Universidad de las Ciencias Informáticas
centro de desarrollo, Facultad de Tecnologías Interactivas

Implementación de un sistema conversacional inteligente para el proceso de análisis de datos de la UJC del Cerro

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Pedro Alejandro Cabrera Enríquez

Tutores: M.Sc. Bienvenido Hanley Roque Orfe

Dr. C. Natalia Martínez Sánchez

La Habana, 2024

La informática se convertirá en una poderosísima fuerza científica, económica e incluso política del país Fidel Castro

Dedicatoria

Dedico este trabajo de diploma:

A mi madre Mayte , por su amor y apoyo incondicional, por ser un ejemplo de lucha y ser mi principal motivación para seguir adelante. ¡Por ser la mejor madre del mundo!

Agradecimientos

Primeramente, quiero expresar mi más sincero agradecimiento a mis padres, a mi abuela y a mi familia por su constante apoyo y amor incondicional durante todo el proceso de realización de esta tesis. Su paciencia, comprensión y aliento han sido fundamentales para que hoy pueda culminar esta etapa tan importante en mi formación académica. Agradezco profundamente a mis tutores por su invaluable orientación, apoyo y dedicación durante todo el proceso de realización de esta tesis. Su disposición para escucharme, su guía constante y sus sugerencias han enriquecido el presente trabajo. También Quiero agradecer a Julian por su apoyo como amigo y mentor con el cual no puedo más que sentirme afortunado por contar con él en los momentos más críticos de mi vida estudiantil.

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Pedro Alejandro Cabrera Enríquez
Autor

M.Sc. Bienvenido Hanley Roque Orfe
Tutor

Dr. C. Natalia Martínez Sánchez
Tutora

Resumen

La investigación se centra en el desarrollo de un sistema conversacional inteligente para el proceso de análisis de datos de la UJC del Cerro, con el objetivo de mejorar la accesibilidad y utilización de la información contenida en documentos no estructurados, como PDFs. La solución propuesta emplea técnicas avanzadas de procesamiento de lenguaje natural (PLN) y algoritmos de machine learning para transformar grandes volúmenes de datos en conocimiento procesable. Se seleccionó la metodología AUP-UCI para guiar el desarrollo del módulo, permitiendo una respuesta rápida a cambios y una entrega continua de valor.El sistema se encarga de ingresar los documentos generados en las actas de los comités de base de la UJC del Cerro, extraer su contenido y almacenarlo en un modelo de conocimiento estructurado. Las técnicas de tokenización, extracción de entidades nombradas y análisis de sentimientos se utilizan para procesar el contenido textual. Además, se aplican algoritmos de aprendizaje automático redes neuronales, para mejorar la búsqueda y respuesta semántica.Se definieron requisitos funcionales y no funcionales, así como historias de usuario, para guiar el desarrollo. La implementación de estándares de codificación, como los patrones GRASP y GOF, asegura un código legible y mantenible. Las pruebas funcionales y de rendimiento confirmaron la capacidad del sistema para manejar múltiples solicitudes simultáneamente lo que indica estabilidad y confiabilidad.

Palabras clave: Extracción de conocimiento , Procesamiento de lenguaje natural (PLN) , Machine learning
.

Índice general

Introducción **1**

1 FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS SOBRE EL OBJETO DE ESTUDIO. **4**

1.1 Marco Teórico de la Investigación: Conceptos Fundamentales Relacionados con la Investigación . . . 4

1.2 Estado del arte . . . 5

1.2.1 Modelos de lenguaje de gran tamaño . . . 7

1.2.2 Extracción de Documentos . . . 8

1.2.3 Técnicas de Procesamiento del Lenguaje Natural . . . 8

1.2.4 Técnicas de Minería de Datos . . . 9

1.2.5 Algoritmos de Aprendizaje Automático . . . 9

1.3 Herramientas y Tecnologías para el Desarrollo . . . 9

1.4 Metodología de Desarrollo de Software . . . 12

2 DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO **13**

2.1 Requisitos del software . . . 13

2.1.1 Requisitos funcionales . . . 13

2.1.2 Requisitos no funcionales . . . 14

2.2 Historias de usuario . . . 16

2.3 Arquitectura de software . . . 17

2.4 Patrones de diseño . . . 19

2.5 Modelo de Inteligencia Artificial . . . 22

2.6 Modelo de datos . . . 24

2.7 Implementación de la propuesta de solución . . . 25

2.7.1 Diagrama de despliegue . . . 25

2.7.2 Estándares de codificación . . . 26

3 VALIDACIÓN DE LA SOLUCIÓN PROPUESTA **29**

3.1 Validación de requisitos . . . 29

3.1.1 Estándares de codificación 29
3.1.2 Métricas aplicadas a los requisitos 30
3.2 Verificación de diseño 30
3.2.1 Relaciones entre Clases (RC) 30
3.2.2 Tamaño Operacional de Clases (TOC) 33
3.3 Pruebas de sofware 35
3.3.1 Pruebas unitarias 35
3.3.2 Pruebas funcionales 38
3.3.3 Pruebas de aceptación 41
3.3.4 Pruebas de rendimiento 41
3.4 Validación de las variables de investigación 43
3.5 Evaluación del modelo con Bleu 44

Conclusiones **46**

Recomendaciones **47**

Referencias bibliográficas **48**

Apéndices **52**
.1 Historias de usuario 53
.2 Aval del cliente 54

Índice de figuras

2.1 Representación de la arquitectura. (Fuente: Elaboración propia) 18
2.2 Ejemplo del patrón GRASP Experto. (Fuente: Elaboración propia) 20
2.3 Ejemplo del patrón GRASP Experto. (Fuente: Elaboración propia) 21
2.4 Ejemplo del patrón GOF Singleton. (Fuente: Elaboración propia) 22
2.5 Modelo de inteligencia Artificial.. (Fuente: Elaboración propia) 23
2.6 Modelo de datos de la solución propuesta (Fuente: Elaboración propia) 25
2.7 Diagrama de despliegue (Fuente: Elaboración propia) 26
2.8 Ejemplo del código que cumple con los estándares según la guía de python(Fuente: Elaboración propia) 27
2.9 Ejemplo del código que cumple con los estándares según la guía de python (Fuente: Elaboración propia) 27
2.10 Ejemplo del código que cumple con los estándares según la guía de python (Fuente: Elaboración propia) 27

3.1 Representación de la cantidad de clases por cantidad de dependencias. (Fuente: Elaboración propia) 32
3.2 Representación en porciento del nivel de acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización de las clases . (Fuente: Elaboración propia) 32
3.3 Representación de la cantidad de clases por cantidad de procedimientos que contienen. (Fuente: Elaboración propia) 34
3.4 Representación en porciento del nivel de responsabilidad, complejidad de implementación y reutilización de las clases. (Fuente: Elaboración propia) 34
3.5 Código de la función Listar documentos del RF No.6: Listar documentos. (Fuente: Elaboración propia) 36
3.6 Grafo perteneciente al código de la función Listar documentos No.6: Listar documentos (Fuente: Elaboración propia) 36
3.7 Formas para calcular la complejidad ciclomática (Fuente: Elaboración propia) 37
3.8 Resultados de la complejidad ciclomática (Fuente: Elaboración propia) 37
3.9 Gráfica con las no conformidades (NC) por iteración (Fuente: Elaboración propia) 40

3.10 Gráfica con las no conformidades (NC) por clasificación en cada iteración(Fuente: Elaboración propia) 40
11 Acta de aceptación 54

Índice de tablas

1.1 Comparación de sistemas similares Fuente: Elaboración propia. 6
1.2 Comparación de LLMs Fuente: Elaboración propia. 8

2.1 Descripción de los Requisitos Funcionales. (Fuente: Elaboración propia). 14
2.2 Historia de usuario # 1 16

3.1 Métrica RC. Categoría por atributos y criterio de evaluación (Fuente: Elaboración Propia). . 31
3.2 Valores de los umbrales para la métrica TOC (Fuente: Elaboración Propia). 33
3.3 Diseño de caso de prueba para el camino 2 38
3.4 Comparación de sistemas similares Fuente: Elaboración propia. 42
3.5 Comparación de sistemas similares Fuente: Elaboración propia. 44
3.6 Comparación de sistemas similares Fuente: Elaboración propia. 45
7 Historia de usuario # 2 53
8 Historia de usuario # 3 53

Introducción

Con el incremento de las tecnologías y las investigaciones durante estos últimos años se ha dado pie para el desarrollo de este tipo de herramientas, siendo una temática recurrente la implementación de softwares que permitan mantener una conversación coherente con el usuario en su lenguaje natural a través de técnicas de Procesamiento de lenguaje natural (NLP, por sus siglas en inglés) (Kang; Cai; Tan; Huang y H. Liu, 2020), las cuales extraen características propias del lenguaje, entendiendo los mensajes, y dando una respuesta con base en las fuentes de información disponibles. El número de Sistemas conversacionales inteligentes (SCI) en su modelo de negocio no para de crecer.

En el contexto nacional, Cuba no ha quedado ajena a estas transformaciones. Aunque persisten desafíos en cuanto a infraestructura y acceso a tecnologías, se han implementado iniciativas para integrar herramientas de SCI en las organizaciones. Una de las plataformas que utiliza inteligencia artificial, la cual facilita la enseñanza y el aprendizaje en universidades y centros de estudio. Sin embargo, el uso de tecnologías avanzadas, como los Agentes conversacionales (AC) , sigue siendo limitado, a pesar de que su integración podría mejorar la interacción y la accesibilidad a la información en estos entornos educativos. La Universidad de las Ciencias Informáticas (UCI), como líder en el desarrollo de soluciones tecnológicas educativas en Cuba, tiene la oportunidad de adoptar innovaciones que fortalezcan su plataforma de enseñanza online y contribuyan a superar los retos actuales en la educación.

La Unión de Jóvenes Comunistas (UJC) constituida el 4 de abril de 1962, es la organización política de la juventud cubana, creada con el fin de asegurar la unidad de los jóvenes cubanos, movilizarlos en torno a la Revolución Socialista y contribuir a su educación política. El ingreso es de carácter voluntario y selectivo. En ella militan jóvenes que estén entre los 16 y 32 años de edad. Su trabajo se proyecta más allá de sus miembros, está dirigida a toda la población joven del país. Su principal objetivo es la formación integral de cada joven.

El municipio del Cerro agrupa a la juventud de vanguardia de la sociedad cubana y cuenta con un porcentaje elevado de miembros entre estudiantes, profesores y trabajadores. Desde el propio surgimiento de la institución, las estructuras de la UJC han necesitado utilizar las potencialidades que las tecnologías de la información y las comunicaciones ofrecen, los documentos de actas ordinarias son las que proceden de las reuniones mensuales de cada Comité de Base (CB), al igual en los Comité Primarios (CP) y Comité UJC (CUJC) donde se generan de manera manual los reportes que constituyen el listado de acuerdos tomados en cada una de las reuniones realizadas, como son, reporte sobre sanciones, rendiciones de cuenta, listado de ausentes, de presentes, por ciento de asistencia, la participación de estructuras superiores. Independiente-

mente de la realización de las actas, la calidad de las mismas se ve afectada debido a que los secretarios en muchas ocasiones no cuentan con los modelos adecuados para el tipo de reunión que están realizando, toda esa documentación debe perdurar en el tiempo debido que en caso crítico es necesario revisar más de 200 archivos mensualmente. Esta descentralización de la información provoca que exista más de un responsable de la información y que se archive en diferentes lugares, impidiendo el acceso a la misma y retrasando así los análisis estadísticos, que de manera muy periódica se ven afectados debido a la gran cantidad de documentos, el número de actividades y el intercambio entre los miembros de la organización que conviven en el municipio.
La UJC del municipio Cerro se enfrenta al desafío de gestionar un gran volumen de información proveniente de los comités de base, con el objetivo de comprender mejor las necesidades, intereses y tendencias de sus miembros. La información actual se encuentra dispersa en diferentes formatos y ubicaciones, lo que dificulta su análisis y utilización efectiva para la toma de decisiones estratégicas.
Por lo que se tiene como deficiencia: La falta de un sistema centralizado para el análisis de datos de los comités de base impide a la UJC obtener una visión completa y actualizada de las tendencias que afectan a la organización. Esto limita la capacidad de la UJC para:

- Identificar oportunidades y desafíos: La UJC no puede identificar de manera eficiente las áreas donde se requieren acciones o programas específicos para atender las necesidades de sus miembros.
- Tomar decisiones informadas: La toma de decisiones se basa en información incompleta o desactualizada, lo que puede llevar a decisiones ineficaces o incluso perjudiciales para la organización.
- Optimizar recursos: La UJC no puede asignar sus recursos de manera eficiente para maximizar su impacto en la comunidad.
- Fortalecer la comunicación: La UJC no puede comunicar de manera efectiva sus mensajes y programas a sus miembros, lo que genera una falta de participación y compromiso.

Por lo antes planteado se propone como pregunta científica: ¿Como contribuir al análisis de tendencias planteadas en las reuniones del comité de base en el municipio Cerro? Por lo que se realiza el análisis en el objeto de estudio: sistema conversacional inteligente para el análisis de tendencias para la UJC, enmarcado en el campo de acción: la gestión de la información y la toma de decisiones en la UJC del municipio Cerro
Objetivo general: Implementación de un sistema conversacional inteligente para el proceso de análisis de datos de la UJC del municipio Cerro, con el fin de mejorar la toma de decisiones y fortalecer el trabajo de la organización a nivel municipal.
Objetivos específicos:

1. Analizar de las tendencias actuales de los portales web para el análisis de los datos.
2. Seleccionar de las tendencias analizadas cual se acopla mejor a las necesidades del proyecto.
3. Implementación del sistema conversacional inteligente para la información de los comités de bases a través de la minería de texto.
4. Validación del sistema conversacional inteligente para la información de los comités de bases.

Preguntas de la investigación:

1. ¿Cómo puede la UJC desarrollar un sistema conversacional inteligente para procesar y analizar de manera eficiente la información de los comités de base?
2. ¿Qué técnicas de minería de texto y análisis de datos son más adecuadas para extraer información relevante y útil de la información de los comités de base?
3. ¿Cómo puede visualizarse la información procesada por el sistema conversacional inteligente para facilitar su comprensión y uso por parte de los tomadores de decisiones de la UJC?
4. ¿Qué estrategias de implementación y capacitación se requieren para asegurar que el sistema conversacional inteligente sea utilizado de manera efectiva por la UJC?

Como parte del desarrollo de esta investigación se utilizaron un conjunto de métodos científicos investigativos como parte de estos métodos se encuentran: Los métodos teóricos:

- Histórico-lógico: Determina las tendencias actuales de los sistemas conversacionales inteligentes, los modelos grandes de lenguaje, lenguajes de programación y herramientas a utilizar en la investigación.
- Analítico-Sintético: Se aplicó este diseño en base al análisis e investigación de los diferentes enfoques y tecnologías utilizadas en el desarrollo del sistema conversacional inteligente.
- Inductivo-Deductivo: Se emplea principalmente para la elaboración del sistema conversacional inteligente, para configurar las intenciones y respuestas de dicho sistema en base a las reglas establecidas.

Entre los métodos empíricos utilizado se encuentra:

- La Entrevista: Al usuario y al tutor con el objetivo de obtener información valiosa y comprender sus necesidades como base para el desarrollo del sistema conversacional inteligente.

El trabajo de diploma se encuentra estructurado en tres capítulos:

- Capítulo 1: Fundamentos de la investigación En este capítulo se realiza un estudio del estado del arte del tema en investigación haciendo referencia a los principales elementos teóricos en los cuales está basado. Se exponen las herramientas, tecnologías, lenguajes y metodología utilizada para el desarrollo de la solución y los conceptos relacionados con el contenido.
- Capítulo 2: Diseño de la solución propuesta al problema científico En este capítulo se describe la solución propuesta ante el problema de investigación existente, detallándose la utilización de cuatro de siete etapas empleadas de la minería de datos. Se describe el proceso de implementación y se muestran los resultados obtenidos.
- Capítulo 3: Evaluación de la solución propuesta En este capítulo se verifica el resultado de la implementación de la solución. Se describen las pruebas que se realizaron al modelo y los resultados obtenidos de dichas pruebas con el objetivo de obtener un producto que cumpla con los requerimientos establecidos.

CAPÍTULO 1

FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS SOBRE EL OBJETO DE ESTUDIO.

En el presente capítulo se tiene como principal objetivo la comprensión del objeto de estudio de la investigación. Para ello, se realiza una definición de los conceptos básicos necesarios para el entendimiento de las terminologías estrechamente relacionadas con el tema principal y dominio del problema a resolver, así como un estudio de las soluciones similares a la propuesta de solución existentes, ya sea en el marco nacional como el internacional. Se determinan, además las herramientas, tecnologías y metodología a emplear en el desarrollo de la propuesta de solución ofrecida.

