

Universidad de Oriente

Sede “Julio Antonio Mella”

Facultad de Ingeniería en Telecomunicaciones, Informática y Biomédica

Trabajo de Diploma

En opción al título de Ingeniera en Informática

**Título:** “Plataforma de creación automática de asistentes virtuales con audio y video.”.

**Autora:** Claudia Queipo García.

**Tutor:** Dr. C. Dionis López Ramos.

Santiago de Cuba,

“Año de la Revolución”

Resumen

Los asistentes virtuales se han vuelto cada vez más populares y se espera que su uso continúe creciendo en el futuro. Estos sistemas de inteligencia artificial son capaces de comprender y responder a las preguntas y solicitudes del usuario de manera natural, lo que los convierte en una herramienta útil para automatizar tareas y mejorar la eficiencia en distintos ámbitos.

En esta tesis, se aborda la continuación de la investigación "Herramienta digital para la construcción de conocimiento automático para un Asistente Virtual" del año 2022 sobre creación un sistema que permita crear asistentes virtuales utilizando el marco de desarrollo de software RASA, y la integración de herramientas de procesamiento del lenguaje natural, como Spacy, para mejorar la capacidad de los asistente para comprender y responder a las preguntas y solicitudes del usuario. Además, se implementaron tecnologías de reconocimiento de voz para convertir audio a texto y permitir al asistente virtual interactuar con el usuario a través de la voz.

La tesis demuestra que es posible crear asistentes virtuales eficientes y naturales usando RASA, procesamiento del lenguaje natural y reconocimiento de voz con Whisper. Se evaluaron y entrenaron los asistentes, obteniendo resultados prometedores en precisión y comprensión del lenguaje natural. En conclusión, se confirma la viabilidad de esta tecnología.

Palabras clave: Asistente Virtual, RASA, Reconocimiento de voz, Whisper

*ABSTRACT*

Virtual assistants have become increasingly popular, and their usage is expected to continue growing in the future. These artificial intelligence systems can understand and respond to user questions and requests naturally, making them a valuable tool for automating tasks and enhancing efficiency in various domains.

In this thesis, the continuation of the research "Digital Tool for the Construction of Automatic Knowledge for a Virtual Assistant" from 2022 is addressed, focusing on the development of a system that enables the creation of virtual assistants using the RASA software development framework, and the integration of natural language processing (NLP) tools, such as Spacy, to enhance the assistant's ability to understand and respond to user questions and requests. Additionally, voice recognition technologies were implemented to convert audio to text, allowing the virtual assistant to interact with the user through voice.

The thesis demonstrates the possibility of creating efficient and natural virtual assistants using RASA, natural language processing, and voice recognition with Whisper. The assistants were evaluated and trained, yielding promising results in accuracy and natural language understanding. In conclusion, the feasibility of this technology is confirmed.

**Keywords**: Virtual Assistant, RASA, Voice Recognition, Whisper

índice

[INTRODUCCIÓN 7](#_Toc1)

[CAPITULO 1 . MARCO TEÓRICO 12](#_Toc2)

[Procesamiento del Lenguaje Natural. 12](#_Toc3)

[Asistente Personal Inteligente. 12](#_Toc4)

[Reconocimiento Automático del habla. 13](#_Toc5)

[Estado del arte de las herramientas. 14](#_Toc6)

[DialogFlow 14](#_Toc7)

[Amazon Lex 14](#_Toc8)

[IBM Watson 15](#_Toc9)

[RASA marco de trabajo 16](#_Toc10)

[BotPress 16](#_Toc11)

[Herramienta de Reconocimiento automático del Habla y Transcripción de Voz a Texto. 17](#_Toc12)

[Lenguajes de Pogramación utilizados. 18](#_Toc13)

[Python 18](#_Toc14)

[Marco de Trabajo de Interfaz de Usuario. 19](#_Toc15)

[Marco de trabajo del servidor 20](#_Toc16)

