutar LLMs locales dentro del proyecto. No obstante, varias utilidades avanzadas de LangGraph, especialmente las Tools diseñadas para servicios SaaS como GPT o Mistral, presentan limitaciones de compatibilidad con este cliente local. Preservar la portabilidad y garantizar un código agnóstico al modelo nos permite mantener la arquitectura funcional con cualquier LLM disponible al momento de la implementación.

En las pruebas prácticas se vio una diferencia clara: los modelos locales (Ollama) respondían más rápido, mientras que los de la nube (OpenAI-GPT) ofrecían respuestas más precisas, aunque con mayor demora. Esto confirma que la elección depende del uso: para rapidez conviene lo local, y para exactitud lo remoto.

La arquitectura buscó un equilibrio: evitar llamadas innecesarias y aprovechar modelos más livianos, que reducen costos sin afectar la calidad de las respuestas. Aunque los modelos grandes ayudan a disminuir alucinaciones, también elevan mucho el gasto en cómputo.

Otro aspecto fue el diseño de los prompts. Los modelos grandes permiten usar mensajes más breves, pero apoyarse solo en ellos limitaría la independencia del chatbot. Por eso se usaron prompts más explícitos, que aunque consumen más tokens, aseguran que el sistema sea compatible con diferentes proveedores y versiones.

Aunque el LLM constituye el núcleo generativo, la calidad de la respuesta depende fundamentalmente de la recuperación de información desde ChromaDB y de la correcta estructuración de los documentos ingeridos. Sin un contexto relevante y bien formateado, incluso el modelo más avanzado mostrará un rendimiento subóptimo.

Cuando se piensa en un uso intensivo del chatbot, los modelos en la nube son más cómodos porque se adaptan solos al tráfico y reducen la carga de mantenimiento. En cambio, correr el modelo en local es útil si se tiene insfraestructura y se trabaja en un entorno con poca o nula conectividad.

Se decidió no guardar todo el historial de la conversación. Con esto se logra respuestas más claras en cada consulta, aunque se pide al usuario que formule sus preguntas completas.

Finalmente, se descarta emplear la técnica de fine-tuning, esto porque requiere mucho tiempo y dinero para entrenar y mantener el modelo. Además, como los modelos se encuentran en constante evolución, cualquier ajuste corre el riesgo de quedar desfasado en poco tiempo.

RECOMENDACIONES

Para continuar con este proyecto, lo más práctico es centrar los esfuerzos en mejorar la manera en que se formulan las instrucciones al modelo. Afinar los prompts con técnicas claras y reutilizables puede marcar una gran diferencia en la calidad de las respuestas, sin necesidad de invertir en entrenamientos costosos o difíciles de mantener.

La utilidad del sistema depende de que los documentos estén bien organizados, sin duplicados y en un formato consistente. Se recomienda un proceso periódico de revisión y depuración, acompañado de métricas simples que permitan saber si la base está completa y actualizada, ayudaría a mantener el servicio confiable en el tiempo.

Desde la perspectiva de los usuarios, se recomienda aplicar una evaluación constante. Encuestas breves, análisis de cómo interactúan los estudiantes y seguimientos periódicos darán pistas claras sobre la satisfacción y los puntos que requieren ajustes. Este monitoreo permitirá tomar decisiones oportunas y alinearlas con los objetivos de la Universidad.

Por último, sería interesante probar en el futuro un sistema de memoria resumida (Si la infraestructura futura de la UTN lo permite). De esa forma, el chatbot podría retener solo lo esencial de cada sesión, ofreciendo interacciones más fluidas sin comprometer la rapidez ni la eficiencia.

Referencias Bibliográficas

[1] Naranjo-Toro Miguel, “UNIVERSIDAD TÉCNICA DEL NORTE 14 - Portafolios Universitarios,” Ibarra, Oct. 2014.

[2] Universidad Técnica del Norte, “UTN Página oficial.” Accessed: Oct. 06, 2024. [Online]. Available: https://www.utn.edu.ec/

[3] N. Unidas, “Transformar nuestro mundo: la Agenda 2030 para el Desarrollo Sostenible,” Asamblea General de las Naciones Unidas, 2015.

[4] S. N. de Planificación, “Plan de Creación de Oportunidades 2021-2025,” Quito, Ecuador, 2021. [Online]. Available: http://www.planificacion.gob.ec

[5] Subsecretaría de Fomento de la Sociedad de la Información y Economía Digital, J. Chiluiza, Ortega Jorge, and Sotaminga Marcelo, “Diagnostico Sobre la Inteligencia Artificial en el Ecuador,” Dec. 2021.

[6] V. Sharma, M. Goyal, and D. Malik, “An intelligent behaviour shown by chatbot system,” *International Journal of New Technology and Research*, vol. 3, no. 4, p. 263312, 2017.

[7] A. Berdasco, G. López, I. Diaz, L. Quesada, and L. A. Guerrero, “User experience comparison of intelligent personal assistants: Alexa, Google Assistant, Siri and Cortana,” in *Proceedings*, MDPI, 2019, p. 51.