1.1. Marco Teórico de la Investigación: Conceptos Fundamentales Relacionados con la Investigación

Chatbot: Un Chatbot es un programa de computadora diseñado para mantener una comunicación con un usuario humano. La comunicación del Chatbot se basa en reglas y en el uso de inteligencia artificial para poder establecer una comunicación en lenguaje natural con una persona como si se tratase de una conversación entre dos humanos. Actualmente, los Chatbots son sumamente utilizados en lo referente a la comunicación entre los sistemas informáticos y sus usuarios, debido a su gran capacidad de interpretar el lenguaje natural y transformarlo en acciones que pueden realizar diversas tareas como la generación de un reporte, la atención de preguntas frecuentes y generación de pagos de servicios (Adamopoulou y Moussiades, 2020).

Asistente virtual: Un asistente virtual es una tecnología creada con inteligencia artificial para apoyar o sustituir en tareas específicas de expertos además de automatizarlas. Un asistente virtual posee la capacidad de percibir un entorno, tener una representación parcial del mismo y de actuar sobre él. Además, un asistente virtual puede comunicarse con otros agentes, humanos o no, que comparten su hábitat. Dicha tecnología tiene un conjunto de objetivos que gobiernan su comportamiento y posee recursos propios (Medina; Cabeza y Peña, 2013).

Inteligencia Artificial: La inteligencia Artificial, es una rama del conocimiento de naturaleza multidiscipli-

nar, involucra campos como las ciencias de la computación y de la información, la lógica, la matemática, la estadística y la lingüística. En sinergia con tecnologías avanzadas, busca que los equipos informáticos y distintos dispositivos tecnológicos, realicen tareas que normalmente requerirían inteligencia humana, como por ejemplo las capacidades de aprender, razonar, resolver problemas, la percepción visual, el reconocimiento de voz, la toma decisiones y la traducción de idiomas. En conclusión, la Inteligencia Artificial (IA) involucra los sistemas que tienen la capacidad de emular el comportamiento humano inteligente (Rouhiainen, 2018).
Procesamiento de lenguaje natural: El NLP es una parte fundamental para la construcción de los chatbots. Es un campo de la inteligencia artificial que se encarga de las interacciones entre las máquinas y los humanos. El NLP se utiliza para el reconocimiento de voz o texto permitiendo así que una computadora sea capaz de analizar y procesar el lenguaje humano. El procesamiento del lenguaje está considerado uno de los problemas más difíciles de la informática. El lenguaje natural raramente es preciso o sigue reglas determinadas. No solo basta con entender las palabras, sino que hace falta entender también el contexto. Si bien los seres humanos pueden aprender fácilmente un idioma, la ambigüedad del lenguaje y el no tener reglas precisas para todos los casos son lo que hacen que el NLP sea difícil de aplicar para las máquinas (Gelbukh, 2010).
Large Lengual Model: Larger Lengual Model (LLM, por sus siglas en inglés) o Modelos de Lenguaje de gran tamaño son avanzados sistemas de inteligencia artificial diseñados para comprender y generar texto de manera similar a la humana. Utilizan Redes Neuronales Profundas (RNP) y están entrenados en vasta cantidad de datos textuales, lo que les permite reconocer patrones y contextos lingüísticos. Durante el entrenamiento, los LLMs ajustan millones o incluso billones de parámetros para mejorar su precisión y capacidad de predicción. Este proceso les permite desarrollar un entendimiento detallado de la sintaxis, semántica y contexto de las palabras y frases (Greenberg; Lengua; Coie; Pinderhughes; Bierman; Dodge; Lochman y McMahon, 1999).

1.2. Estado del arte

En la actualidad con el avance y el despliegue total de las TIC es frecuente encontrar soluciones similares a una propuesta de solución para resolver una problemática determinada. Dichas soluciones pueden ser básicamente otras aplicaciones que junten características comunes y que muestren el camino a seguir en el desarrollo de la solución problemática (Katz, 2009). A continuación, se realiza un estudio de los componentes sistemas comunicacionales inteligentes.
ChatPDF: Es una herramienta que mejora la experiencia de lectura de documentos PDF. Permite a los usuarios cargar PDFs y obtener resúmenes instantáneos, así como hacer preguntas sobre el contenido del documento. Facilita la comprensión rápida y eficiente de tex-tos complejos. Ofrece funciones multilingües, permitiendo interactuar con documentos en diferentes idiomas. También proporciona resúmenes, traducciones y explicaciones de gráficos y tablas haciendo que la navegación a través de documentos técnicos sea menos rigurosa sin dejar de garantizar la seguridad de los datos del usuario (Panda, 2023).
Chatize: es un asistente de documentos impulsado por IA que utiliza ChatGPT para analizar y resumir

rápidamente datos en varios formatos de documentos, como PDF, DOC, DOCX, PPT, entre otros. Permite al usuario carga documentos y hacer preguntas sobre el contenido, obteniendo respuestas instantáneas y relevantes (N. W. Kim; Ko; G. Myers y Bach, 2024).

Answer Bot: Es un producto chatbot de Zendesk, el cual es un software de soporte, ventas e interacción con el cliente flexible y escalable. Answer Bot, utiliza el aprendizaje automático para hacer búsquedas en la base de conocimientos y encontrar respuestas. Cada respuesta responde a una pregunta o problema particular del cliente. De no tener una respuesta con-figurada ante la inquietud de un cliente, Answer Bot, simplemente recurre a la búsqueda de recomendaciones de artículos que puedan servir de ayuda (Tian; Thung; Sharma y Lo, 2017).

CLaroBot: Claro es una de las empresas líderes en servicios integrados de telecomunicaciones en América Latina. ClaroBot es un software de Inteligencia artificial de tipo chatbot, utilizado por la empresa, que opera los 7 días de la semana y las 24 horas del día, simulando una conversación para resolver consultas sobre productos y servicios o brin-dar una respuesta inmediata a preguntas frecuentes de sus clientes o colaboradores. Entre sus principales beneficios se evidencian: la posibilidad de integración con otras aplicaciones de la empresa para responder a consultas más complejas, la derivación de consultas a agentes especializados, la atención de múltiples solicitudes a la vez y la generación reportes sobre las consultas recibidas, preguntas resueltas, horarios de mayor conexión (Khaddad; Robert; Gross-goupil; Deslandes; Capon; Alezra; Blanc; Bernhard y Bladou, 2023). TelcelBot Telcel es la empresa líder de teleco.

Valoración de las soluciones.

A continuación, se muestra una tabla resumen de las principales características que se tomaron en cuenta para el estudio de las soluciones similares. Las características a evaluar son:

- Conectividad: Se realiza un análisis de la necesidad de conexión a Internet para el acceso al sistema conversacional inteligente.
- Código accesible: Se realiza un análisis de la posibilidad acceso al código fuente.
- Disponibilidad: Se realiza un análisis de la disponibilidad para los usuarios actualmente.
- Facil de usar:No hacen falta conocimientos previos para utilizarlos.

Tabla 1.1. Comparación de sistemas similares Fuente: Elaboración propia.

Sistemas	Conectividad	Codigo abierto	Disponibilidad	Fácil de usar
ChatPDF	Si	No	Si	Si
Chatize	Si	No	Si	Si
Answer Bot	Si	Si	No	3
CLaroBot	Si	No	SI	Si

A modo general, estas aplicaciones informáticas son de uso liberado para los usuarios, pero no ocurre así con el código fuente de las mismas, por lo que no pueden ser reutilizadas y adaptadas al problema existente.

Ya que ninguna de las soluciones estudiadas anteriormente puede ser adaptada al problema a resolver se decide la implementación de un componente que dé solución a las necesidades del cliente.

1.2.1. Modelos de lenguaje de gran tamaño

Los LLM actualmente demuestran que se puede tener un lenguaje muy fluido sin una comprensión genuina. Un LLM pueden facilitar labores de traducción, de revisión gramatical, de corrección de código computacional, de transformación de discurso verbal en texto y viceversa, y de reproducir diferentes tonos de escritura. También pueden ser usados para hacer exploraciones iniciales sobre temas o para estimular la inspiración (Wu; Fei; Qu; Ji y Chua, 2023).

Los modelos estudiados para utilizar en el sistema conversacional inteligente para la UJC del cerro Fueron:

GPT-4: Es la cuarta generación del modelo de lenguaje desarrollado por OpenAI, conocido por su capacidad de transformadores, GPT-4 ha sido entrenado en un conjunto de datos masivos y diversos que abarca una amplia gama de temas y estilos de escritura. Esto le permite generar respuestas contextualmente relevantes y de alta calidad en múltiples idiomas. Una de las mejoras clave de GPT-4 sobre sus predecesores es su capacidad para comprender y manejar entradas más complejas, lo que lo hace útil para aplicaciones más avanzadas como la redacción creativa, la generación de código, y la asistencia en investigaciones académicas (Achiam et al., 2023).

Falcon: Representa una notable mejora en el campo de la inteligencia artificial y el procesamiento de lenguaje natural. Diseñado por TII en Abu Dhabi, estos modelos han sido entrenados en grandes conjuntos de datos multilingüe que incluye texto de diversos idiomas, lo que les permite entender y generar el texto de manera precisa y coherente. Falcon LLM es conocido por su capacidad de manejar tareas complejas como la generación de texto, la traducción, la respuesta a preguntas y análisis de sentimientos, entre otras aplicaciones (Vázquez; Ricardo y Vega-Falcón, 2022).

Llama 3: Es la última versión del modelo de lenguaje grande desarrollado por Meta (anteriormente Facebook). Este modelo conocido como Large Language Model Meta AI (LLAMA, por sus siglas en inglés) ha sido diseñado para mejorar la comprensión y generación de texto en múltiples idiomas. Incorpora la Atención Agrupada de consultas (GQA), lo que permite manejar contextos más largos de manera eficiente. Incluye Llama Guard 2, una versión ajustada para seguridad que clasifica las entradas y respuestas del modelo para detectar contenido inapropiado (Dubey et al., 2024).

Bert: Es un modelo de lenguaje grande desarrollado por Google en 2018.Utiliza la arquitectura de transformadores y se entrena mediante aprendizaje auto-supervisado, lo que le permite aprender representaciones contextuales de tokens en su contexto. Se destaca por su capacidad para mejorar significativamente el rendimiento en diversas tareas de procesamiento de lenguaje natural, como la comprensión del lenguaje general, la respuesta a preguntas y el reconocimiento de entidades nombradas (Z.-W. Liu; Röpke y Han, 2023).

Tabla 1.2. Comparación de LLMs Fuente: Elaboración propia.

LLM	Disponibilidad	Escalabilidad	Ejecutar de forma local	Integración	Baja demanda computacional
Bert	Si	No	Si	No	No
LLaMA 3	Si	Si	Si	Si	No
GPT-4	Si	Si	No	Si	-
qwen2.5-coder:0.5b	Si	Si	Si	Si	Si
Falcon	Si	Si	No	Si	-

Después de esta comparativa se llegó a la conclusión de que qwen2.5-coder:0.5b es altamente eficiente en términos de recursos computacionales lo que se traduce en que se pueda ejecutar en dispositivos con capacidades más limitadas sin sacrificar la capacidad del rendimiento. Esto es particularmente útil en organizaciones y proyectos que necesitan soluciones poderosas, pero accesibles.

1.2.2. Extracción de Documentos

Una vez recopilados los enlaces a los documentos o descargados los archivos, es necesario procesarlos para extraer el contenido textual. Las técnicas y herramientas para la extracción de documentos incluyen:

- PDFMiner: Una herramienta en Python que permite extraer texto y metadatos de archivos PDF. PDFMiner es útil para analizar la estructura de los documentos PDF y extraer información textual de manera precisa (Jiang e Y. Li, 2023).
- PyPDF2: Biblioteca en Python que facilita la lectura y extracción de texto de archivos PDF. Permite dividir, fusionar y manipular archivos PDF, lo que facilita la obtención de contenido textual para su análisis posterior (Suresh et al., 2023).
- python-docx: Biblioteca que permite leer y escribir archivos .docx (Microsoft Word). Facilita la extracción del texto contenido en documentos de Word, permitiendo el acceso a información estructurada y bien organizada (Zhao; Wang y Chai, 2023).
- Lectura de Archivos .txt: Utilizar funciones básicas en Python (por ejemplo, open()) para leer el texto directamente desde archivos .txt. Este enfoque es útil para documentos que ya están en formato de texto plano, facilitando una extracción directa y eficiente del contenido.

1.2.3. Técnicas de Procesamiento del Lenguaje Natural

- Tokenización: Consiste en dividir el texto extraído en unidades más pequeñas, como palabras o frases, para su análisis posterior. La tokenización facilita el procesamiento del texto y la aplicación de técnicas de análisis más avanzadas (Calvo, 2023).

- Extracción de Entidades Nombradas: Identificar y clasificar elementos clave dentro del texto, como nombres de personas, organizaciones y lugares. Esta técnica permite estructurar la información de manera que se pueda acceder fácilmente a datos relevantes y organizar el conocimiento de manera significativa (Diaz Lupone y Tafka Ghazal, 2024).
- Análisis de Sentimientos: Determinar la actitud expresada en un texto, clasificándola como positiva, negativa o neutral. El análisis de sentimientos es útil para comprender la percepción de los usuarios y el contexto emocional en el contenido textual (Alaminos-Fernandez, 2023).

1.2.4. Técnicas de Minería de Datos

- Clasificación: Asignar categorías predefinidas a los datos extraídos. La clasificación facilita la organización de la información y la identificación de patrones en el contenido educativo, permitiendo una mejor organización y acceso a la información relevante .
- Agrupamiento (Clustering): Agrupar documentos similares basándose en características comunes. El agrupamiento permite identificar patrones y relaciones en el contenido educativo, facilitando la organización de grandes volúmenes de datos en categorías coherentes y útiles (Oyewole y Thopil, 2023).

1.2.5. Algoritmos de Aprendizaje Automático

Para tareas complejas de análisis y procesamiento de datos textuales, se emplean varios algoritmos de aprendizaje automático:

- Redes Neuronales: Utilizadas para tareas de clasificación y reconocimiento de patrones en grandes volúmenes de datos textuales. Las redes neuronales profundas pueden identificar patrones complejos y realizar análisis detallados de los datos extraídos (Martin; Saez-Delgado y Lepe-Martinez, 2023).
- Árboles de Decisión: Algoritmo que toma decisiones basadas en preguntas sobre las características del texto. Los árboles de decisión permiten estructurar la información y realizar predicciones basadas en criterios específicos, facilitando la toma de decisiones informadas (Arda Suarez; Niu e Ivan-Baragano, 2023).

1.3. Herramientas y Tecnologías para el Desarrollo

Para el desarrollo del sistema conversacional inteligente para el análisis de tendencias en los comités de bases de la UJC del municipio Cerro, se utilizará una combinación de lenguajes de programación, herramientas CASE, IDEs, frameworks, librerías y gestores de bases de datos. A continuación, se presenta un análisis detallado de cada uno de estos elementos, junto con las razones para su selección.
Modelo de lenguaje de gran tamaño
qwen2.5-coder:0.5b

qwen2.5-coder en su versión 0.5 (500 millones de parametros) fue seleccionada como mo-delo de lenguaje dado que esta versión se diseña para ser eficiente en términos de recursos computacionales, permitiendo su implementación en una variedad de dispositivos y aplica-ciones con menor capacidad de procesamiento, sin sacrificar la calidad de las respuestas.La inclusión de Grouped Query Attention (GQA) permite manejar contextos mas largos para aplicaciones que necesitan procesar gandes cantidades de información en tiempo real .Ademas integra avances en la mitigación de sesgos y seguridad, reduciendo la probabilidad de generar respuestas inapropiadas (Hui et al., 2024).

Lenguajes de Programación

Python:

Python ha sido seleccionado como el lenguaje principal para el desarrollo del sistema debido a su simplicidad y versatilidad. Su amplia gama de bibliotecas especializadas en procesamiento de datos, web scraping y aprendizaje automático lo convierte en una opción ideal para el desarrollo del módulo. La sintaxis clara y legible de Python facilita la colaboración entre desarrolladores y permite una implementación rápida de prototipos. Además, su comunidad activa y el soporte continuo de nuevas bibliotecas y herramientas refuerzan su posición como el lenguaje preferido para proyectos de este tipo (Pinargote-Zambrano; Lino-Calle; Vera-Almeida et al., 2024).