[Langchain 24](#_Toc17)

[Base de datos Vectorial. 24](#_Toc18)

[Herramienta para el reconocimiento de rostros. 25](#_Toc19)

[LLM a utilizar en el algoritmo de generación de preguntas y respuestas 26](#_Toc20)

[Base de Datos utilizada. 29](#_Toc21)

[Entorno de desarrollo. 29](#_Toc22)

[Sitema de Control de Versiones GIT. 30](#_Toc23)

[XP 30](#_Toc24)

[Conclusiones del capítulo. 31](#_Toc25)

[CAPITULO 2 . ORGANIZACIÓN Y DISEÑO 32](#_Toc27)

[Propuesta de Sistema. 32](#_Toc28)

[Arquitectura del sistema. 32](#_Toc29)

[Historias de Usuario. 34](#_Toc32)

[Autenticar al Usuario en el Sistema. 34](#_Toc33)

[Crear asistente. 35](#_Toc34)

[Administrar asistentes. 36](#_Toc35)

[Requisitos No Funcionales. 36](#_Toc36)

[Software. 36](#_Toc37)

[Hardware. 37](#_Toc38)

[Diseño de Base de Datos. 37](#_Toc39)

[Usuarios del Sistema. 38](#_Toc41)

[Diagrama de Secuencia de creación de un asistente. 38](#_Toc43)

[Diagrama de Actividades de la creación de un asistente 39](#_Toc45)

[Diseño de la interfaz del Chat para probar el asistente 40](#_Toc47)

[CAPITULO 3 . IMPLEMENTACIÓN Y PRUEBA 42](#_Toc49)

[Algoritmos Importantes 42](#_Toc50)

[Pruebas al sistema 47](#_Toc55)

[Análisis económico del costo de producción del sistema 49](#_Toc56)

[Conlusiones del capítulo 53](#_Toc62)

[CONCLUSIONES Y RECOMENDACIONES 54](#_Toc63)

[REFERENCIAS BIBLIOGRÁFICAS 55](#_Toc64)

INTRODUCCIÓN

Los asistentes virtuales se han convertido en una parte integral de la vida cotidiana en el siglo XXI. Estas herramientas de innovación han desempeñado un papel clave en la transformación de la interacción entre las empresas y los consumidores. Están diseñados para ser una forma de ayuda interactiva, capaz de proporcionar apoyo tanto a los usuarios como a los clientes.

Los chatbot tuvieron sus inicios en los años 60, simultáneamente con el procesamiento del lenguaje natural (NLP, Natural Language Processing), en un sistema de preguntas y respuestas (QA systems, question answering systems). A su vez, se denominan como un modelo característico de sistemas de conversaciones que ejecutan tareas específicas de interacción textual (Oliveira, 2018).

Hay tres partes principales en un sistema de chatbot: una parte de comprensión del lenguaje natural (NLU) que obtiene las intenciones del usuario, una parte de administración de diálogo que monitorea el sistema actual y el estado de la conversación, y una parte de generación de lenguaje natural que responde al usuario. A medida que pasaron los años y la tecnología se hizo más eficiente, los chatbots se volvieron más versátiles y utilizados, aunque el principal catalizador podría haber sido el impulso de Facebook por “Messenger Chatbots” (5 learnings from our ‘Chatbot Survey — 2017’ | by Mindbowser | Chatbots Journal, s. f.), donde se popularizó en gran medida su uso. Cabe destacar el trabajo de varias plataformas líderes en la nube para la creación de chatbots: DialogFlow de Google, wit.ai de Facebook, Microsoft LUIS, IBM Watson Conversation, Amazon Lex, SAP Conversation AI, y otras plataformas conocidas como RASA, Botsify, Chatfuel, Manychat, Flow XO, Chatterbot, Pandorabots, Botkit y Botlytics.