[8] P. Knob *et al.*, “Surveying the evolution of virtual humans expressiveness toward real humans,” *Comput Graph*, vol. 123, p. 104034, Oct. 2024, doi: 10.1016/J.CAG.2024.104034.

[9] Google, “Google Assistant.” Accessed: Dec. 16, 2024. [Online]. Available: https://assistant.google.com/intl/es_es/

[10] A. Vaswani, “Attention is all you need,” *Adv Neural Inf Process Syst*, 2017, Accessed: Jan. 07, 2025. [Online]. Available: https://arxiv.org/pdf/1706.03762

[11] M. O. Topal, A. Bas, and I. van Heerden, “Exploring transformers in natural language generation: Gpt, bert, and xlnet,” *arXiv preprint arXiv:2102.08036*, 2021.

[12] Z. Jiang *et al.*, “Active retrieval augmented generation,” *arXiv preprint arXiv:2305.06983*, 2023.

[13] P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *Adv Neural Inf Process Syst*, vol. 33, pp. 9459–9474, 2020.

[14] Y. Gao *et al.*, “Retrieval-augmented generation for large language models: A survey,” *arXiv preprint arXiv:2312.10997*, 2023.

[15] Y. Dong, S. Wang, H. Zheng, J. Chen, Z. Zhang, and C. Wang, “Advanced RAG Models with Graph Structures: Optimizing Complex Knowledge Reasoning and Text Generation,” *arXiv preprint arXiv:2411.03572*, 2024.

[16] Y. Gao, Y. Xiong, M. Wang, and H. Wang, “Modular rag: Transforming rag systems into lego-like reconfigurable frameworks,” *arXiv preprint arXiv:2407.21059*, 2024.

[17] P. Rathod, “Efficient Usage of RAG Systems in the World of LLMs,” *Authorea Preprints*, 2024.

[18] Z. F. Han *et al.*, “Improving assessment of tutoring practices using retrieval-augmented generation,” *arXiv preprint arXiv:2402.14594*, 2024.

[19] Z. Levonian *et al.*, “Retrieval-augmented generation to improve math question-answering: Trade-offs between groundedness and human preference,” *arXiv preprint arXiv:2310.03184*, 2023.

[20] F. Miladi, V. Psyché, and D. Lemire, “Leveraging GPT-4 for Accuracy in Education: A Comparative Study on Retrieval-Augmented Generation in MOOCs,” in *International Conference on Artificial Intelligence in Education*, Springer, 2024, pp. 427–434.

[21] Y. Lee, “Developing a computer-based tutor utilizing Generative Artificial Intelligence (GAI) and Retrieval-Augmented Generation (RAG),” *Educ Inf Technol (Dordr)*, pp. 1–22, 2024.

[22] M. Thway, J. Recatala-Gomez, F. S. Lim, K. Hippalgaonkar, and L. W. T. Ng, “Harnessing GenAI for Higher Education: A Study of a Retrieval Augmented Generation Chatbot’s Impact on Human Learning,” *arXiv preprint arXiv:2406.07796*, 2024.

[23] S. Arora and S. Bedathur, “On embeddings in relational databases,” *arXiv preprint arXiv:2005.06437*, 2020.

[24] J. Huber and A. Ratner, “Chroma.” Accessed: Dec. 08, 2024. [Online]. Available: https://docs.trychroma.com/about

[25] C. Popescu, “Vergleichsanalyse der Vektordatenbanken ChromaDB und Qdrant für die semantische Suche,” Bachelor Thesis, Technische Hochschule Ingolstadt, Ingolstadt, 2024.

[26] H. Chase, “Langchain.” Accessed: Jan. 07, 2025. [Online]. Available: https://www.langchain.com/

[27] S. Vidivelli, M. Ramachandran, and A. Dharunbalaji, “Efficiency-Driven Custom Chatbot Development: Unleashing LangChain, RAG, and Performance-Optimized LLM Fusion,” *Computers, Materials and Continua*, vol. 80, no. 2, pp. 2423–2442, Aug. 2024, doi: 10.32604/CMC.2024.054360.

[28] S. Prabhune and D. J. Berndt, “Deploying Large Language Models With Retrieval Augmented Generation,” *arXiv preprint arXiv:2411.11895*, 2024.

[29] Y. Jeong and S. Park, “Are Streaming engines and Vector Databases Integrated Well?,” Nov. 2022, Accessed: Jan. 07, 2025. [Online]. Available: https://discos.sogang.ac.kr/file/2024/intl_conf/PDSW24_WIP_VD.pdf

[30] M. Chiang and J. Morgan, “Ollama.” Accessed: Dec. 08, 2024. [Online]. Available: https://ollama.com/

[31] ENIIT Innova Business School, “Ollama y la IA generativa en ciberseguridad,” Campus Internacional de Ciberseguridad. Accessed: Dec. 08, 2024. [Online]. Available: https://www.campusciberseguridad.com/blog/item/190-ollama-inteligencia-artificial-generativa-en-ciberseguridad

[32] I. Radeva, I. Popchev, L. Doukovska, and M. Dimitrova, “Web Application for Retrieval-Augmented Generation: Implementation and Testing,” *Electronics (Basel)*, vol. 13, no. 7, p. 1361, 2024.