Herramientas Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés)

Visual Paradigm v8.0

Es una herramienta profesional de CASE que soporta el desarrollo de software desde el análisis y diseño orientado a objeto, construcción, pruebas y despliegue. Ayuda a los equipos de desarrollo de softwares a capturar los requisitos correctos y transformarlos en diseños precisos, lo que ayuda a los desarrolladores a crear el software adecuado según los requisitos. Además, permite la importación y exportación de archivos de Lenguaje de Marcado Extensible (XML, por sus siglas en inglés) de diferentes versiones. Presenta licencia gratuita y comercial. Es multiplataforma, fácil de instalar, actualizar y compatible entre ediciones. Entre sus principales características se encuentra (Gaines, y otros):

- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Diagramas de flujo de datos.

Se ha optado por utilizar la versión 8.0 de la herramienta para la elaboración de los artefactos, ya que ofrece numerosas ventajas como la capacidad de una gran cantidad de diagramas de clases, código inverso, generar código a partir de los diagramas y generar documentación. Además, el equipo de desarrollo cuenta con experiencia en su empleo, lo que permite acelerar el proceso de desarrollo del software al no ser necesario capacitar a los desarrolladores.

Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés)

Visual Studio Code (VSC, por sus siglas en inglés): Visual Studio Code ha sido elegido como el IDE principal debido a su flexibilidad y soporte para múltiples lengua-jes de programación. VSC ofrece características avanzadas como la depuración, la inte-gración con sistemas de control de versiones como Git, y una amplia gama de extensiones que facilitan el desarrollo en Python y otros lenguajes utilizados en el proyecto. Su capaci-dad para adaptarse a diferentes necesidades de desarrollo y su soporte para una variedad de herramientas de desarrollo hacen de VSC una opción robusta y eficiente (M. Zhang; Yuan; H. Li y Xu, 2024).

Frameworks

Flask:

Flask será utilizado para el desarrollo del backend del sistema debido a su ligereza y flexibilidad como microframework para Python. Flask permite la creación de aplicaciones web de manera rápida y es altamente extensible, facilitando la integración con otras bibliotecas y herramientas necesarias para el procesamiento de datos. Su simplicidad y el control que ofrece sobre los componentes del servidor permiten una personalización efectiva y un desarrollo ágil del módulo (Ye; D. Kim; Sungdong Kim; Hwang; Seungone Kim; Jo; Thorne; J. Kim y Seo, 2023).

Librerías

Beautiful Soup: Beautiful Soup se empleará para realizar web scraping, permitiendo la extracción de enlaces a documentos desde páginas web. Su capacidad para nave-gar por estructuras HTML la convierte en una herramienta valiosa para recopilar in-formación inicial de manera efectiva. La facilidad con la que se pueden buscar y ex-traer datos específicos hace de Beautiful Soup una opción adecuada para la recopi-lación de datos no estructurados (Anbu; Miriam y Robin, 2024).

Pandas: Pandas se utilizará para la manipulación y análisis de los datos extraídos. Su capaci-dad para manejar estructuras tabulares facilita la organización y el procesamiento de los datos antes de almacenarlos en la base semántica. La versatilidad de Pandas en la limpieza, transformación y análisis de datos es crucial para preparar la información de manera que sea útil y accesible para los usuarios del sistema (Reback et al., 2020) .

Transformers (de Hugging Face): La librería Transformers proporcionará acceso a modelos preentrenados como BERT, que permiten obtener embeddings semánticos a partir del texto extraído. Estos embeddings facilitan la construcción de representacio-nes vectoriales que optimizan la búsqueda y recuperación eficiente en la base de da-tos. La capacidad de trabajar con modelos avanzados de lenguaje natural mejora sig-nificativamente la precisión y relevancia de las consultas realizadas en el sistema (Jain, 2022).

LangChain: Es una solución innovadora que aborda uno de los desafíos más importantes en el desarrollo de chatbots y aplicaciones conversacionales basadas en LLM: la gestión de diálogos. Aunque hay consideraciones que tener en cuenta al utilizar LangChain, su enfoque de código abierto y sus funcionalidades lo convierten en una herramienta valiosa para los desarrolladores. Al final, la capacidad de combinar LLM con otras fuentes de computación y conocimiento es lo que permite crear aplicaciones verdaderamente potentes y transformadoras (Topsakal y Akinci, 2023) .

Gestores de Bases de Datos

PostgreSQL: es un Sistema de Gestión de Bases de Datos Objeto Relacional (ORDBMS) de código abierto. Es conocido por su robustez, extensibilidad y cumplimiento con los estánda-res SQL. PostgreSQL soporta una amplia variedad de tipos de datos, incluyendo JSON y XML, lo que lo hace ideal para aplicaciones modernas que requieren flexibilidad en el ma-nejo de datos (Quest Software 2024).
Una de las características destacadas de PostgreSQL es su capacidad para manejar grandes volúmenes de datos y realizar consultas complejas de manera eficiente. Además, ofrece so-porte para transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), lo que garantiza la integridad y confiabilidad de los datos. PostgreSQL también incluye potentes funciones de búsqueda de texto completo, lo que permite realizar búsquedas avanzadas y rápidas dentro de grandes conjuntos de datos tex-tuales. Su arquitectura extensible permite a los desarrolladores añadir nuevas funcionali-dades mediante extensiones, como PostGIS para datos geoespaciales (Viloria; Acuña; Franco; Hernández-Palma; Fuentes y Rambal, 2019).

1.4. Metodología de Desarrollo de Software

AUP-UCI: es una Proceso Unificado Ágil (AUP) desarrollado por la UCI. Sus características incluyen:

- Fases bien definidas: Comprende Inicio, Elaboración, Construcción y Transición, proporcionando una estructura clara al proceso.
- Adaptabilidad: Puede aplicarse a diversos tipos de proyectos y equipos, permitiendo su uso en distintos entornos.
- AUP-UCI es adecuada para proyectos que requieren una estructura más formal, pero desean beneficiarse de las ventajas de las metodologías ágiles (Alfonso Benitez, 2017).

Conclusiones parciales

- El estudio de los conceptos y características asociadas a los portales web para la divulgación de la información permitió comprender el objeto de estudio de la investigación.
- El estudio de los sistemas homólogos ratificó la necesidad de crear una solución que resuelva las exigencias del cliente.
- El carácter de la investigación define AUP-UCI en su escenario 4 como metodología que guía el proceso de desarrollo que acompaña a la investigación.
- Se definen las herramientas y lenguajes de desarrollo a emplear durante el diseño e implementación del sistema conversacional inteligente.

CAPÍTULO 2

DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO

En el presente capítulo se presenta la propuesta de solución y se describen detalladamente las características con las que debe contar. Se identifican los requisitos funcionales y no funcionales con los que debe cumplir, así como estilo arquitectónico y los patrones de diseño. Además, se muestran los principales artefactos de ingeniería de software propuestos por la metodología utilizada, entre ellos, las historias de usuario correspondientes a dichos requisitos y los prototipos de diseño de interfaz de usuario (Lazo y Botero, 2016).

2.1. Requisitos del software

En la disciplina Requisitos el esfuerzo principal es desarrollar el modelo del sistema a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto . Por consiguiente: Requisitos del software es la descripción de los servicios y restricciones de un sistema de software, es decir, lo que el software debe hacer y bajo qué circunstancias debe hacerlo (ibíd.). Como se menciona en la introducción de la presente investigación los métodos utilizados para la captura de requisitos son la entrevista y la encuesta

2.1.1. Requisitos funcionales

Los Requisitos Funcionales (RF) de un sistema, son aquellos que describen cualquier actividad que este deba realizar, en otras palabras, el comportamiento o función particular de un sistema cuando se cumplen ciertas condiciones. Por lo general, estos deben incluir funciones desempeñadas por pantallas específicas, descripciones de los flujos de trabajo a ser desempeñados por el sistema y otros requerimientos de negocio (Pressman, 2005). Fueron identificados 12 RF, en la Tabla siguiente se muestra la especificación de estos requisitos:

Tabla 2.1. Descripción de los Requisitos Funcionales. (Fuente: Elaboración propia).

No.	Nombre	Descripción	Priorida
RF1	Enviar mensaje	El componente debe enviar al sistema el mensaje escrito por el usuario	Alta
RF2	Recibir mensaje de consulta del usuario.	El componente debe obtener el mensaje enviado por el usuario y extraer la intención del cuerpo del mensaje.	Alta
RF3	Procesar intención del usuario.	Este componente debe procesar la intención del usuario debe para hacer una búsqueda más precisa en la base de conocimiento	Alta
RF4	Enviar respuesta al usuario.	El componente debe enviar en un mensaje de texto al usuario con los resultados de la consulta realizada por éste. En caso de no poder responder a la consulta debe enviar un mensaje que diga: Lo siento, no entendí tu mensaje.	Alta
RF5	Generar PDF.	El componente de proporcionarle la posibilidad al usuario de generar un PDF de la respuesta que le dio a su incógnita.	Baja
RF6	Listar documentos	El componente de proporcionarle la posibilidad al usuario de generar un PDF de la respuesta que le dio a su incógnita.	Baja
RF7	Insertar documento	Esta funcionalidad permite insertar un documento generado por los comités de base.	Media
RF8	Parsear el contenido de un documento PDF ya insertado.	Una vez insertado, el sistema debe procesar y extraer el contenido de estos documentos para su análisis.	Alta
RF9	Particionar el contenido en fragmentos de datos.	El contenido extraído debe ser dividido en fragmentos manejables para facilitar su análisis y procesamiento.	Alta
RF10	Generar una representación de los fragmentos de datos/vectores de embeddings densos.	Los fragmentos de datos deben ser convertidos en vectores que representen su significado semántico.	Alta
RF11	Actualizar el modelo de conocimiento con los datos extraídos del documento	La información extraída debe ser integrada en la base de conocimiento semántica para su uso posterior.	Alta
RF12	Eliminar documento.	Esta funcionalidad permite eliminar un documento eliminándolo de la base de conocimiento.	Baja

2.1.2. Requisitos no funcionales

Los requisitos no funcionales representan características generales y restricciones de la aplicación o sistema que se esté desarrollando. Suelen presentar dificultades en su definición dado que su conformidad o no

conformidad podría ser sujeto de libre interpretación, por lo cual es recomendable acompañar su definición con criterios de aceptación que se puedan medir (Pressman, 2005).

Usabilidad RnF 1. Requisito de usabilidad 1

El sistema debe poseer una curva de aprendizaje baja. Se considera que el usuario debe alcanzar su máxima productividad en no menos de 15 días, estableciendo como valor crítico los 25 días de trabajo. Los usuarios deben tener un dominio del negocio, dependiendo del rol que tenga en el mismo para que se cumplan los tiempos antes definidos.

RnF 2. Requisito de usabilidad 2

En el sistema se deben visualizar todos los mensajes en idioma español. La tipografía debe ser uniforme, de un tamaño adecuado (12 px). RnF 3. Requisito de usabilidad 3

El sistema debe ofrecer una interfaz amigable, fácil de operar. Igualmente tiene que mantener la línea de diseño establecida la cual mantiene la uniformidad y representatividad de la solución. Las interfaces deben poseer un diseño sencillo, con pocas entradas, permitiendo un balance adecuado entre funcionalidad y simplicidad de tal manera que no se haga difícil para los usuarios la utilización del mismo. Confiabilidad RnF 4. Requisito de confiabilidad 1

El sistema debe permitir la visualización de la información necesaria para advertir cualquier excepción en el mismo. La información detallada es almacenada en registros de los diferentes componentes de la aplicación.

RnF 5. Requisito de confiabilidad 2

Se debe configurar un sistema de respaldo ante fallos del servidor que garantice una copia de la aplicación, así como del estado diario de la base de datos.

Eficiencia

RnF 6. Requisito de eficiencia 1

Los procesos del sistema que se implementan con transacciones donde se modifica la base de datos, deben tener tiempos de respuesta no mayores a los 3 segundos. En el caso de la obtención de información a través de transacciones que involucran consultas a la base de datos, el tiempo está dado por el volumen de la misma, por lo cual se implementará en todo momento buscando simplificar al máximo, los datos que se consulten, de manera que la cifra no exceda los 10 segundos.

Restricciones del diseño y la implementación

RnF 7. Requisito de restricción de diseño e implementación 1

La PC cliente debe poseer resolución de 1024 por 768 píxeles o superior.

RnF 8. Requisito de restricción de diseño e implementación 2

Se utilizará la herramienta CASE Visual Paradigm, en su versión 8.0, teniendo en cuenta sus ventajas para modelar los diferentes artefactos que se obtienen en los flujos de trabajo y sus diferentes fases. Las restricciones propias del diseño radican en las pautas que se establecerán, así como las diferentes relaciones que se formen durante el modelado del sistema.

RnF 9. Requisito de restricción de diseño e implementación 3

El sistema se debe desarrollar utilizando el lenguaje de programación del lado del servidor: (Python v

3.11.0+). Lenguaje de programación del lado del cliente: HTML 5, CSS 3 y JavaScript, TypeScript. Debe estar instalado en el servidor Redis para el manejo de cache, así como todas las bibliotecas y configuraciones de las que depende el funcionamiento correcto de PostgreSQL v15

2.2. Historias de usuario

Uno de los artefactos generados en la disciplina de requisitos fue Historias de usuario (HU) agrupado en el escenario 4 condicionado por el Modelado de negocio . Por consiguiente: HU constituyen descripciones cortas y esquemáticas que resumen la necesidad concreta de un usuario al utilizar un producto o servicio, así como la solución que la satisface. Su función principal es identificar problemas percibidos, proponer soluciones y estimar el esfuerzo que requieren implementar las ideas propuestas (Menzinsky; Lopez; Palacio; Sobrino; Alvarez y Rivas, 2018). En la presente solución se definieron cincuenta historias de usuario, una por cada requisito funcional, de la cuales se presenta, por su alta complejidad, la correspondiente al requisito funcional uno:

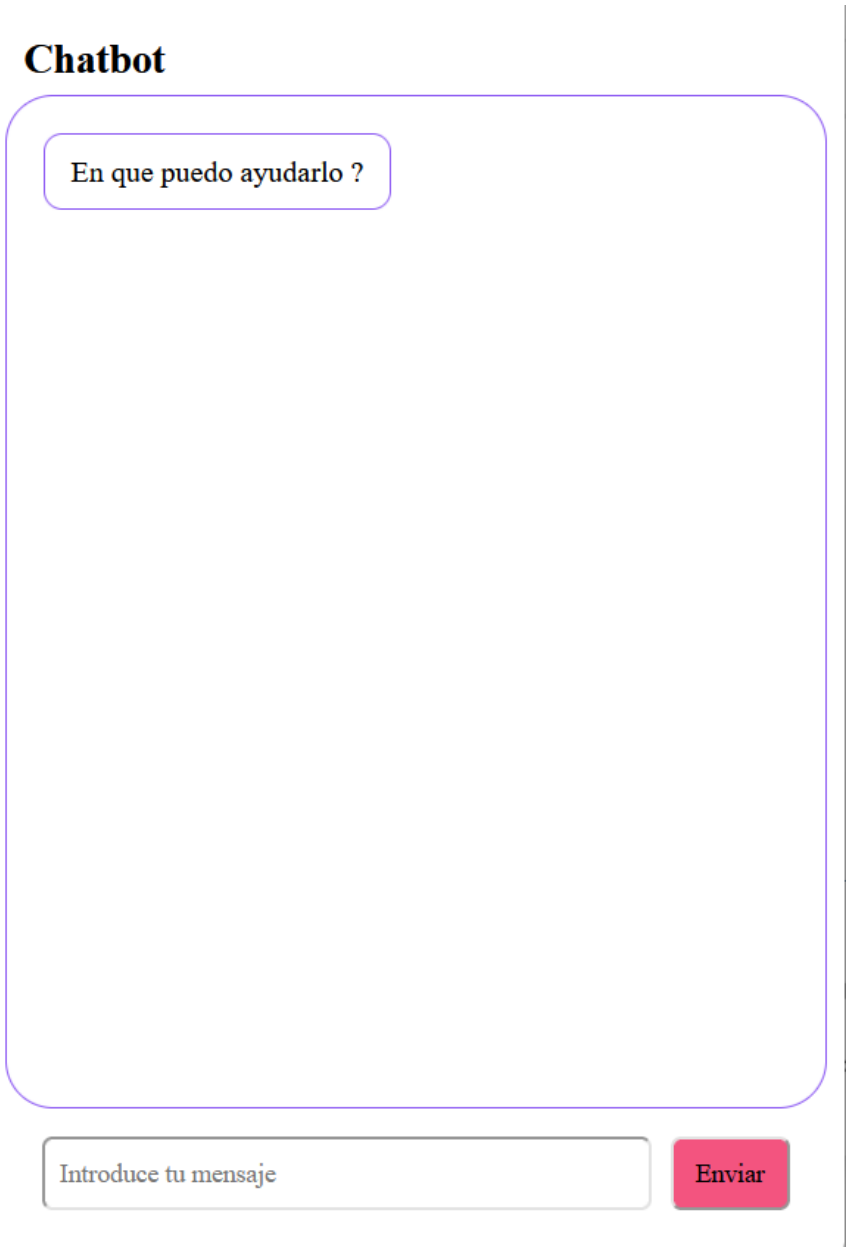
Tabla 2.2. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Enviar mensaje
Usuario: Usuario común	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** No. 1
Programador responsable: Pedro Alejandro Cabrera	
Descripción: : Al usuario abrir el chatbot, debe poder escribir una pregunta al chatbot y este debe ser enviado al sistema para ser procesada. • campo de texto para escribir la pregunta que desee hacer el usuario..	
Observaciones: -.	