[33] O. Kamal, “Ollama: El panorama de los potentes LLMs de código abierto,” Medium. Accessed: Dec. 08, 2024. [Online]. Available: https://medium.com/@omkamal/ollama-the-landscape-for-a-powerful-llm-from-meta-ai-6792d7dad718

[34] V. Mavroudis, “LangChain,” *Preprints.org*, Nov. 2024, doi: 10.20944/preprints202411.0566.v1.

[35] E. Nascimento *et al.*, “A family of natural language interfaces for databases based on chatgpt and langchain,” in *Proc. 42nd Int. Conf. on Conceptual Modeling–Posters&Demos, Lisbon, Portugal*, 2023.

[36] I. Docker, “Docker,” 2020, Accessed: Dec. 10, 2024. [Online]. Available: https://www. docker. com/what-docker

[37] T. Combe, A. Martin, and R. Di Pietro, “To docker or not to docker: A security perspective,” *IEEE Cloud Computing*, vol. 3, no. 5, pp. 54–62, 2016.

[38] NVIDIA, “¿Qué es CUDA?,” NVIDIA. Accessed: Dec. 10, 2024. [Online]. Available: https://support.nvidia.eu/hc/es/articles/4850516229266--Qu%C3%A9-es-CUDA

[39] B. Chen, “An Exploration of Using Retrieval-Augmented Generation With Access Control,” 2024.

[40] L. Dagne, “Flutter for cross-platform App and SDK development,” May 2019, Accessed: Jan. 07, 2025. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/172866/Lukas%20Dagne%20Thesis.pdf?sequence=2

[41] A. Nirumand Jazi, I. Alfonso, and J. Cabot, “Low-Code Flutter Application Development Solution,” in *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, 2024, pp. 838–847.

[42] S. Y. Ameen and D. Y. Mohammed, “Developing cross-platform library using flutter,” *European Journal of Engineering and Technology Research*, vol. 7, no. 2, pp. 18–21, 2022.

[43] V. N. SARA, “LA METODOLOGÍA CRISP-DM,” 2016, Accessed: Jan. 07, 2025. [Online]. Available:

http://ri.uaemex.mx/bitstream/handle/20.500.11799/64111/secme-12408.pdf?sequence=1

[44] M. Mohanan, “Competitive Analysis of Embedding Models in Retrieval-Augmented Generation for Indian Motor Vehicle Law Chatbots,” Dublin, Jan. 2024.

[45] C. Rodríguez and R. D. Vicente, “¿ Por qué implementar Scrum?,” *Revista Ontare*, vol. 3, no. 1, pp. 125–144, 2015.

[46] M. Trigás Gallego, “Metodologia scrum,” *Universitat Oberta de Catalunya*, Jun. 2012, Accessed: Jan. 07, 2025. [Online]. Available: https://openaccess.uoc.edu/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf

[47] H. Nakai, N. Tsuda, K. Honda, H. Washizaki, and Y. Fukazawa, “Initial framework for software quality evaluation based on iso/iec 25022 and iso/iec 25023,” in *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE, 2016, pp. 410–411.

[48] J. Oviedo, M. Rodriguez, A. Trenta, D. Cannas, D. Natale, and M. Piattini, “ISO/IEC quality standards for AI engineering,” *Comput Sci Rev*, vol. 54, p. 100681, 2024.

[49] B. Jaradat and G. Weheba, “A PRACTICAL GUIDE TO ISO 25022: MEASUREMENT OF SOFTWARE QUALITY IN-USE,” *EDITORIAL REVIEW BOARD*, p. 47, 2024.

[50] H. Nakai, N. Tsuda, K. Honda, H. Washizaki, and Y. Fukazawa, “Initial framework for software quality evaluation based on iso/iec 25022 and iso/iec 25023,” in *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE, 2016, pp. 410–411.

[51] N. Bevan, J. Carter, J. Earthy, T. Geis, and S. Harker, “New ISO standards for usability, usability reports and usability measures,” in *Human-Computer Interaction. Theory, Design, Development and Practice: 18th International Conference, HCI International 2016, Toronto, ON, Canada, July 17-22, 2016. Proceedings, Part I 18*, Springer, 2016, pp. 268–278.

[52] S. Patiño, “EVALUACIÓN DE LA SATISFACCIÓN QUE BRINDA UN ASISTENTE VIRTUAL CON INTELIGENCIA ARTIFICIAL EN LA MESA DE SERVICIOS DE TI PARA DETERMINAR LA CALIDAD EN USO BASADO EN LA NORMA 25022.,” PREVIO AL GRADO ACADÉMICO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN , PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE ESMERALDAS, Esmeraldas, 2021.

[53] J. Sulla-Torres, A. Gutierrez-Quintanilla, H. Pinto-Rodriguez, R. Gómez-Campos, and M. A. Cossio-Bolaños, “Quality in use of an android-based mobile application for calculation of bone mineral density with the standard ISO/IEC 25022,” 2020.