Continúa en la próxima página

Tabla 2.2. Continuación de la página anterior

Interfaz:



2.3. Arquitectura de software

La arquitectura de software es la etapa en el proceso de construcción del software que constituye el enlace crucial entre el diseño y la ingeniería de requerimientos, ya que identifica los principales componentes

estructurales en un sistema y la relación entre ellos. Su salida consiste en un modelo que describe la forma en que se organiza el sistema como un conjunto de componentes en comunicación (Sommerville, 2005).El proceso de desarrollo de aplicaciones web modernas se compone principalmente de dos grandes áreas:

- En la disciplina Análisis y diseño se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis .
- El diseño arquitectónico es un proceso para identificar los subsistemas que componen un sistema y el marco para el control y la comunicación del subsistema. La salida de este proceso de diseño es una descripción de la arquitectura de software. El diseño arquitectónico es una etapa temprana del proceso de diseño del sistema (Sommerville, 2015).

A continuación, se muestran los elementos fundamentales usados por Flask para el desarrollo del proyecto:

- Modelo (Model): Capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- Vista (View): Capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada. Es considerada un puente entre el modelo y las plantillas.
- Plantilla (Template): Capa encargada de representar la estructura visual y el diseño de la interfaz de usuario. El patrón arquitectónico de Flask posee un enfoque basado en la simplicidad y la mínima complejidad.

El mismo, evita la sobrecarga innecesaria y se centra en los aspectos esenciales del sistema. Flask también se caracteriza por su flexibilidad y adaptabilidad y se basa en principios como el modularidad y la reutilización de componentes, lo que permite una fácil escalabilidad y adaptación a los cambios en los requisitos del sistema (Ye; D. Kim; Sungdong Kim; Hwang; Seungone Kim; Jo; Thorne; J. Kim y Seo, 2023).

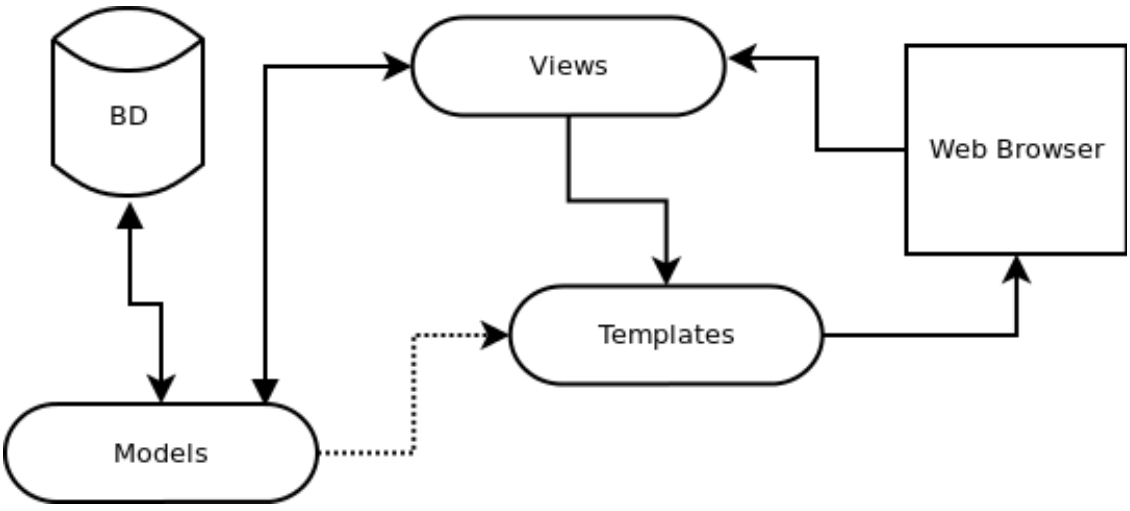


Figura 2.1. Representación de la arquitectura.

2.4. Patrones de diseño

Los patrones de diseño en ingeniería de software son soluciones generales, reutilizables y aplicables a diferentes problemas de diseño de software. Son plantillas que identifican problemas comunes en el sistema y proporcionan soluciones apropiadas a problemas generales a los que se han enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error. Estas soluciones son programas reutilizables orientados a objetos que ofrecen respuestas probadas en el tiempo a problemas de diseño recurrentes, describiendo tanto la descripción de una solución como su uso para abordar un problema concreto. A diferencia de una biblioteca o un marco de trabajo, un patrón de diseño no es algo que se pueda introducir y empezar a utilizar inmediatamente. En cambio, es una técnica de pensamiento establecida y recomendada para emplear cuando se enfrenta a un problema que muchos desarrolladores han superado previamente. Los patrones de diseño, en esencia, le ayudan a evitar hacer lo que ya se ha hecho, proporcionando una base sólida para la resolución de problemas comunes en el diseño de software (Giraldo; Acevedo y Moreno, 2011). Existen dos grupos de patrones de diseño ampliamente utilizados en el desarrollo de software: los patrones RGeneral Responsibility Assignment Software Patterns (GRASP, por sus siglas en inglés) y GOF.
Patrones GRASP Los GRASP son un conjunto de principios de diseño orientado a objetos que ayudan a tomar decisiones sobre dónde asignar responsabilidades en el código. Estos principios, explicados en el libro .Applying UML and Patterns"de Craig Larman, se enfocan en la asignación de responsabilidades de manera que se promueva la cohesión y se reduzca el acoplamiento entre los componentes del sistema. Los patrones GRASP son considerados más genéricos y fundamentales que los patrones GOF, y a menudo sirven como implementaciones de los principios GRASP . Se conocen alrededor de 8 patrones GRASP siendo estos los más usados:

- Alta cohesión: Indica la relación que existe entre los elementos de un mismo módulo. Es la medida de la relación funcional de los elementos de un módulo. El objetivo es organizar estos elementos de manera que los que tengan una mayor relación a la hora de realizar una tarea pertenezcan al mismo módulo, y los elementos no relacionados, se encuentren en módulos separados.
- Bajo acoplamiento: Impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que nos lleve a los resultados negativos que puede producir un acoplamiento alto. Es el grado de interdependencia entre los módulos. Un buen diseño se caracteriza por un acoplamiento mínimo, es decir, unos módulos tan independientes los unos de los otros como sea posible.
- Controlador: Proporciona guías acerca de las opciones generalmente aceptadas y adecuadas para manejar eventos. Es conveniente utilizar la misma clase controlador para todos los eventos del sistema de un caso de uso, de manera que es posible manejar la información acerca del estado del caso de uso en el controlador.
- Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación.

- Experto: Se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. Expresa la intuición común de que los objetos hacen las cosas relacionadas con la información que tienen. Los patrones GRASP encontrados en el código son los siguientes:

Experto: Esta función se ajusta al patrón Experto porque se le asigna la responsabilidad de extraer el texto de un PDF. La función pdf to text tiene todo el conocimiento necesario para manejar la lectura y extracción de texto de documentos PDF utilizando PyPDF2.

```python
def pdf_to_text(pdf_file):
    pdf_reader = PyPDF2.PdfReader(pdf_file)
    text = ''
    for page_num in range(len(pdf_reader.pages)):
        page = pdf_reader.pages[page_num]
        text += page.extract_text()
    return text
```

Figura 2.2. Ejemplo del patrón GRASP Experto.

Controlador: El método upload file actúa como un controlador en el patrón de diseño de controladores. Maneja la recepción de archivos, el procesamiento del texto, la generación de embeddings y la actualización de la base de datos. Se asegura de coordinar todas las operaciones necesarias para la gestión de archivos subidos.

```python
@app.route('/upload', methods=['GET','POST'])
def upload_file():
    if request.method == 'POST':
        text_input = request.form['text_input']
        pdf_file = request.files['pdf_file']
        if pdf_file.filename == '':
            return 'No selected fffile'
        print(text_input)
        pdf_text = pdf_to_text(pdf_file)
        save_to_postgresql(text_input,pdf_text)
        text_splitter = CharacterTextSplitter(
            separator="\n",
            chunk_size=650,
            chunk_overlap= 80,
            length_function=len
        )

        pdf_chunks = text_splitter.split_text(pdf_text)


        print('GENERANDO EMBEDDINGS ..................................................................')

        vectorstore = PGVector(
            embeddings=embeddingds,
            connection=connection,
            use_jsonb=True,
            async_mode=False,
            collection_name="my_collection"
        )

        documents = [ Document(page_content=pdf_chunks[i], metadata={ "id": i }) for i in range(len(pdf_chunks)) ]
        ids = [doc.metadata['id'] for doc in documents]
        print(',,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,',documents)
        print(',,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,',ids)
        vectorstore.add_documents(
            documents=documents,
            ids = ids
        )

        print('EMBEDDINGS GENERADOS y ALMACENADOS ....................................................')

        # save_to_postgresql(text_input, vectorstore._results_to_docs_and_scores())
```

Figura 2.3. Ejemplo del patrón GRASP Controlador.

Patrones Gang of Four (GOF, por sus siglas en inglés)

Por otro lado, los patrones GOF son un conjunto de 23 patrones de diseño estructurales y de comportamiento que se presentan en el libro "Design Patterns: Elements of Reusable Object-Oriented Software.escrito por Erich Gamma y varios autores. Estos patrones son soluciones probadas a problemas comunes de diseño de software y son ampliamente conocidos y utilizados en la comunidad de programación. Los patrones GOF se han popularizado en lenguajes de programación como C++ y Java, y su uso se ha extendido a través de la comunidad de programación debido a su aplicabilidad en una amplia gama de contextos de diseño(Gamma, 1995) . Se distinguen tres tipos de patrones GOF: patrones de comportamiento, patrones creacionales y patrones estructurales los cuales tienen asociados otros en cada tipología sumando 24 de ellos:

- Patrones de comportamiento: Definen la comunicación e iteración entre los objetos de un sistema. El propósito de este tipo de patrón es reducir el acoplamiento entre los objetos. Existen 11 tipos

de patrones de comportamiento, Cadena de responsabilidad, Orden, Intérprete, Iterador, Mediador, Recuerdo, Observador, Estado, Estrategia, Método plantilla y Visitante.

- Patrones estructurales: Describen cómo clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos. Existen 8 tipos de patrones estructurales, Adaptador o Envoltorio, Puente, Objeto compuesto, Decorador, Fachada, Peso ligero, Proxy y Módulo.
- Patrones creacionales: Tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados. Existen 5 tipo de patrones creacionales, Fábrica abstracta, Constructor virtual, Método de fabricación, Prototipo e Instancia única .

Entre los patrones GOF encontrados en el código tenemos:
Singleton (implícito): Aunque no está implementado explícitamente como un Singleton, esta función actúa como un único punto de acceso para obtener una conexión a la base de datos. Esto asegura que todas las operaciones de base de datos usan una única instancia de conexión, minimizando el riesgo de inconsistencias y mejorando la gestión de recursos.

```python
def pdf_to_text(pdf_file):
    pdf_reader = PyPDF2.PdfReader(pdf_file)
    text = ''
    for page_num in range(len(pdf_reader.pages)):
        page = pdf_reader.pages[page_num]
        text += page.extract_text()
    return text
```

Figura 2.4. Ejemplo del patrón GOF Singleton.

2.5. Modelo de Inteligencia Artificial

Para la implementación del requisito funcional RF9: Generar una representación de los fragmentos de datos/vectores de embeddings densos requiere del análisis y diseño del modelo de inteligencia artificial, ya que es la que requiere la IA qwen2.5-coder:0.5b. Con el objetivo de tener bien definido la entrada de los datos y la configuración de la misma. El modelo se divide en dos fases: la fase 1: se encarga de la trasformación de los datos y la fase 2: de la configuración de los datos para realizar el entrenamiento de la red neuronal de tipo Perceptrón Multicapa (MLP).

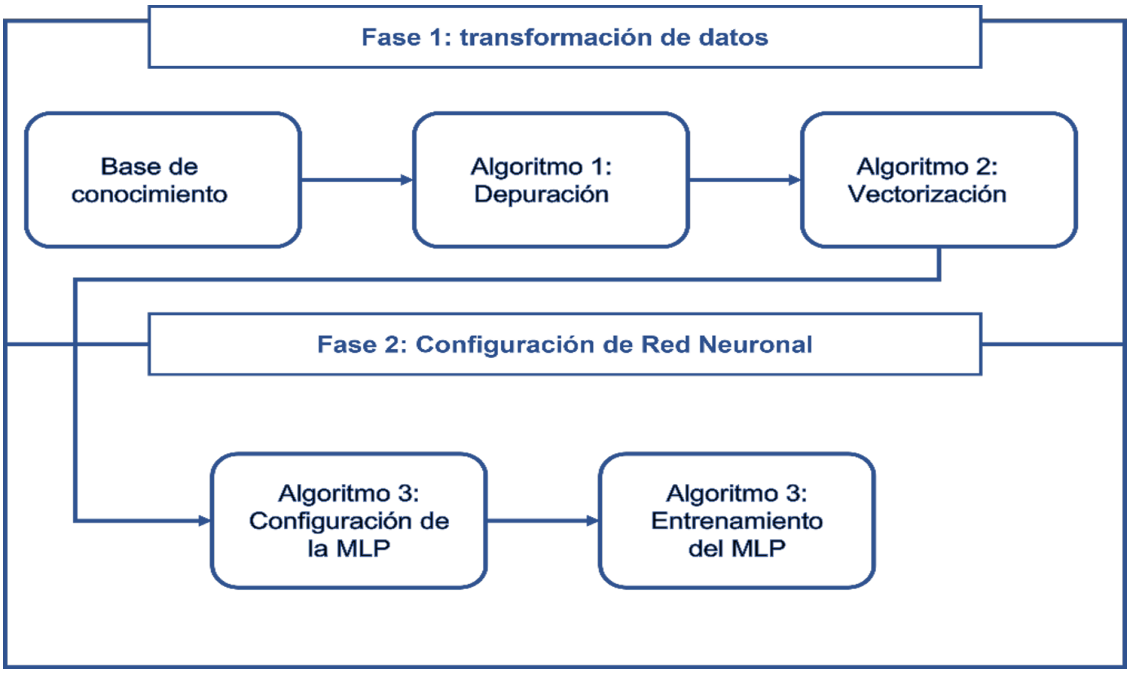


Figura 2.5. Modelo de inteligencia Artificial..

Fase 1: Trasformación de los datos En esta fase se encarga de trasformar la base de conocimiento utilizando los algoritmos 1. Depuración y el 2. Que es vectorización donde se encarga de modificar los valores para que estén listo para entrarlo al MLP.

Base de Conocimiento: Es un repositorio estructurado de datos e información que se usa para entrenar, mejorar el rendimiento y darle un contexto a la red. La diversidad y calidad de la base de conocimiento son cruciales para que el modelo pueda entender y generar lenguaje de manera efectiva en diferentes contextos. La actualización constante de esta base también es vital para que el modelo se mantenga relevante y preciso.

Algoritmo 1: Depuración: Es el proceso de identificar y corregir errores en el código y los datos usados para entrenar el modelo. Implica revisar las entradas y salidas del modelo para asegurarse de que esté funcionando como se espera y ajustando los parámetros y arquitecturas cuando sea necesario. También incluye la eliminación de datos corruptos o irrelevantes.

Algoritmo 1: Depuración.

Entrada: Base de conocimiento.

Paso1: Limpiar los datos eliminando ruido (como puntuación innecesaria o stopwords).

Paso 2: Revisión de las entradas y las salidas del modelo.

Salida: Base de conocimiento depurada

Algoritmo 2: Vectorización: es el proceso de convertir texto en vectores numéricos que representan el significado semántico de las palabras o frases. Estos vectores se almacenan en una base de datos PostgresSQL usando la extensión vector, lo que permite realizar búsquedas rápidas y eficientes de similitudes semánticas.

Algoritmo 2: Vectorización

Entrada: Base de conocimiento depurado
Paso1: Convertir el texto en tokens
Paso 2: Convertir los tokens en vectores numéricos(embeddings)
Salida: Base de conocimiento vectorizada
Fase 2: Configuración de la Red Neuronal
Esta fase se encarga de todos los datos transformados en vectores entrarlo a la red neuronal MLP con el objetivo de realizar el entrenamiento de la misma para poder clasificar los datos.
Algoritmo 3: configuración del MLP: Esta estructura incluye tanto un codificador como un decodificador, lo que le permite manejar tareas complejas de procesamiento del lenguaje natural (NLP).
Cantidad de parametros de entrada:0.5 mil millones Cantidad de capas Ocultas:24
Cantidad de neuronas de salida: Similar a las de entrada
Pesos de las aristas: Varían según la conexión con las diferentes capas
Algoritmo 3: Configuración del MLP
Entrada: vectores
Paso1: Llamar al MLP seleccionado
Paso 2: Seleccionar el nivel de temperatura
Salida: MLP configurado
Algoritmo 4: entrenamiento del MLP: a partir de la configuración de la red que se realiza en el algoritmo 3, se realiza el entrenamiento de la red neuronal y a partir de ahí, entra los nuevos patrones para la clasificación de los datos. Todo esto se realiza a través de la inteligencia artificial qwen2.5-coder:0.5b de meta.

2.6. Modelo de datos

El modelo de datos se refiere a las interrelaciones lógicas y el flujo de datos entre los diferentes elementos de datos involucrados en el mundo de la información. También documenta la forma en que se almacenan y recuperan los datos. Los modelos de datos facilitan el desarrollo técnico y comercial de la comunicación al representar con precisión los requisitos del sistema de información y al diseñar las respuestas necesarias para esos requisitos. Los modelos de datos ayudan a representar qué datos se requieren y qué formato se utilizará para los diferentes procesos comerciales. Su enfoque principal es apoyar y ayudar a los sistemas de información al mostrar el formato y la definición de los diferentes datos involucrados. También ayudan a prevenir la redundancia de datos. La información almacenada en los modelos de datos es de gran importancia para las empresas porque determina las relaciones entre las tablas de la base de datos, las claves externas y los eventos involucrados (PIÑEIRO GOMEZ, 2013). A continuación, se muestra el modelo de datos de la solución propuesta observándose las relaciones entre tablas y las columnas de cada una con sus llaves primarias y foráneas.

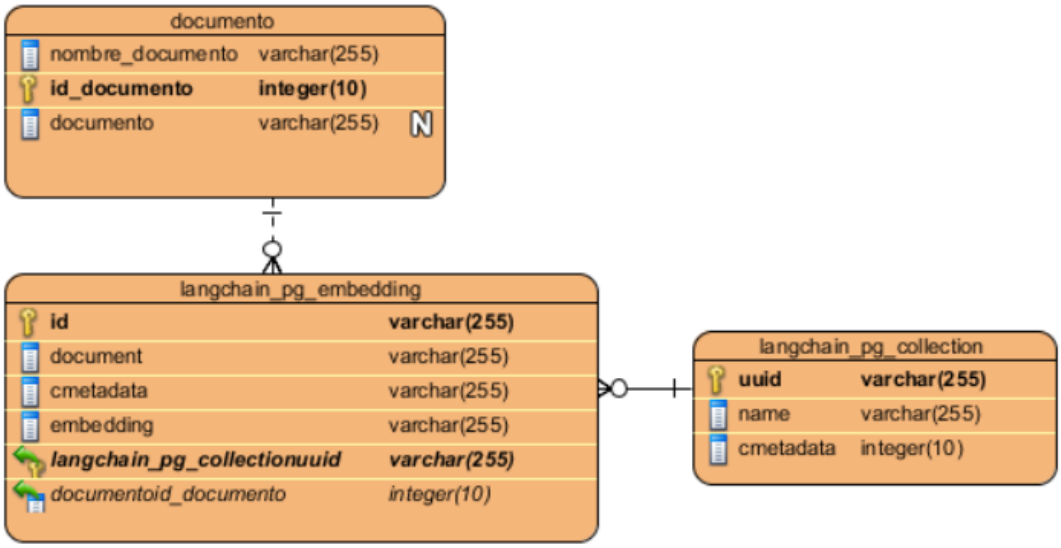


Figura 2.6. Modelo de datos de la solución propuesta

2.7. Implementación de la propuesta de solución

Sommerville afirma que la ingeniería de software incluye todas las actividades implicadas en el desarrollo de software, desde los requerimientos iniciales del sistema hasta el mantenimiento y la administración del sistema desplegado. Una etapa crítica de este proceso es, desde luego, la implementación del sistema, en la cual se crea una versión ejecutable del software. La implementación quizá requiera el desarrollo de programas en lenguajes de programación de alto o bajo niveles, o bien, la personalización y adaptación de sistemas comerciales genéricos para cubrir los requerimientos específicos de una organización (Sommerville, 2005).

2.7.1. Diagrama de despliegue

El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. Permite el mapeo de procesos dentro de los nodos, asegurando la distribución del comportamiento a través de aquellos que son representados . En la ilustración se muestra el diagrama de despliegue en donde estará desplegada la solución propuesta.

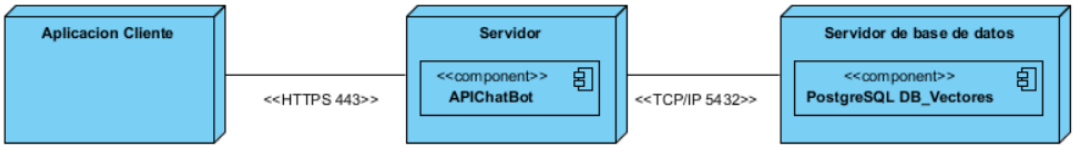


Figura 2.7. Diagrama de despliegue

Descripción de los nodos del diagrama de despliegue
Aplicación cliente: representa cualquier dispositivo que ejecute alguna aplicación consumidora de los servicios ofrecidos por la propuesta de solución.
Servidor de aplicaciones: contendrá la API implementada.
Servidor de bases de datos: contendrá la base de datos en PostgreSQL de la cual consumirá la API.
Conexiones: HTTPS 443: representa una conexión http segura, TCP/IP 5432: representa el puerto por donde se conecta la API a la base de datos en PostgreSQL.

2.7.2. Estándares de codificación

El uso de estándares de codificación, es una práctica altamente recomendada para desarrollar software de alta calidad, los estándares permiten establecer criterios únicos que los programadores deben implementar cuando escriben código, al utilizar un estándar de codificación se busca que el código fuente pueda ser entendido por cualquier miembro del equipo de desarrollo, esto permite que el código pueda ser modificado por otro programador evitando así la dependencia de personal, o en el peor de los casos tener que volver a escribir la totalidad del código, lo que ocasionaría costos extras y mayor tiempo del requerido . En el desarrollo de la implementación de la solución propuesta se usaron los siguientes estándares de codificación (Van Rossum y Drake Jr, 2017).
Nomenclatura: el idioma del código es una combinación de idioma español e inglés . El código sigue las recomendaciones de la PEP 8 , que es la guía de estilo oficial para el código Python, esta sugiere para los nombres de variables y funciones se deben escribir en minúsculas separadas por guiones bajos y el uso de líneas en blanco para separar secciones de código . Importaciones: Continuando con la guía, para las importaciones sugiere que se coloquen en la parte superior del archivo separadas por una línea y siguiendo este orden:

1. Importaciones de bibliotecas estándar.
2. Importaciones de terceros relacionados.
3. Importaciones específicas de aplicaciones/bibliotecas locales.

```
m psycopg2 import connect
m psycopg2 import extras
m cryptography.fernet import Fernet

m ia import pan

m langchain_text_splitters.character import CharacterTextSplitter
m langchain_community.embeddings import OllamaEmbeddings
m langchain_postgres.vectorstores import PGVector
m langchain_core.documents import Document
m ia import rag_chain
m generar_documentos import generar_documento
```

Figura 2.8. Ejemplo del código que cumple con los estándares según la guía de python

Indentación: Se deben usar 4 espacios de indentación .

```
@app.route('/generar', methods=['GET', 'POST'])
def descargar_pdf():
    if request.method == 'POST':
        conn = get_connection()
        cur = conn.cursor()
        cur.execute('SELECT mensaje FROM respuesta')
        pdf_data = cur.fetchone()  # Obtenemos la primera fila de resultados
        cur.close()
        conn.close()


            # Convertir la tupla a una cadena
        pdf_info = ', '.join(map(str, pdf
        print(pdf_info)
        pdf_output = generar_documento(pd
        print('ooooooooooooooooooooooooooo
        response = send_file(pdf_output, as_attachment=True, download_name='UJC_Informe.pdf', mimetype='application/pdf')
            # Enviar el archivo PDF como respuesta
        return response
```

(parameter) as_attachment: bool

as_attachment
Indicate to a browser that it should offer to save the file instead of displaying it.

Figura 2.9. Ejemplo del código que cumple con los estándares según la guía de python

Comentarios: Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación. Según la guía estos deben ser en oración completa, y la primera palabra en mayúscula .

```
# Configurar la conexion a la base de datos de postrgreSQL
connection = "postgresql+psycopg://postgres:2001@localhost:5432/vectores"
# Configurar embeddings
embeddingds = OllamaEmbeddings(model="qwen2.5-coder:0.5b")
```

Figura 2.10. Ejemplo del código que cumple con los estándares según la guía de python

Conclusiones parciales

1. La representación de un modelo conceptual en el cual se muestran las entidades que cubren el dominio del problema, así como, sus relaciones y responsabilidades, permitió una mayor comprensión de entorno en que se enmarca problema.
2. Durante el proceso de captura de requisitos se identificaron 22 requisitos funcionales y 9 requisitos no funcionales que le aportan cualidades significativas al producto.
3. Para estructurar el sistema se empleó la arquitectura modelo-vista-template, en conjunto con los patrones de diseño GRASP: controlador, alta cohesión y bajo acoplamiento y GOF: observador.
4. El diseño del modelo entidad relación representado evidencia la entidad existente en la base de datos.
5. El diseño del modelo de inteligencia artificial representando el uso del modelo de red neuronal qwen2.5-coder:0.5b

CAPÍTULO 3

VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En el presente capítulo, se abordará la fase de implementación y validación de la propuesta de solución desarrollada en este estudio. Después de la realización del análisis y diseño, es fundamental llevar a cabo la materialización y comprobación práctica de la solución propuesta, con el objetivo de evaluar su efectividad y verificar su viabilidad en un entorno real. Por ello, se realiza el modelo de implementación, se diseña y aplican las pruebas para comprobar el correcto funcionamiento de la aplicación. Posteriormente, se analizan los resultados obtenidos y se compararán con los estándares y las expectativas establecidas, permitiendo realizar conclusiones sobre la efectividad y el impacto de la solución propuesta.

3.1. Validación de requisitos

Según el libro Ïngeniería de Software"de Ian Sommerville, la validación de requisitos de software consiste en çomprobar que los requisitos realmente describen el sistema que el cliente quiere". Es decir, la validación de requisitos busca asegurar que los requisitos definidos para el sistema reflejen adecuadamente las necesidades y expectativas del cliente o usuario (Sommerville, 2005).

3.1.1. Estándares de codificación

Sommerville señala que la validación de requisitos se realiza mediante diferentes técnicas, como las siguientes:

- Revisiones de requerimientos: Los requerimientos se analizan sistemáticamente usando un equipo de revisores que verifican errores e inconsistencias. Para concretar esta técnica se hizo una revisión de cada uno de los requisitos funcionales de software en conjunto con el cliente generando un total de 2 no conformidades.
- Generación de casos de prueba: Los requerimientos deben ser comprobables. Si las pruebas para los requerimientos se diseñan como parte del proceso de validación, esto revela con frecuencia problemas en los requerimientos. Si una prueba es difícil o imposible de diseñar, esto generalmente significa que

los requerimientos serán difíciles de implementar, por lo que deberían reconsiderarse. El desarrollo de pruebas a partir de los requerimientos del usuario antes de escribir cualquier código es una pieza integral de la programación extrema. En el proceso de validación de los requisitos se crearon 16 casos de prueba, cada uno correspondiente a un requisito funcional identificado.

3.1.2. Métricas aplicadas a los requisitos

Con el objetivo de medir la calidad de la especificación de los requisitos se aplicó la métrica Calidad de la Especificación (CE). A fin de obtener cuán entendibles y precisos son los requisitos, para ello se calcula el total de requisitos de la especificación como se muestra a continuación:
Nr: el total de requisitos de especificación.
Nf: cantidad de requisitos funcionales.
Nnf: cantidad de requisitos no funcionales.
Nr = Nf + Nnf
Teniendo en cuenta que Nr = Nf + Nnf, como resultado de la sustitución de los valores, se obtiene:
Nr = 12 + 9 Nr = 21 Para determinar, la Especificidad de los Requisitos (ER) o ausencia de ambigüedad en los mismos se realiza la siguiente operación: ER = Nui / Nr Donde Nui es el número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas. Mientras más cerca de 1 esté el valor de ER, menor será la ambigüedad.
Para el caso de los requisitos obtenidos en la solución propuesta, 2 produjeron contradicción en las interpretaciones. Sustituyendo las variables se obtiene:
ER = 19/ 21 ER = 0.90
Arrojando un resultado final satisfactorio, indicando que el grado de ambigüedad de los requisitos es bajo (0.10) ya que el 0.90 son entendibles. Los requisitos ambiguos fueron modificados y validados para garantizar una correcta interpretación.

3.2. Verificación de diseño

Las métricas de diseño permiten medir de forma cuantitativa la calidad de los atributos internos del software. A nivel de componentes se centran en las características internas de los componentes del software con medidas que pueden ayudar al desarrollador a juzgar la calidad de un diseño a nivel de componente (Arregui, 2005). Para la validación del diseño se usan las métricas basadas en clases: RC y TOC. El resultado de aplicar ambas métricas permite la validación del diseño propuesto en la investigación.

3.2.1. RC

Permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas de unidad necesarias para probar una clase teniendo en cuenta las relaciones existentes entre ellas (Pressman, 2005).

La métrica RC está dada por el número de relaciones de uso de una clase con otra. El primer paso es evaluar un conjunto de atributos de calidad entre los que se encuentran el acoplamiento, complejidad de mantenimiento y reutilización de cada clase (Pressman, 2005).
Para ello mide los siguientes atributos de calidad:

- Acoplamiento: grado de dependencia de una clase o estructura de clase, con otras. Un aumento del RC provoca aumento del acoplamiento de la clase.
- Complejidad de mantenimiento: grado de esfuerzo necesario para desarrollar un arreglo, rectificación de algún error o mejora de un diseño. Un aumento del RC provoca aumento de la complejidad del mantenimiento de la clase.
- Reutilización: grado de reutilización presente en una clase o estructura de clase. Un aumento del RC provoca disminución en el grado de reutilización de la clase.
- Cantidad de pruebas: grado de esfuerzo necesario para realizar pruebas de unidad al sistema diseñado. Un aumento del RC provoca aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

El valor obtenido al aplicar la métrica debe ser directamente proporcional al acoplamiento y la complejidad de mantenimiento e inversamente proporcional con el nivel de reutilización del código. Para utilizar la métrica RC, se clasifica cada clase según la cantidad de relaciones que posea. Los criterios de evaluación para la métrica RC aplicados se muestra e la siguiente tabla.

Tabla 3.1. Métrica RC. Categoría por atributos y criterio de evaluación (Fuente: Elaboración Propia).