[54] W. G. Puma Quilumba, “Implementación de un chatbot como estrategia de apoyo en el servicio de soporte bibliotecario de la Universidad Técnica del Norte aplicando técnicas de procesamiento de lenguaje natural,” Universidad Técnica del Norte, Ibarra, 2023. [Online]. Available: https://repositorio.utn.edu.ec/handle/123456789/14912

[55] A. J. Morán Bastidas, “Implementación de un asistente virtual (CHATBOT) para el blog de la Carrera de Software de la Universidad Técnica del Norte utilizando inteligencia artificial,” Universidad Técnica del Norte, Ibarra, 2023. [Online]. Available: https://repositorio.utn.edu.ec/handle/123456789/14731

[56] Q. Zhou *et al.*, “GastroBot: a Chinese gastrointestinal disease chatbot based on the retrieval-augmented generation,” *Front Med (Lausanne)*, vol. 11, p. 1392555, 2024.

[57] T. Feridooni *et al.*, “Development of a vascular surgery-specific artificial intelligence chat interface using retrieval-augmented generation: VASC.AI, a specialized vascular surgery chatbot,” *JVS-Vascular Insights*, vol. 2, p. 100137, 2024, doi: 10.1016/j.jvsvi.2024.100137.

[58] P. De Privacidad, Y. Tratamiento, and D. E. Datos, “UNIVERSIDAD TÉCNICA DEL NORTE,” Ibarra, Mar. 2025. [Online]. Available: www.utn.edu.ec

[59] Scrum Manager, “Historia de usuario.” Accessed: Jun. 21, 2025. [Online]. Available: https://www.scrummanager.com/bok/index.php/Historia_de_usuario

[60] A. Menzinsky, G. López, J. Palacio, M. Á. Sobrino, R. Álvarez, and V. Rivas, “Historias de usuario,” *Ingeniería de requisitos ágil*, 2018.

[61] S. M. Yépez García, “Proceso de Atención - Centro de Educación Inicial ‘Chispitas de Ternura,’” Universidad Técnica del Norte, Dirección de Bienestar Universitario, Ibarra, Ecuador, 2025. [Online]. Available: http://www.utn.edu.ec/

[62] D. De, B. Universitario, and V. Académico, “Flujograma de Atenciones Médicas y de Enfermería,” 2025. [Online]. Available: www.utn.edu.ec

[63] U. T. del N. Dirección de Bienestar Universitario, “Proceso de Atención - Laboratorio Clínico,” Universidad Técnica del Norte, Ibarra, Ecuador, 2025. Accessed: Aug. 25, 2025. [Online]. Available: www.utn.edu.ec

[64] C. S. Benavides Morillo, “Proceso de Atención en el Servicio Odontológico,” Universidad Técnica del Norte, Dirección de Bienestar Universitario, Ibarra, Ecuador, 2025. [Online]. Available: http://www.utn.edu.ec/

[65] U. T. del Norte, “Reglamento de la Unidad de Integración Curricular de Grado,” Universidad Técnica del Norte, Honorable Consejo Universitario, Ibarra, Ecuador, 2022. Accessed: Aug. 25, 2025. [Online]. Available: http://www.utn.edu.ec/

[66] U. T. del Norte, “Instructivo para la Distribución de Actividades del Personal Académico,” Universidad Técnica del Norte, Honorable Consejo Universitario,

Ibarra, Ecuador, 2024. Accessed: Aug. 25, 2025. [Online]. Available: http://www.utn.edu.ec/

[67] U. T. del Norte, “Reglamento de Régimen Académico de la Universidad Técnica del Norte,” Universidad Técnica del Norte, Consejo de Educación Superior (CES), Ibarra, Ecuador, Feb. 2024. Accessed: Aug. 25, 2025. [Online]. Available: http://www.utn.edu.ec/

[68] U. T. del Norte, “Lineamiento Transitorio para el Cumplimiento del Requisito de Prácticas Preprofesionales y Vinculación con la Sociedad,” Universidad Técnica del Norte, Honorable Consejo Universitario, Ibarra, Ecuador, 2021. Accessed: Aug. 25, 2025. [Online]. Available: http://www.utn.edu.ec/

[69] U. T. del N. Dirección de Bienestar Universitario, “Proceso de Atención en Psicología,” Universidad Técnica del Norte, Ibarra, Ecuador, 2025. [Online]. Available: http://www.utn.edu.ec/

[70] U. T. del N. Dirección de Bienestar Universitario, “Proceso de Atención - Orientación Profesional,” Universidad Técnica del Norte, Ibarra, Ecuador, 2025. [Online]. Available: http://www.utn.edu.ec/

[71] M. J. Alegría Flores, “Funciones y Procesos del Servicio de Trabajo Social,” Universidad Técnica del Norte, Dirección de Bienestar Universitario, Ibarra, Ecuador, 2024. [Online]. Available: http://www.utn.edu.ec/

[72] U. T. del Norte, “Reglamento para Cursos de Actualización de Conocimientos en Carreras de Grado,” Universidad Técnica del Norte, Honorable Consejo Universitario, Ibarra, Ecuador, 2022. Accessed: Aug. 25, 2025. [Online]. Available: http://www.utn.edu.ec/

[73] U. T. del Norte, “Reglamento de Becas Reformado para Estudiantes de Carreras de Grado,” Universidad Técnica del Norte, Honorable Consejo Universitario, Ibarra, Ecuador, 2024. Accessed: Aug. 25, 2025. [Online]. Available: http://www.utn.edu.ec/

[74] A. Joshi, S. Kale, S. Chandel, and D. K. Pal, “Likert scale: Explored and explained,” *Br J Appl Sci Technol*, vol. 7, no. 4, p. 396, 2015.