Atributo	Categoría	Criterios
Acoplamiento	Ninguno	RC=0
	Bajo	RC=1
	Media	RC=2
	Alta	RC>2
Complejidad de mantenimiento	Bajo	RC<=Promedio
	Media	Promedio<=RC<=Promedio*2
	Alta	RC>2* Promedio
Reutilización	Bajo	RC>2* Promedio
	Media	Promedio<=RC<=Promedio*2
	Alta	RC<= Promedio
Cantidad de pruebas	Bajo	RC<= Promedio
	Media	Promedio<=RC<=Promedio*2
	Alta	RC>2* Promedio

La gráfica muestra la representación de los resultados obtenidos agrupados en los intervalos definidos, según las relaciones de uso de cada una de las clases del diseño:

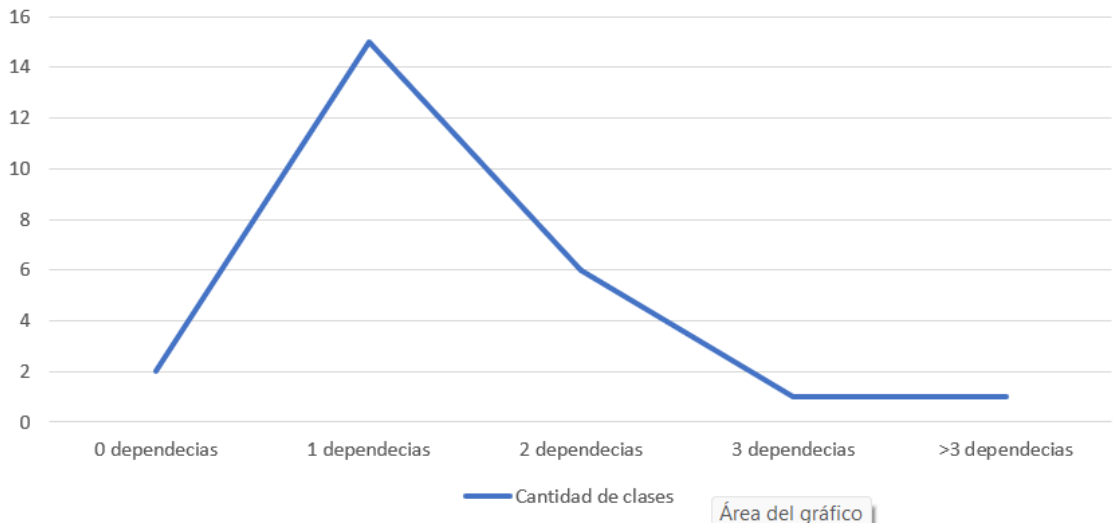


Figura 3.1. Representación de la cantidad de clases por cantidad de dependencias.

Como se muestra en la figura anterior, dos (2) del total de las clases no poseen dependencias, quince (15) poseen una sola dependencia, seis (6) clases poseen dos dependencias, una (1) clase posee 3 dependencias y por último uno (1) del total de las clases poseen 3 dependencias o más. A continuación, se muestra la representación en de los resultados obtenidos, agrupados en los intervalos definidos:

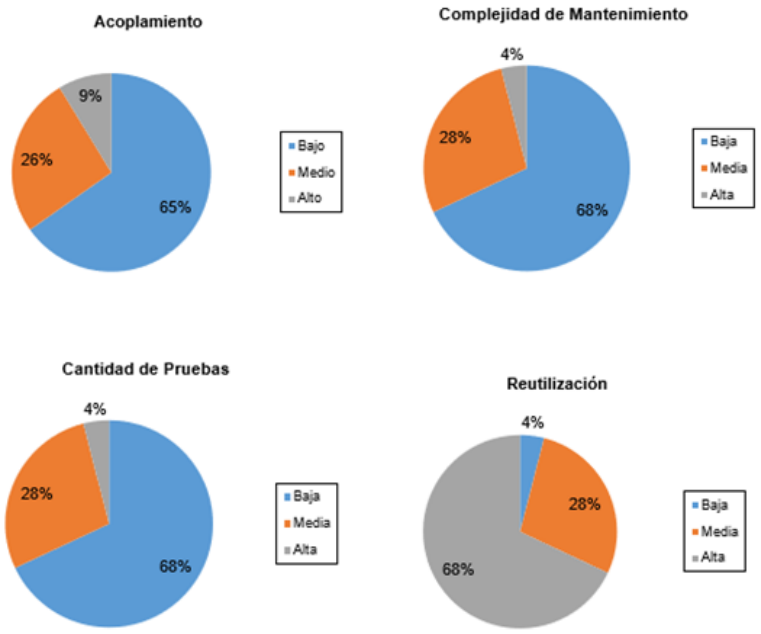


Figura 3.2. Representación en porciento del nivel de acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización de las clases .

Después de evaluar los resultados obtenidos de cada uno de los atributos de calidad que mide esta métrica,

se obtuvieron los siguientes resultados:
El 65 porciento de las clases poseen un bajo acoplamiento.
El 68 porciento de las clases poseen una baja complejidad de mantenimiento y un porciento bajo de cantidad de pruebas.
El 68 porciento de las clases poseen una alta reutilización.
El análisis de las gráficas anteriores demuestra que el acoplamiento entre las clases es bajo, lo que significa que hay poca dependencia entre ellas. Esto resulta en una alta probabilidad de reutilización. Además, se puede observar que el nivel de complejidad de mantenimiento es bajo, lo que implica que, al optimizar los métodos y las operaciones, no es necesario realizar una gran cantidad de pruebas. Esto a su vez minimiza el tiempo de implementación y pruebas de los componentes propuestos.

3.2.2. TOC

Permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas de unidad necesarias para probar una clase teniendo en cuenta las relaciones existentes entre ellas (Pressman, 2010).
Esta métrica es propuesta por Lorenz y Kidd, en su libro sobre métricas Orientada a Objetos (OO) y se centra en el recuento de atributos y operaciones para cada clase individual, y los valores promedio para el sistema como un todo (Toledo et al., 2015). Evalúa los siguientes atributos de calidad:
Responsabilidad: un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación: un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización: un aumento del TOC implica una disminución del grado de reutilización de la clase.
Una vez analizado el indicador tamaño de clase, si el valor resultante tiende al crecimiento, es probable que la clase esté saturada de responsabilidades; en consecuencia, el nivel de reutilización sería mínimo y la implementación altamente compleja. Para medir con precisión los atributos de calidad, los umbrales aplicados fueron:

Tabla 3.2. Valores de los umbrales para la métrica TOC (Fuente: Elaboración Propia).

Atributo	Categoría	Criterios
Complejidad de Mantenimiento	Baja	-<=Promedio
	Media	Entre Promedio Y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	RC=0
	Media	Entre Promedio Y 2*Promedio
	Alta	<=Promedio

Continúa en la siguiente página

Atributo	Categoría	Criterios
Cantidad de Pruebas	Bajo	<=Promedio
	Media	Entre Promedio Y 2*Promedio
	Alta	>2*Promedio

Luego de un análisis de las diferentes categorías presentes en el diseño de la propuesta de solución, en relación a la cantidad de procedimientos presentes en cada una de las clases del diseño se han obtenido los siguientes resultados:

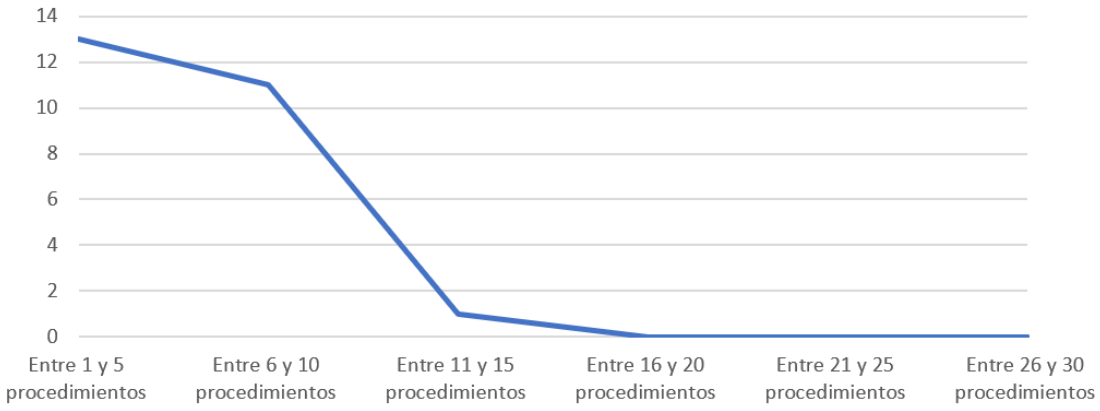


Figura 3.3. Representación de la cantidad de clases por cantidad de procedimientos que contienen.

La figura anterior muestra que, en el diseño propuesto, se identifican trece (13) clases que contienen entre 1 y 5 procedimientos, mientras que once (11) clases tienen entre 6 y 10 procedimientos, una clase (1) contienen entre 11 y 15 procedimientos y ninguna clase supera los 16 procedimientos a 30 procedimientos. Después de un análisis de estos resultados en relación a la métrica TOC, se han obtenido los siguientes indicadores específicos para cada atributo de calidad:

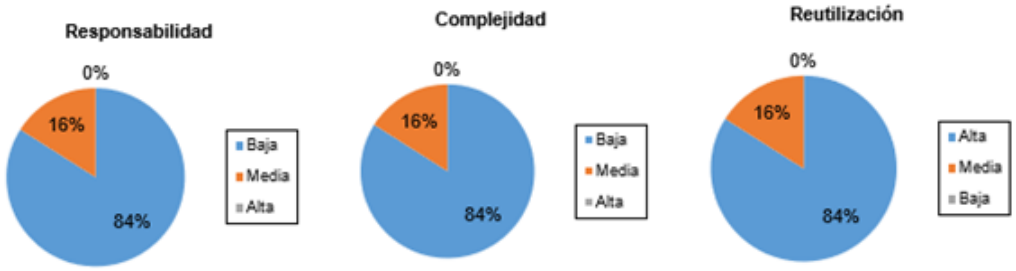


Figura 3.4. Representación en porciento del nivel de responsabilidad, complejidad de implementación y reutilización de las clases.

Haciendo un análisis de los resultados obtenidos en la evaluación de la herramienta de medición de la

métrica TOC, se puede concluir que el diseño del sistema tiene una calidad aceptable teniendo en cuenta que el 84 porciento de clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad y Complejidad de Implementación) además que el 84 porciento de las clases incluidas en este sistema poseen una alta reutilización, evidenciando que las clases no tienen tanta responsabilidad, no son tan complejas, y poseen un elevado grado de reutilización.
En términos generales, los resultados obtenidos a través de las métricas TOC y RC indican que el diseño no es complejo, las clases tienen un bajo nivel de acoplamiento y un alto grado de reutilización. Estas características favorecen la reutilización de las clases y facilitan su implementación en otros proyectos, lo que a su vez puede contribuir a una mayor eficiencia y productividad en el desarrollo de software.

3.3. Pruebas de sofware

Para (Pressman, 2005) una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados. Esta debe ser suficientemente flexible para promover un uso personalizado de la prueba. Al mismo tiempo, debe ser suficientemente rígida para alentar la planificación razonable y el seguimiento de la gestión conforme avanza el proyecto.
Continuado con el estudio de Roger S. Pressman, una estrategia para probar el software puede verse en el contexto de una espiral como muestra la ilustración 17. La prueba de unidad comienza en el vértice de la espiral y se concentra en cada unidad (por ejemplo, componente, clase o un objeto de contenido) del software como se implementó en el código fuente. La prueba avanza al moverse hacia afuera a lo largo de la espiral, hacia la prueba de integración, donde el enfoque se centra en el diseño y la construcción de la arquitectura del software. Al dar otra vuelta hacia afuera de la espiral, se encuentra la prueba de validación o aceptación, donde los requerimientos establecidos como parte de su modelado se validan confrontándose con el software que se construyó. Finalmente, se llega a la prueba del sistema, donde el software y otros elementos del sistema se prueban como un todo. Para probar el software de cómputo, se avanza en espiral hacia afuera en dirección de las manecillas del reloj a lo largo de líneas que ensanchan el alcance de las pruebas con cada vuelta.

3.3.1. Pruebas unitarias

Las pruebas unitarias se centran en probar cada componente de código de un software de forma individual para asegurar que funcione de manera apropiada como unidad. Emplean técnicas de prueba que recorren caminos específicos en la estructura de control de los componentes (ibíd.). Para la realización de este nivel de prueba se utilizó el método de prueba de Caja blanca.
Método de prueba: Caja Blanca: en ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño

de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que:

1. garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez
2. revisen todas las decisiones lógicas en sus lados verdadero y falso, 3) ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas y 4) revisen estructuras de datos internas para garantizar su validez (Pressman, 2010). Técnica de prueba

Ruta Básica: La prueba de ruta o trayectoria básica es una técnica de prueba de caja blanca, permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba . A continuación, se presenta la aplicación de esta prueba a la función listar tesis perteneciente al requisito funcional numero 6: Listar documentos

```python
@app.route('/IA')
def document():
    response=pan(1)# no se esta utilizando
    conn = get_connection()
    cur = conn.cursor()
    cur.execute('SELECT * FROM documento')
    nombre_docs = cur.fetchall()
    cur.close()
    conn.close()
    return render_template('documents.html',response=response,nombre_docs=nombre_docs)
```

Figura 3.5. Código de la función Listar documentos del RF No.6: Listar documentos.

Para obtener los casos de prueba a partir de la técnica seleccionada se debe construir el grafo de flujo correspondiente al código de la funcionalidad como se muestra en la siguiente ilustración.

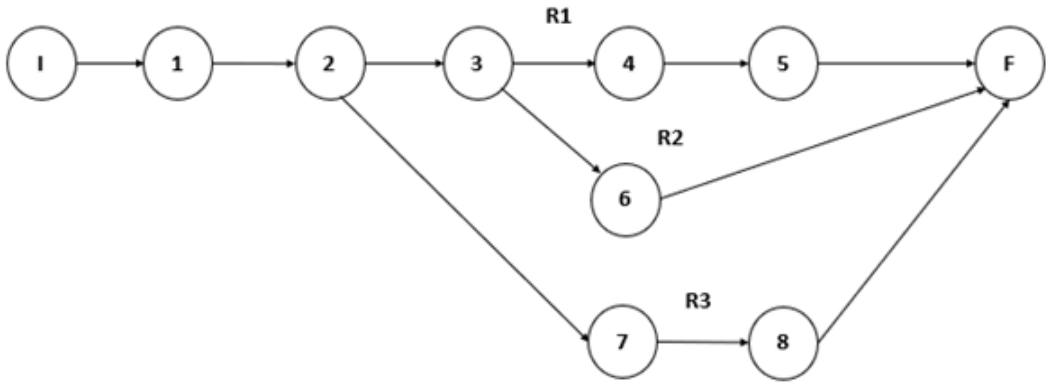


Figura 3.6. Grafo perteneciente al código de la función Listar documentos No.6: Listar documentos

Cálculo de la complejidad ciclomática

La complejidad ciclomática es una mediación de software que proporciona una evaluación cuantitativa de la complejidad lógica de un programa. En el contexto del método de prueba de ruta básica, su valor representa el número de rutas independientes del conjunto básico de un programa, brindando así una cota superior para el número de pruebas que se deben realizar para asegurar que todos los enunciados se ejecuten al menos una vez.

Existen tres formas para calcular la complejidad ciclomática del grafo anterior:

```
V (G) = E – N + 2, donde E es el número de aristas,
        y N el número de nodos de la gráfica de flujo.

V(G) = R, donde R es el número de regiones que favorece a estimar
        el valor de la complejidad ciclomática.

V (G) = P + 1, donde P es el número de nodos predicados (nodos con
        más de una arista de salida) incluidos en el grafo.
```

Figura 3.7. Formas para calcular la complejidad ciclomática

Al realizar el cálculo correspondiente para cada uno se obtiene los siguientes resultados:

```
1.  V(G)= E-N +2        2. V(G)= R        3. V(G)= P +1
    V(G)=11-10+2           V(G)= 3           V(G)= 2+ 1
    V(G)=3                                   V(G)=3
```

Figura 3.8. Resultados de la complejidad ciclomática

Luego se determinan el conjunto básico de caminos linealmente independientes

Camino básico 1: I,1,2,3,4,5,F

Camino básico 2: I,1,2,3,6,F

Camino básico 3: I,1,2,7,8,F

Con el conjunto de caminos básicos se definen los casos de pruebas a ejecutar para garantizar que todas las sentencias del código se ejecuten al menos una vez, el proceso se documenta definiendo los siguientes elementos para cada caso de prueba:

- Descripción: contiene una descripción sobre las restricciones de los datos de entrada que debe tener el caso de prueba.
- Condición de ejecución: se especifican los parámetros que debe poseer el caso de prueba para que se cumpla una condición deseada como respuesta del funcionamiento del procedimiento.
- Entrada: se muestran los parámetros de entrada al procedimiento.

- Resultados esperados: se explica el resultado esperado de la ejecución del procedimiento

Diseño de casos de prueba: Es una parte de las pruebas de componentes y sistemas en las se diseñan los casos de prueba (entradas y salidas esperadas) para probar el sistema. Su objetivo es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requerimientos. Para su diseño, se selecciona una característica del sistema o componente que se está probando, un conjunto de entradas que ejecutan dicha característica, se documentan las salidas esperadas o rasgos de salida y donde sea posible se diseña una prueba automatizada que demuestre que las salidas reales y las esperadas son las mismas . En la siguiente tabla se presenta el caso de prueba para el camino número 2.

Tabla 3.3. Diseño de caso de prueba para el camino 2

Descripción	En este caso de prueba se verifica el endpoint /documentos que se utiliza para listar todos los documentos almacenados en la base de datos
Condición de ejecución	La prueba se efectuará correctamente cuando haya documentos en la base de datos
Entrada	No se requiere ningún parámetro de entrada
Resultado esperado	Si existen documentos en la base de datos, se espera que muestre el listado con todos los documentos de la base de datos con un código de estado HTTP 200 (OK)

Resultados de la aplicación de la ruta básica Luego de completado el análisis de la complejidad ciclomática utilizado para probar la funcionalidad anterior, mostrándose el procedimiento utilizado para obtener la complejidad ciclomática, teniendo en cuenta las estructuras especificadas. Los resultados de este análisis indican el número de rutas independientes que existen dentro de un fragmento de código, la cantidad de casos de prueba que se deben diseñar y permite comprobar que el código de cada camino se ejecute correctamente. en el camino número 2 si el caso de prueba pasa con éxito, se puede concluir que el endpoint /documents"funciona correctamente y es capaz de listar las tesis almacenadas en la base de datos.

3.3.2. Pruebas funcionales

Las pruebas funcionales son aquellas que se centran en los requerimientos del software con el objetivo de ejercitar profundamente el sistema comprobando la integración del sistema globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica (Pressman, 2005).
Método de prueba

Caja Negra: Las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. Las pruebas de caja negra no son una alternativa para las técnicas de caja blanca. En vez de ello, es un enfoque complementario que es probable que descubra una clase de errores diferente que los métodos de caja blanca. Estas pruebas permiten encontrar errores en las siguientes categorías :

- Funciones incorrectas o faltantes.
- Errores de validación.
- Errores de interfaz.
- Errores en la estructura de datos.
- Errores de comportamiento o rendimiento.

Técnica de prueba
Partición de equivalencia: según esta técnica divide el dominio de entrada de un programa en clases de datos, a partir de las cuales pueden derivarse casos de prueba. Un caso de prueba ideal descubre clases de errores, que, de otra manera, requeriría la ejecución de muchos casos antes de que se observe el error general. El diseño de casos de prueba para la partición de equivalencia se basa en una evaluación de las clases de equivalencia para una condición de entrada, donde las clases de equivalencia son un conjunto de estados validos o inválidos para las condiciones de entrada.
Para aplicar el DCP, se efectuaron un total de 3 iteraciones para poder alcanzar resultados satisfactorios atendiendo al correcto comportamiento del sistema ante diferentes situaciones. En la ilustración siguiente se muestra una gráfica con 10 No Conformidades (NC) en una primera iteración y 6 en una segunda iteración y 0 en una tercera iteración.

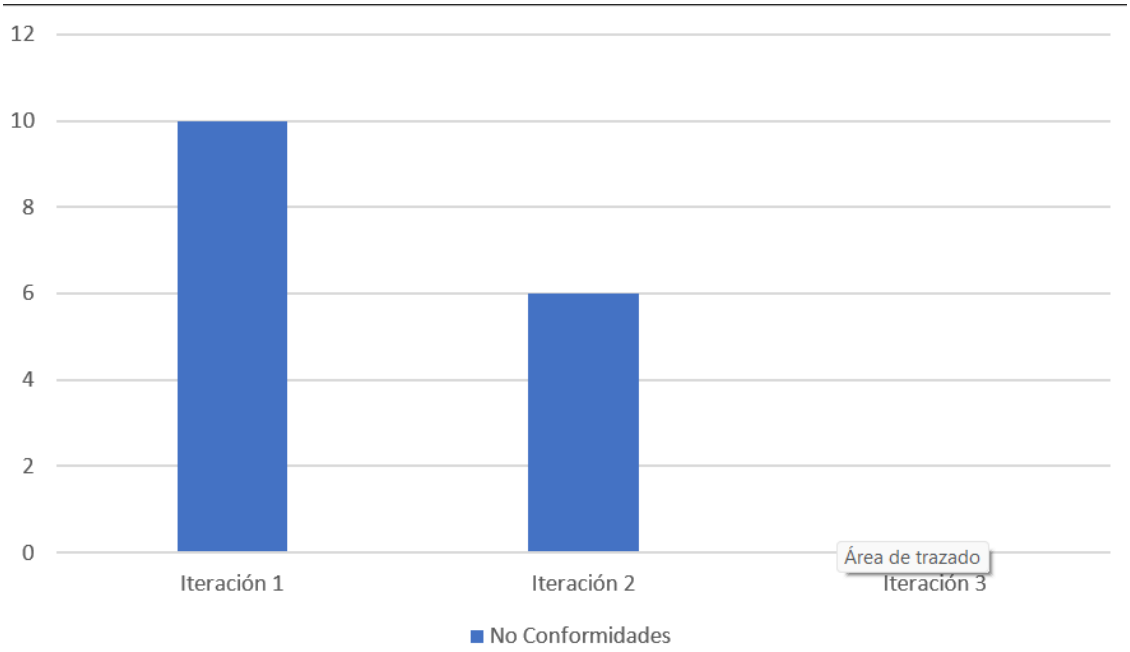


Figura 3.9. Gráfica con las no conformidades (NC) por iteración

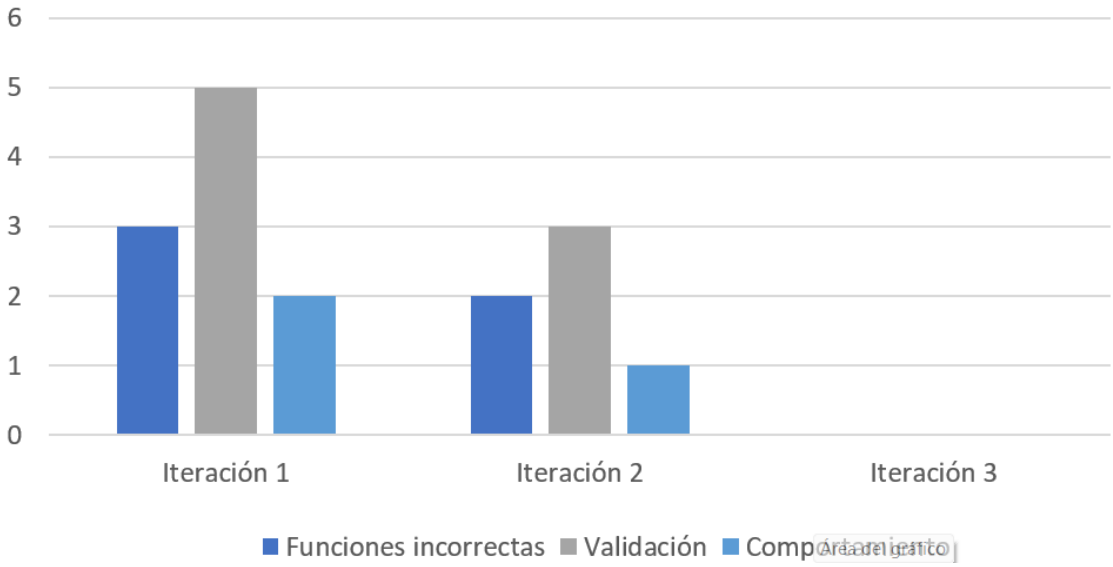


Figura 3.10. Gráfica con las no conformidades (NC) por clasificación en cada iteración

La gran mayoría de errores ocurrieron al integrar los datos del sistema de gestión con la propuesta de solución, evidenciándose respuestas erróneas por parte del sistema, también fueron solventadas otras no conformidades de inconsistencia de los datos al mostrarse en la aplicación cliente. Se lograron detectar y corregir un total 10 errores y como resultado se verificó la correcta integración de la aplicación con las soluciones anteriores.

3.3.3. Pruebas de aceptación

En su libro The Art of software testing, John Myers define las pruebas de aceptación como un proceso que consiste en comparar el programa con sus requisitos iniciales y las necesidades actuales de sus usuarios finales. Es un proceso inusual de prueba que es ejecutado usualmente por los clientes o usuarios finales y normalmente no es considerado responsabilidad del equipo de desarrollo. En caso de contratos establecidos, los usuarios comparan el sistema con el contrato original. Como en la mayoría de los tipos de pruebas, la mejor forma de ejecutar las pruebas de aceptación, es creando casos de pruebas que muestren que el programa desarrollado no cumple con el contrato inicial. Si estos casos de pruebas no son satisfactorios, entonces el sistema puede ser aceptado (G. J. Myers, 2006).

Pressman afirma que cuando se construye software a la medida para un cliente, se realiza una serie de pruebas de aceptación a fin de permitir al cliente validar todos los requerimientos. Estas puedes ser realizada por el usuario final en lugar de por los ingenieros de software. Si el software se desarrolla como un producto que va a ser usado por muchos clientes, no es práctico realizar pruebas de aceptación formales con cada uno de ellos. La mayoría de los constructores de productos de software usan un proceso llamado prueba alfa y prueba beta para descubrir errores que al parecer sólo el usuario final es capaz de encontrar.

Prueba alfa: se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El software se usa en un escenario natural con el desarrollador mirando sobre el hombro de los usuarios y registrando los errores y problemas de uso. Las pruebas alfa se realizan en un ambiente controlado.

Prueba beta: se realiza en uno o más sitios del usuario final. A diferencia de la prueba alfa, por lo general el desarrollador no está presente. Por tanto, la prueba beta es una aplicación en vivo del software en un ambiente que no puede controlar el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que se encuentran durante la prueba beta y los reporta al desarrollador periódicamente. Como resultado de los problemas reportados durante las pruebas beta, es posible hacer modificaciones y luego preparar la liberación del producto de software a toda la base de clientes.

A la API se le realizaron pruebas a nivel de aceptación de tipo alfa, con el propósito de verificar que el software está listo y cumple con cada una de las funcionalidades definidas con el cliente durante la etapa de levantamiento de requisitos. Luego de ejecutadas estas pruebas por los tutores de la presente investigación, quienes también ejercen como clientes, estos dictaminaron que la solución cumple con los requisitos definidos para su desarrollo.

3.3.4. Pruebas de rendimiento

Las pruebas de rendimiento se usan para sistemas en tiempo real y sistemas embebidos, el software que proporcione la función requerida, pero que no se adecue a los requerimientos de rendimiento, es inaceptable. Esta se diseña para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado, ocurriendo a lo largo de todos los pasos del proceso de prueba.Incluso en el nivel de unidad, puede accederse al rendimiento de un módulo individual conforme se realizan las pruebas .

Para este tipo de prueba se realizarán pruebas automatizadas de carga y estrés usando la herramienta Apache JMeter.
Pruebas de carga y estrés .
Las pruebas de carga: consisten en simular una carga de trabajo similar y superior a la que tendrá cuando el sistema esté funcionando, con el fin de detectar si el software instalado (programas y aplicaciones) cumple con los requerimientos de muchos usuarios simultáneos y también si el hardware (servidor y el equipamiento computacional de redes y enlace que lo conecta a Internet) es capaz de soportarla cantidad de visitas esperadas .
Las pruebas de estrés: evalúan la robustez y la confiabilidad del software sometiéndolo a condiciones de uso extremas. Entre estas condiciones se incluyen el envío excesivo de peticiones y la ejecución en condiciones de hardware limitadas. El objetivo es saturar el programa hasta un punto de quiebre donde aparezcan defectos potencialmente peligrosos .
Propiedades del área de desarrollo para las pruebas de rendimiento: Para la realización de las pruebas de carga y estrés se creó un entorno virtual con las siguientes especificaciones:
Hardware de prueba (PC cliente):
Sistema Operativo: Windows 11 Home
Microprocesador: AMD A6-9220 RADEON R4, 5 COMPUTE CORES 2C+ 3G 2.50 GHz Memoria RAM: 16.00 GB (3,89 GB utilizable)
Disco Duro: 512 GB
Hardware de prueba (PC servidor):
Sistema Operativo: Windows 11 Home
Microprocesador: Intel(R) Celeron(R) CPU N3350 @ 1.10GHz 1.10 GHz
Memoria RAM: 16.00 GB (3,87 GB utilizable)
Disco Duro: 512 GB
Software instalado en ambas PC: Tipo de servidor web: Apache 2.4. Plataforma: SO Windows (PC servidor) y SO Windows (PC cliente).
Servidor de BD: PostgreSQL (v 16.2.1)
Resultados de las pruebas: Luego de definido el hardware se configuran los parámetros del Apache JMeter logrando un ambiente de simulación con un total de 100 usuarios conectados concurrentemente luego se realizan peticiones a la API. En la siguiente tabla se puede observar los resultados obtenidos por el sistema.

Tabla 3.4. Comparación de sistemas similares Fuente: Elaboración propia.

Usuarios	Peticiones	Promedio	Tiempo min	Tiempo max	porciento de error	Rendimiento	Kb/s recibidos
100	200	1668	138	5163	0.0	18.8/seg	28.56
200	400	4957	12	12544	0.50	2.9/seg	0.39
300	600	7455	82	24738	2.67	1.7/seg	1.24

Las pruebas realizadas muestran que el sistema es capaz de responder a 200 peticiones de 100 usuarios co-

nectados simultáneamente en un tiempo promedio de 1668 milisegundos (1.68 segundos aproximadamente) con 0 porciento de error, esto evidencia que el sistema puede procesar la carga esperada. Por otra parte, se realizaron 400 peticiones iniciadas por 200 usuarios y en este caso el sistema respondió en 4957 milisegundos (4.957 segundos aproximadamente) como tiempo promedio. Esto demuestra que el sistema puede procesar la carga esperada, aunque no fue capaz de responder correctamente el 0.50 porciento de las peticiones realizadas. Por último, y con el objetivo de analizar el comportamiento del sistema en condiciones extremas, se realizó una prueba de estrés para un conjunto de 300 usuarios conectados simultáneamente. En este caso, el sistema responde a las 600 peticiones en un tiempo promedio de 7455 milisegundos (7.455 segundos aproximadamente), con un por ciento de error de 2.67 porciento. Este resultado está estrechamente relacionado al entorno donde se realizó la prueba, el cual no es un servidor dedicado sino un cliente habilitado.

3.4. Validación de las variables de investigación

Teniendo en cuenta que en la investigación realizada se define como idea a defender que: Si se desarrolla la API para el sistema de gestión de trabajos de diploma del CIGE y la aplicación móvil GraduApp se unificaría la fuente de consulta de información. Para analizar la causa y efecto entre la variable independiente: el desarrollo de la API y la variable dependiente: unificar la fuente de consulta de información, la autora de la presente investigación, definen un conjunto de criterios de medida que permiten validar cómo a través de la API se logra la relación entre ambas variables. La obtención de estos criterios se basa en las deficiencias identificadas en la situación problemática que conducen al desarrollo de la solución propuesta. Criterio de medida definidos Unificar la fuente de consulta de información: este criterio se refiere a que como existen dos sistemas operando con diferentes bases de datos la información se encuentra dividida. Homogenización del proceso de gestión de tesis: este criterio se refiere la necesidad de dedicar ambos sistemas a gestionar todo lo referente a las tesis del centro. Disminuir carga de trabajo: este criterio se refiere a que una vez implementado el sistema el tiempo dedicado a realizar algún proceso desde el sistema de gestión puede verse reducido al realizarse desde la aplicación móvil.

Tabla 3.5. Comparación de sistemas similares Fuente: Elaboración propia.

Criterios	Antes de la solución	Después de la solución
Unificar la fuente de consulta de información.	Existían dos bases de datos diferentes, por lo que, al intentar buscar un recurso desde la aplicación móvil, este podría no encontrarse disponible.	Al trabajar los dos sistemas con una misma base de datos la información del sistema de gestión, estará disponible en cualquier momento desde la aplicación móvil.
Homogenizar el proceso de gestión de tesis.	Los procesos de gestión y obtención se hacían por separado.	Entre las funcionalidades esta la obtención y visualización desde la aplicación móvil de la información presente en el sistema de gestión, además de la modificación del estado de las no conformidades desde la aplicación móvil.
Disminuir la carga de trabajo.	Al existir diferentes bases de datos la información en la base de datos de la aplicación móvil debía ser llenada de forma manual.	Una vez integrados los sistemas la información que se visualizara desde la aplicación móvil es la misma que sede el sistema de gestión.

3.5. Evaluación del modelo con Bleu

Se procede a la aplicación de la métrica para evaluar la calidad del texto generado por el asistente y la coincidencia con el texto de referencia. La hipótesis es la respuesta generada por los modelos respectivamente, y la referencia es un texto generado por una persona en base a la respuesta del modelo. El documento utilizado para realizar la evaluación es un acta sacada de unos de los comités de base del cerro.
Modelo: qwen2.5-coder:0.5b
Hipótesis: Una de las quejas de los trabajadores del centro es que están resentidos por la mala gestión de los directivos a la hora de tomar decisiones con respecto a las negligencias que se están cometiendo en el mismo perjudicando asi el trabajo de varios de los integrantes de la Unión de Jóvenes Comunistas.
Referencia: Los directivos están dejando sin condena alguna a los miembros que están incumpliendo con las guardias haciendo que otros los cubran para que los puestos de guardia no se queden vacíos.