[75] G. Bai *et al.*, “Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues,” *arXiv preprint arXiv:2402.14762*, 2024.

[76] L. Zheng *et al.*, “Judging llm-as-a-judge with mt-bench and chatbot arena,” *Adv Neural Inf Process Syst*, vol. 36, pp. 46595–46623, 2023.

[77] Q. Wu *et al.*, “Autogen: Enabling next-gen LLM applications via multi-agent conversations,” in *First Conference on Language Modeling*, 2024.

[78] W.-C. Kwan *et al.*, “Mt-eval: A multi-turn capabilities evaluation benchmark for large language models,” *arXiv preprint arXiv:2401.16745*, 2024.

[79] D. N. Palacio, A. Velasco, D. Rodriguez-Cardenas, K. Moran, and D. Poshyvanyk, “Evaluating and explaining large language models for code using syntactic structures,” *arXiv preprint arXiv:2308.03873*, 2023.

[80] J. Roberts, “How powerful are decoder-only transformer neural models?,” in *2024 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2024, pp. 1–8.

[81] J. Chen, H. Lin, X. Han, and L. Sun, “Benchmarking large language models in retrieval-augmented generation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024, pp. 17754–17762.

[82] O. Khattab *et al.*, “Dspy: Compiling declarative language model calls into self-improving pipelines,” *arXiv preprint arXiv:2310.03714*, 2023.

[83] S. Santurkar, E. Durmus, F. Ladhak, C. Lee, P. Liang, and T. Hashimoto, “Whose opinions do language models reflect?,” in *International Conference on Machine Learning*, PMLR, 2023, pp. 29971–30004.

[84] G. Bai *et al.*, “Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues,” *arXiv preprint arXiv:2402.14762*, 2024.

[85] A. Yehudai *et al.*, “Survey on evaluation of llm-based agents,” *arXiv preprint arXiv:2503.16416*, 2025.

[86] A. Wolters, A. F. Arz von Straussenburg, and D. M. Riehle, “Evaluation framework for large language model-based conversational agents,” 2024.

ANEXOS

Anexo A. Evidencias fotográficas de la aplicación del taller práctico y encuesta SUS.


BANCO
PICHINCHA

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE SOFTWARE
UTN Móvil

IBARRA - ECUADOR

INDICACIONES:

1	Complete la tabla con la hora de inicio y la hora de fin de cada tarea
3	Indique si la aplicación mostró algún error durante la tarea (Si/No)
4	Indique si completó la tarea con éxito (Si/No)
Nota:	Inicie cada objetivo desde la pantalla principal (Calendario de actividades)

Nivel de carrera:	

Nota: presione (Ctrl + h) para insertar la hora

Objetivo	Nro.	Descripción de la tarea	Inicio	Fin	Total	Errores	Finalizó	Observación
Escribir y recibir un saludo del chatbot	1	Escribir un saludo al chatbot universitario						
	2	Verificar que el chatbot haya respondido de manera cordial						
Charlar informalmente con el chatbot	3	Escribir una pregunta informal como: ¿Cómo te encuentras hoy?						
	4	Verificar que el chatbot responda a la pregunta de forma natural						
Plantear preguntas fuera de contexto al chatbot	5	Escribir un mensaje como: Casate conmigo						
	6	Verificar que el chatbot recomiende al usuario volver al contexto universitario para sus solicitudes						
Consultar al chatbot sobre información universitaria	7	Escrbir una pregunta al chatbot relacionada con algún proceso universitario adjunto al portafolio estudiantil, ejemplo: Quiero información sobre becas - Quiero información sobre prorrogas - Como justificar una falta						
	8	Verificar que el chatbot haya respondido satisfactoriamente la consulta realizada						
Exportar la conversación en pantalla	10	En la pantalla de conversación del chatbot, presione el botón con el símbolo de "PDF" para exportar la conversación en pantalla a este formato.						
Eliminar la conversación en pantalla	9	En la pantalla de conversación del chatbot, presione el botón con el símbolo de basurero para limpiar la conversación en pantalla, dejando unicamente el saludo inicial del chatbot.						

Total

Final	Una vez finalizado el taller guárdelo y complete la encuesta.

Anexo C. Tabulación de los datos obtenidos del taller práctico aplicado en la evaluación del chatbot

Id	Facultad	Carrera	Nivel	A1.1	A1.2	A2.1	A2.2	A3.1	A3.2	A4.1	A4.2	A5.1	A6.1	Errores	Tareas Logradas	Objetivos Logrados	Tiempo Total	Observaciones
1	FICA	Software	6	00:03.2	00:02.8	00:03.9	00:03.0	00:03.1	00:02.3	00:06.4	00:14.2	00:07.2	00:05.6	1	9	5	00:51.7	1
2	FICA	Software	6	00:02.9	00:03.0	00:04.1	00:01.2	00:03.2	00:03.6	00:05.9	00:13.6	00:07.0	00:05.0	0	10	6	00:49.5	0
3	FICA	Software	8	00:02.7	00:03.2	00:04.0	00:01.6	00:03.3	00:02.1	00:06.2	00:13.9	00:07.4	00:05.5	0	10	6	00:49.9	0
4	FICA	Software	7	00:00.0	00:00.0	00:00.0	00:00.0	00:00.0	00:00.0	00:00.0	00:00.0	00:00.0	00:00.0	10	0	0	00:00.0	0
5	FICA	Software	8	00:15.0	00:06.0	00:07.0	00:05.0	00:07.0	00:12.0	00:03.0	00:04.0	00:04.0	00:09.0	0	10	6	01:12.0	1
6	FICA	Software	8	00:11.0	00:04.0	00:03.7	00:05.4	00:03.0	00:06.7	00:03.0	00:04.5	00:04.3	00:09.2	0	10	6	00:54.8	0
7	FICA	Software	6	00:02.9	00:02.8	00:04.0	00:01.8	00:03.6	00:03.5	00:06.2	00:10.8	00:07.1	00:05.2	0	10	6	00:47.9	0

16	15	14	13	12	11	10	9	8
FICA	FICA	FICA	FICA	FICA	FICA	FICA	FICA	FICA
Software	Software	Software	Software	Software	Software	Software	Software	Software
8	8	7	8	6	7	6	7	6
00:20.1	00:02.9	00:02.9	00:02.9	00:09.2	00:02.9	00:04.0	00:27.0	00:40.4
00:10.3	00:03.0	00:02.9	00:12.2	00:02.1	00:02.7	00:02.6	00:03.3	00:18.5
00:40.1	00:04.0	00:04.4	00:04.2	00:14.2	00:04.4	00:03.2	00:05.2	00:27.3
00:10.2	00:01.5	00:02.7	00:12.2	00:02.1	00:01.9	00:05.7	00:08.3	00:20.6
00:10.8	00:03.0	00:03.1	00:03.1	00:09.1	00:03.7	00:04.5	00:05.2	00:50.0
00:40.1	00:03.6	00:02.4	00:03.3	00:02.0	00:01.7	00:03.1	00:04.3	00:19.0
00:40.4	00:06.4	00:06.0	00:06.5	00:14.1	00:06.4	00:05.1	00:50.2	00:25.0
01:00.3	00:10.2	00:14.2	00:14.1	00:10.0	00:11.4	00:05.0	00:18.4	00:40.0
00:50.3	00:06.8	00:06.7	00:07.0	00:12.1	00:07.5	00:06.2	00:09.3	00:32.2
00:20.7	00:05.1	01:00.1	00:05.6	00:04.0	00:05.3	00:02.1	00:08.4	00:29.4
0	0	0	0	0	0	0	0	4
10	10	10	10	10	10	10	10	6
6	6	6	6	6	6	6	6	3
05:03.3	00:46.5	01:45.4	01:11.1	01:18.9	00:47.9	00:41.5	02:19.6	05:02.4
0	2	0	0	0	0	0	0	1

25	24	23	22	21	20	19	18	17
FICA	FICA	FICA	FICA	FICA	FICA	FICA	FICA	FICA
Software	Software	Software	Software	Software	Software	Software	Software	Software
6	8	6	8	8	8	7	8	8
00:13.0	00:22.1	00:02.9	00:05.2	00:14.4	01:03.0	01:04.0	00:06.0	00:02.9
00:02.8	00:01.5	00:02.7	00:02.3	00:08.9	00:02.8	00:07.1	00:10.3	00:02.7
00:03.9	00:25.7	00:04.3	00:09.6	00:16.3	00:04.1	01:00.5	00:06.0	00:04.2
00:02.8	00:02.8	00:01.3	00:02.5	00:15.7	00:01.5	00:12.4	00:06.9	00:02.4
00:03.4	00:53.1	00:02.8	00:08.4	00:12.4	01:03.9	00:59.3	00:05.8	00:03.0
00:01.8	00:05.7	00:03.5	00:02.1	00:11.3	00:07.5	00:07.3	00:10.8	00:03.2
00:06.1	00:36.0	00:05.9	00:01.4	00:23.1	00:06.0	00:09.4	00:30.0	00:05.8
00:14.9	00:01.4	00:11.6	00:09.7	00:15.4	00:13.3	00:10.5	00:11.4	00:11.0
00:07.3	00:17.2	00:07.2	00:02.4	00:14.2	00:06.7	00:07.1	00:07.0	00:06.9
00:05.1	00:02.3	00:05.4	00:02.4	00:14.5	00:15.9	00:05.0	00:20.0	00:05.1
2	0	0	0	0	0	0	0	0
8	10	10	10	10	10	10	10	10
5	6	6	6	6	6	6	6	6
01:01.1	02:47.8	00:47.6	00:46.0	02:26.2	03:04.7	04:02.6	01:54.2	00:47.2
0	2	0	0	0	0	0	0	0