Tabla 3.6. Comparación de sistemas similares Fuente: Elaboración propia.

Metrica	Referencia
Individual 1-gram	0.692308
Individual 2-gram	0.656250
Individual 3-gram	0.619048
Individual 4-gram	0.580645
Penalización por brevedad	1.0082304

El puntaje BLEU obtenido es de aproximadamente 0.64. Esto indica que la generación de texto tiene una calidad razonable en términos de la coincidencia de 4-gramas con las referencias (BLEU Score - Microsoft Azure 2024).

Conclusiones parciales

- En base a los resultados obtenidos en las pruebas de chatbot, se concluye que el modelo qwen2.5-coder:0.5b es el más adecuado para su implementación. Este modelo se destacó por su rendimiento superior en la métrica BLEU, lo que indica una mayor precisión en la generación de respuestas en lenguaje natural. Además, este modelo fue el que consumió menos recursos computacionales, lo que es un factor importante a considerar en la implementación de chatbots, especialmente en entornos donde el rendimiento y la eficiencia son críticos.
- Por lo tanto, se recomienda la utilización del modelo "qwen2.5-coder:0.5b"para la implementación de chatbots, debido a su eficiencia y rendimiento superior. Sin embargo, es importante tener en cuenta que la elección del modelo puede variar dependiendo de las necesidades específicas del proyecto, y se deben realizar pruebas adicionales para validar su efectividad en diferentes contextos y aplicaciones.

Conclusiones

El presente Trabajo de Diploma concluye con el desarrollo de un sistema conversacional inteligente para el proceso de análisis de datos de la UJC del Cerro, el cual sirve de apoyo a la gestión del quehacer de la organización. Se destaca, además que:

- A partir de la sistematización de los principales referentes teóricos que sustenta la investigación se evidencia que el diseño de la propuesta de solución permitió generar los artefactos más significativos de acuerdo con la metodología de desarrollo de software AUP-UCI.
- La implementación del sistema a través de las herramientas y lenguajes seleccionados permitió obtener un sistema conversacional capaz de gestionar la información referente a las actas de los comités de base del Municipio del Cerro; logrando el cumplimiento del objetivo planteado.
- La definición de una estrategia de prueba, permitió comprobar el correcto funcionamiento del sistema conversacional inteligente para el proceso de análisis de datos de la UJC del Cerro, a partir del cumplimiento de los requisitos definidos por el cliente.
- Las técnicas de validación aplicadas a la propuesta de solución permitieron la detección y corrección de las no conformidades detectadas y evidenciaron que el sistema conversacional constituye una solución funcional.

Recomendaciones

Añadir una sección de configuración para el Chatbot y un historial de conversaciones

Referencias bibliográficas

ACHIAM, Josh et al., 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (vid. pág. 7).

ADAMOPOULOU, Eleni y MOUSSIADES, Lefteris, 2020. An overview of chatbot technology. En: *An overview of chatbot technology*. *IFIP international conference on artificial intelligence applications and innovations*, págs. 373-383 (vid. pág. 4).

ALAMINOS-FERNANDEZ, Antonio Francisco, 2023. Introduccion a la mineria de texto y analisis de sentimiento con R (vid. pág. 9).

ALFONSO BENITEZ, Drymon, 2017. *Herramienta para generar productos de trabajos de la metodologia variacion AUP-UCI*. B.S. thesis. Universidad de las Ciencias Informáticas. Facultad 3. (vid. pág. 12).

ANBU, Akhilan; MIRIAM, D Doreen Hephzibah y ROBIN, CR Rene, 2024. A Comprehensive WebScraping of IMDbs Top 50 Movies using Beautiful Soup. En: *A Comprehensive WebScraping of IMDbs Top 50 Movies using Beautiful Soup*. *2024 International Conference on Communication, Computing and Internet of Things (IC3IoT)*, págs. 1-9 (vid. pág. 11).

ARDA SUAREZ, Antonio; NIU, Zunqi e IVAN-BARAGANO, Iyan, 2023. Analisis multivariante mediante arbol de decision de los tiros libres indirectos en la Superliga China 2020. *Retos: nuevas tendencias en educacion fisica, deporte y recreacion*. N.º 48, págs. 358-365 (vid. pág. 9).

ARREGUI, Juan Jose Olmedilla, 2005. Revision Sistematica de Metricas de Diseno Orientado a Objetos. *Madrid, Espana.: Universidad Politecnica de Madrid, Facultad de Informatica* (vid. pág. 30).

CALVO, Patrici, 2023. Metaverso: desafios eticos de la tokenizacion de la economia. *Filosofia Unisinos*. Vol. 24, págs. e24106 (vid. pág. 8).

DIAZ LUPONE, Mauro y TAFKA GHAZAL, Yasser, 2024. Extraccion de relaciones basadas en entidades nombradas en espanol utilizando tecnicas de procesamiento de lenguaje natural (vid. pág. 9).

DUBEY, Abhimanyu et al., 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (vid. pág. 7).

GAMMA, Erich, 1995. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley (vid. pág. 21).

GELBUKH, Alexander, 2010. Procesamiento de lenguaje natural y sus aplicaciones. *Komputer Sapiens*. Vol. 1, págs. 6-11 (vid. pág. 5).

GIRALDO, Gloria L; ACEVEDO, Juan F y MORENO, David A, 2011. Una ontologia para la representacion de conceptos de diseno de software. *Revista Avances en Sistemas e Informatica*. Vol. 8, n.º 3, págs. 103-110 (vid. pág. 19).

GREENBERG, Mark T; LENGUA, Liliana J; COIE, John D; PINDERHUGHES, Ellen E; BIERMAN, Karen; DODGE, Kenneth A; LOCHMAN, John E y MCMAHON, Robert J, 1999. Predicting developmental outcomes at school entry using a multiple-risk model: Four American communities. *Developmental psychology*. Vol. 35, n.º 2, págs. 403 (vid. pág. 5).

HUI, Binyuan et al., 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186* (vid. pág. 10).

JAIN, Shashank Mohan, 2022. Hugging face. En: *Hugging face*. *Introduction to transformers for NLP: With the hugging face library and models to solve problems*. Springer, págs. 51-67 (vid. pág. 11).

JIANG, Shuming y LI, Yan, 2023. Research and Implementation of PDF Specific Element Fast Extraction. En: *Research and Implementation of PDF Specific Element Fast Extraction*. *2023 4th International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*, págs. 77-83 (vid. pág. 8).

KANG, Yue; CAI, Zhao; TAN, Chee-Wee; HUANG, Qian y LIU, Hefu, 2020. Natural language processing (NLP) in management research: A literature review. *Journal of Management Analytics*. Vol. 7, n.º 2, págs. 139-172 (vid. pág. 1).

KATZ, Raul Luciano, 2009. *El papel de las TIC en el desarrollo*. Raul Katz (vid. pág. 5).

KHADDAD, A; ROBERT, G; GROSS-GOUPIL, M; DESLANDES, M; CAPON, G; ALEZRA, E; BLANC, P; BERNHARD, J y BLADOU, F, 2023. Curage lombo-aortique par voie laparoscopique robot-assisté pour la prise en charge de tumeurs germinales métastatiques ganglionnaires. *Progrès en Urologie-FMC*. Vol. 33, n.º 3, págs. S133-S135 (vid. pág. 6).

KIM, Nam Wook; KO, Hyung-Kwon; MYERS, Grace y BACH, Benjamin, 2024. ChatGPT in Data Visualization Education: A Student Perspective. *arXiv preprint arXiv:2405.00748* (vid. pág. 6).

LAZO, Alonso Toro y BOTERO, Juan Guillermo Galvez, 2016. Especificacion de requisitos de software: una mirada desde la revision teorica de antecedentes. *Entre Ciencia e Ingenieria*. Vol. 10, n.º 19, págs. 108-115 (vid. pág. 13).

LIU, Zheng-Wei; RÖPKE, Friedrich K y HAN, Zhanwen, 2023. Type Ia Supernova Explosions in Binary Systems: A Review. *Research in Astronomy and Astrophysics*. Vol. 23, n.º 8, págs. 082001 (vid. pág. 7).

MARTIN, Erwin Blanco-San; SAEZ-DELGADO, Fabiola y LEPE-MARTINEZ, Nancy, 2023. El rol predictivo de la red neuronal por defecto sobre la atencion sostenida en edades escolares: una revision sistematica. *Revista chilena de neuro-psiquiatria*. Vol. 61, n.º 1, págs. 87-97 (vid. pág. 9).

MEDINA, Javier; CABEZA, Eduardo M Eisman y PEÑA, Juan Luis Castro, 2013. Asistentes virtuales en plataformas 3.0. *IE comunicaciones: revista iberoamericana de informática educativa*. N.º 18, págs. 41-49 (vid. pág. 4).

MENZINSKY, Alexander; LOPEZ, Gertrudis; PALACIO, Juan; SOBRINO, Miguel Angel; ALVAREZ, Ruben y RIVAS, Veronica, 2018. Historias de usuario. *Ingenieria de requisitos agil* (vid. pág. 16).

MYERS, Glenford J, 2006. *The art of software testing*. John Wiley & Sons (vid. pág. 41).

OYEWOLE, Gbeminiyi John y THOPIL, George Alex, 2023. Data clustering: application and trends. *Artificial Intelligence Review*. Vol. 56, n.º 7, págs. 6439-6475 (vid. pág. 9).

PANDA, Subhajit, 2023. Enhancing PDF interaction for a more engaging user experience in library: Introducing ChatPDF. *IP Indian Journal of Library Science and Information Technology*. Vol. 8, n.º 1, págs. 20-25 (vid. pág. 5).

PINARGOTE-ZAMBRANO, Juan Jose; LINO-CALLE, Victor Alejandro; VERA-ALMEIDA, Boris Jerfreir et al., 2024. Python en la ensenanza de las Matematicas para estudiantes de nivelacion en Educacion Superior. *MQRInvestigar*. Vol. 8, n.º 3, págs. 3966-3989 (vid. pág. 10).

PIÑEIRO GOMEZ, JOSE, 2013. *Bases de datos relacionales y modelado de datos*. Ediciones Paraninfo, SA (vid. pág. 24).

PRESSMAN, Roger S, 2005. Software Engineering: a practitioners approach. *Pressman and Associates* (vid. págs. 13, 15, 30, 31, 35, 38).

REBACK, Jeff et al., 2020. pandas-dev/pandas: Pandas 1.0. 5. *Zenodo* (vid. pág. 11).

ROUHIAINEN, Lasse, 2018. Inteligencia artificial. *Madrid: Alienta Editorial*, págs. 20-21 (vid. pág. 5).

SOMMERVILLE, Ian, 2005. *Ingenieria del software*. Pearson educacion (vid. págs. 18, 25, 29).

SURESH, Yeresime et al., 2023. Machine Learning-Based Recommendations and Classification System for Unstructured Resume Documents. *Revue d'Intelligence Artificielle*. Vol. 37, n.º 3 (vid. pág. 8).

TIAN, Yuan; THUNG, Ferdian; SHARMA, Abhishek y LO, David, 2017. APIBot: question answering bot for API documentation. En: *APIBot: question answering bot for API documentation. 2017 32nd IEEE/ACM international conference on automated software engineering (ASE)*, págs. 153-158 (vid. pág. 6).

TOPSAKAL, Oguzhan y AKINCI, Tahir Cetin, 2023. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. En: *Creating large language model applications utilizing langchain: A primer on developing llm apps fast. International Conference on Applied Engineering and Natural Sciences*. Vol. 1, págs. 1050-1056. N.º 1 (vid. pág. 11).

VAN ROSSUM, Guido y DRAKE JR, Fred L, 2017. *El tutorial de Python*. Python Software Foundation (vid. pág. 26).

VÁZQUEZ, Maikel Yelandi Leyva; RICARDO, Jesús Estupiñán y VEGA-FALCÓN, Vladimir, 2022. La inteligencia artificial y su aplicación en la enseñanza del Derecho. *Estudios del desarrollo social: Cuba y América Latina*. Vol. 10, págs. 368-380 (vid. pág. 7).

VILORIA, Amelec; ACUÑA, Genesis Camargo; FRANCO, Daniel Jesús Alcázar; HERNÁNDEZ-PALMA, Hugo; FUENTES, Jorge Pacheco y RAMBAL, Etelberto Pallares, 2019. Integration of data mining techniques to PostgreSQL database manager system. *Procedia Computer Science*. Vol. 155, págs. 575-580 (vid. pág. 12).

WU, Shengqiong; FEI, Hao; QU, Leigang; JI, Wei y CHUA, Tat-Seng, 2023. Next-gpt: Any-to-any multimodal llm. *arXiv preprint arXiv:2309.05519* (vid. pág. 7).

YE, Seonghyeon; KIM, Doyoung; KIM, Sungdong; HWANG, Hyeonbin; KIM, Seungone; JO, Yongrae; THORNE, James; KIM, Juho y SEO, Minjoon, 2023. Flask: Fine-grained language model evaluation based on alignment skill sets. *arXiv preprint arXiv:2307.10928* (vid. págs. 11, 18).

ZHANG, Mingxuan; YUAN, Bo; LI, Hanzhe y XU, Kangming, 2024. LLM-Cloud Complete: Leveraging cloud computing for efficient large language model-based code completion. *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023*. Vol. 5, n.º 1, págs. 295-326 (vid. pág. 11).

ZHAO, Zhengting; WANG, Jing y CHAI, Mingjiong, 2023. Closed-Loop Automation with Python in Translation Technology Teaching–The Acquisition of a Meta-technology Competence Through Project-Based Learning. En: *Closed-Loop Automation with Python in Translation Technology Teaching–The Acquisition of a Meta-technology Competence Through Project-Based Learning*. *International Symposium on Emerging Technologies for Education*, págs. 295-310 (vid. pág. 8).

Generado con LaTeX: 30 de noviembre de 2024: 2:30pm

Apéndices

.1. Historias de usuario

Tabla 7. Historia de usuario # 2

Historia de usuario	
Número: 2	**Nombre:** Listar documentos
Usuario: Usuario común	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** No. 2
Programador responsable: Pedro Alejandro Cabrera	
Descripción: : El usuario debe poder ver una lista de los documentos que se han subido al sistema y con los cuales el chatbot está interactuando. Botones: • Eliminar: permite eliminar el documento que se selecciona	
Observaciones: -.	
Interfaz: no aplica	

Tabla 8. Historia de usuario # 3

Historia de usuario	
Número: 3	**Nombre:** Generar PDFs
Usuario: Usuario común	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** No. 3
Programador responsable: Pedro Alejandro Cabrera	
Descripción: : El usuario debe poder generar un pdf a partir de una respuesta dada por el chatbot. Botones: • PDF: permite generar un pdf con la información dada por el chatbot.	
Observaciones: -.	
Interfaz: no aplica	

.2. Aval del cliente

La Habana, 21 de noviembre del 2024

"Año 66 de la Revolución Cubana"

De: El comité Municipal UJC Cerro

A: Universidad de las Ciencias Informáticas

Estimados miembros del tribunal:

Por medio de la presente, me complace informarles sobre la aceptación e implementación de un sistema conversacional inteligente que tiene como objetivo principal realizar un análisis de tendencias en las conversaciones que se llevan a cabo en los comités de bases de nuestro municipio.

Este sistema nos permitirá recopilar y analizar información valiosa sobre las inquietudes, propuestas y necesidades de nuestros jóvenes, facilitando así una mejor toma de decisiones y el diseño de políticas que respondan a sus expectativas. La tecnología se convierte en una herramienta fundamental para fortalecer nuestra comunicación interna y potenciar el trabajo en nuestras comunidades.

Estamos convencidos de que este avance tecnológico no solo optimizará nuestros procesos, sino que también contribuirá a la creación de un entorno más participativo y dinámico, donde cada voz sea escuchada y considerada. La implementación del sistema se llevará a cabo de manera gradual, asegurando que todos los miembros estén capacitados y familiarizados con su uso.

Agradecemos el apoyo y compromiso de todo el equipo de desarrollo, en especial al desarrollador de este sistema conversacional, el estudiante Pedro Alejandro Cabrera Enríquez que ha sido un eslabón fundamental en este proceso. Juntos, podremos construir un futuro más inclusivo y representativo para nuestros jóvenes.

Atentamente,

Comité Municipal Primer Secretario

Msc. Bienvenido Hanley Roque Orfe

Primer Secretario de la UJC del Municipio Cerro

XIAOMI 13T | LEICA 25/11/2024 23:11

Figura 11. Acta de aceptación.