34	33	32	31	30	29	28	27	26
FICA	FICA	FICA	FICA	FICA	FICA	FICA	FICA	FICA
Software	Software	Software	Software	Software	Software	Software	Software	Software
7	7	7	7	8	7	7	8	8
00:00.0	00:01.5	00:02.9	00:02.9	00:02.9	00:02.9	00:02.9	00:03.3	00:15.4
00:00.0	00:01.6	00:02.3	00:02.8	00:02.8	00:02.9	00:02.5	00:01.2	00:08.4
00:00.0	00:02.3	00:04.0	00:03.9	00:04.4	00:04.4	00:03.7	00:02.4	00:15.6
00:00.0	00:05.7	00:03.0	00:01.4	00:01.3	00:02.1	00:01.7	00:04.8	00:15.1
00:00.0	00:47.7	00:03.5	00:03.0	00:02.7	00:03.4	00:03.0	00:03.0	00:10.5
00:00.0	00:06.5	00:03.1	00:03.1	01:02.3	00:01.6	00:03.6	00:03.0	00:10.7
00:00.0	00:09.4	00:06.5	00:06.6	02:00.3	00:06.3	00:06.3	00:03.2	00:23.4
00:00.0	00:01.8	00:14.4	00:13.6	00:13.1	00:14.8	00:12.6	00:01.3	00:15.2
00:00.0	00:13.4	02:00.3	00:07.2	00:07.2	00:06.8	00:06.8	00:06.7	00:28.1
00:00.0	00:01.9	00:05.5	00:04.8	00:05.2	00:05.1	00:05.2	00:05.0	00:20.3
10	0	0	0	0	0	0	0	0
0	10	10	10	10	10	10	10	10
0	6	6	6	6	6	6	6	6
00:00.0	01:31.8	02:45.5	00:49.3	03:42.2	00:50.3	00:48.3	00:33.9	02:42.7
1	0	0	0	0	0	0	2	0

43	42	41	40	39	38	37	36	35
FICA	FICA	FICA	FICA	FICA	FICA	FICA	FICA	FICA
Software	Software	Software	Software	Software	Software	Software	Software	Software
7	7	7	7	7	8	7	6	7
00:02.9	00:02.9	00:02.9	00:02.9	00:06.3	00:03.6	00:10.2	00:02.9	01:16.3
01:03.0	01:02.2	01:00.3	00:02.3	00:10.1	00:02.6	00:05.4	00:03.0	00:55.3
01:00.3	01:00.2	01:00.7	00:04.4	00:05.4	00:03.9	00:03.1	00:04.2	00:15.2
00:02.4	00:02.8	00:02.4	00:02.7	00:07.2	00:20.1	00:30.6	00:01.6	00:16.7
01:02.1	01:00.6	01:00.7	00:03.4	00:05.3	00:02.9	00:20.2	00:02.7	00:11.2
00:01.4	00:03.8	00:03.1	00:02.7	00:05.1	00:10.5	00:05.6	00:03.9	00:15.3
00:06.2	00:06.3	00:06.1	00:05.9	00:10.2	00:20.3	00:05.3	00:06.5	00:28.3
00:12.6	00:14.6	00:11.0	00:15.0	00:10.5	00:12.4	00:06.3	00:11.4	01:05.2
00:07.4	00:06.9	00:06.8	00:07.1	00:10.2	00:17.8	00:25.1	00:06.8	00:12.3
00:05.2	00:05.2	00:05.5	00:04.9	00:05.2	00:09.2	00:10.9	00:04.9	00:10.2
2	2	2	0	0	0	2	0	0
8	8	8	10	10	10	8	10	10
5	5	5	6	6	6	5	6	6
03:43.5	03:45.5	03:39.5	00:51.3	01:15.5	01:43.3	02:02.7	00:47.9	05:06.0
0	0	1	0	0	1	1	0	0

44	FICA	Software	7	00:02.9	00:02.5	00:04.3	00:03.0	00:03.4	00:01.2	00:06.3	00:11.7	00:07.4	00:05.0	0	10	6	00:47.7	0
45	FICA	Software	7	00:10.2	00:35.1	00:10.1	00:17.2	00:35.7	00:24.2	00:31.1	00:08.5	00:02.8	00:20.1	0	10	6	03:15.0	0
46	FICA	Software	8	00:25.1	00:13.2	00:07.1	00:14.6	00:10.1	00:18.3	00:12.1	00:20.4	00:50.5	00:10.2	0	10	6	03:01.6	0

Anexo D. Datos tabulados de la encuesta SUS aplicada

Id	Facultad	Carrera	Nivel	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	FICA	Software	6	1	3	1	1	3	1	4	2	3	2
2	FICA	Software	6	3	1	5	1	5	4	5	1	5	1
3	FICA	Software	8	5	1	5	1	5	1	5	1	5	1
4	FICA	Software	7	5	1	5	2	5	2	4	1	4	1
5	FICA	Software	8	3	2	3	2	3	2	3	2	3	2
6	FICA	Software	8	4	3	3	1	5	3	4	2	3	2
7	FICA	Software	6	4	2	3	1	5	3	4	2	4	2
8	FICA	Software	6	3	1	5	2	4	2	4	1	4	1

9	FICA	Software	7	3	1	5	1	5	1	5	1	5	2
10	FICA	Software	6	3	1	5	1	5	1	5	1	5	1
11	FICA	Software	7	5	2	4	1	5	2	4	1	4	2
12	FICA	Software	6	3	1	5	1	4	1	5	1	5	1
13	FICA	Software	8	3	3	4	3	3	3	3	4	1	3
14	FICA	Software	7	5	1	4	1	5	2	5	1	5	1
15	FICA	Software	8	4	1	4	1	4	1	1	1	4	1
16	FICA	Software	8	3	3	4	3	3	3	3	4	1	3
17	FICA	Software	8	3	1	4	1	5	1	5	1	5	1

18	FICA	Software	8	5	1	5	1	5	1	5	1	5	4
19	FICA	Software	7	4	1	5	1	5	1	5	1	5	1
20	FICA	Software	8	4	1	4	1	2	1	5	1	2	1
21	FICA	Software	8	5	1	5	1	5	1	5	1	5	1
22	FICA	Software	8	4	2	4	1	4	2	5	1	1	2
23	FICA	Software	6	1	1	3	2	4	2	5	1	5	1
24	FICA	Software	8	1	1	5	1	5	1	5	1	5	1
25	FICA	Software	6	2	3	4	1	5	3	5	1	4	2
26	FICA	Software	8	4	1	5	1	5	1	5	1	5	1

27	FICA	Software	8	3	1	5	1	5	1	5	1	1	1
28	FICA	Software	7	4	1	4	1	4	1	2	1	4	1
29	FICA	Software	7	4	1	4	1	4	1	2	1	4	1
30	FICA	Software	8	3	1	5	1	5	1	5	1	5	3
31	FICA	Software	7	5	1	5	1	5	1	5	1	5	1
32	FICA	Software	7	5	1	5	1	5	1	5	1	4	1
33	FICA	Software	7	5	1	5	1	5	1	5	1	5	1
34	FICA	Software	7	3	3	4	3	4	2	3	2	5	3
35	FICA	Software	7	4	1	5	1	5	2	5	1	3	1

36	FICA	Software	6	4	1	4	1	4	3	5	1	5	1
37	FICA	Software	7	2	2	4	1	4	4	4	1	3	1
38	FICA	Software	8	3	1	5	1	5	2	5	1	5	1
39	FICA	Software	7	3	1	2	1	4	2	3	2	3	3
40	FICA	Software	7	5	1	5	4	4	1	5	1	5	3
41	FICA	Software	7	2	1	5	2	5	1	5	1	5	1
42	FICA	Software	7	2	4	5	2	5	1	5	1	5	1
43	FICA	Software	7	2	4	5	2	5	5	5	1	5	1
44	FICA	Software	7	5	1	5	5	5	1	5	1	5	1

45	FICA	Software	7	4	1	5	1	5	1	4	1	4	1
46	FICA	Software	8	3	3	5	1	3	2	4	2	5	2

Anexo E. Certificado de entrega del trabajo de integración curricular al DDTI

UNIVERSIDAD TÉCNICA DEL NORTE

DIRECTOR DE LA DIRECCIÓN DE DESARROLLO TECNOLÓGICO E INFORMÁTICO

CERTIFICA

QUE: El señor Brayan Orlando de la Cruz Espinosa, con cédula identidad 0401751227 estudiante de la Carrera de Ingeniería de Software de la Facultad de Ingeniería en Ciencias Aplicadas, ha terminado su trabajo de titulación denominado ""Desarrollo de un chatbot para la gestión del portafolio estudiantil basado en un modelo de lenguaje de gran escala (LLM) utilizando SCRUM y la ISO/IEC 25022". Además, debo informar que se ha entregado la documentación técnica pertinente:

- Manual de Usuario
- Manual Técnico
- Diagramas y scripts de bases de datos
- Arquitectura y configuración de servidores
- Manual de Instalación y despliegue
- Plantilla de requisitos

Es todo cuanto puedo certificar, por consiguiente, el estudiante puede hacer uso de este certificado como evidencia de cumplimiento de la solicitud presentada previo la aprobación del tema de titulación.

Ibarra, 09 diciembre del 2025

CIENCIA Y TECNICA AL SERVICIO DEL PUEBLO

Firmado electrónicamente por:
JORGE ADRIAN CARAGUAY PROCEL
Validar únicamente con FirmaEC

Jorge Caraguay Procel

DIRECTOR