A partir del 2020 con la crisis desatada por el COVID19 los chatbots han emergido como una herramienta clave para la lucha global contra la pandemia, ya que permite la identificación de personas con síntomas y muchas otras enfermedades con un alto grado de precisión, además de proporcionar respuestas precisas a la mayoría de las preguntas, por medio de fuentes creíbles de información ante la gran cantidad de personas necesitadas de consejos (Chatbot: una propuesta viable para la atención al cliente en el centro de soporte de la UCI. | Caballero Ramírez | Revista Cubana de Ciencias Informáticas, s. f.).

Actualmente han evolucionado y pueden proporcionar información significativa que ayude a las empresas a mejorar su servicio al cliente. Estos utilizan un lenguaje natural para que sea más fácil para los usuarios interactuar con ellos y obtener respuestas inmediatas. A través de la Inteligencia Artificial, los chatbots pueden seleccionar la mejor respuesta entre varias posibilidades, lo que les permite aprender y mejorar con cada conversación.

Estos también han ayudado a las empresas a reducir los costos, ya que pueden manejar diferentes tareas a la vez, como la atención al cliente, el seguimiento de pedidos, el procesamiento de pagos y la recopilación de datos. Esto permite a las empresas ahorrar tiempo y dinero al reducir los costos laborales y en general se han convertido en una parte invaluable del mundo empresarial del siglo XXI.

Es importante mencionar que existen dos tipos principales de chatbots: los de dominios específicos de conocimiento y los de dominios generales.

Los chatbots de dominios específicos de conocimiento son aquellos que han sido diseñados para trabajar en un área particular de conocimiento o campo de especialización. Estos chatbots están diseñados para comprender el lenguaje y la terminología específica de ese campo y para responder preguntas y proporcionar información relacionada con ese tema en particular. Por ejemplo, un chatbot de dominio específico de conocimiento en medicina podría responder preguntas sobre síntomas, enfermedades y tratamientos médicos.

Por otro lado, los chatbots de dominios generales son aquellos que han sido diseñados para comprender el lenguaje natural en general y responder preguntas sobre una amplia variedad de temas. Estos chatbots no están especializados en un campo particular de conocimiento y pueden responder preguntas generales sobre temas como el clima, las noticias y el entretenimiento (Xu, 2020).

Los chatbots de dominios específicos de conocimiento ofrecen información precisa y detallada en un campo particular, pero pueden tener dificultades para comprender preguntas fuera de su especialización. En comparación, los asistentes virtuales como ChatGpt y otros Grandes Modelos del Lenguaje (LLM por sus siglas en inglés) son más flexibles pero presentan sesgos y errores, lo que puede afectar la precisión de los resultados y limitar su capacidad para comprender ciertos aspectos del lenguaje, además pueden ser utilizados de manera inapropiada o malintencionada, lo que puede tener graves consecuencias sociales y éticas.

Por otro lado, los chatbots de dominios generales son muy útiles cuando se trata de proporcionar respuestas rápidas y generales a una variedad de preguntas. Estos chatbots son flexibles y pueden comprender una amplia variedad de preguntas y temas. Sin embargo, debido a que no están especializados en ningún campo en particular, pueden proporcionar respuestas menos precisas y detalladas a preguntas específicas en un área de conocimiento determinada.

**Problema de investigación:**

La investigación "Herramienta digital para la construcción de conocimiento automático para un Asistente Virtual" (2022) presenta algunas limitaciones en los asistentes virtuales, como la ausencia de un chat de voz para una interacción más fluida y la incapacidad de reconocer las emociones del usuario (ej: enojo, alegría) a través del análisis facial. Estas limitaciones dificultan la interacción con los usuarios.

**Objeto de la investigación:**

Los algoritmos de procesamiento de lenguaje natural, reconocimiento de voz, análisis de video y sistemas de inteligencia artificial como sistemas de apoyo a personas con vulnerabilidades.

**Campo de Estudio:**

La elaboración de una plataforma de creación automática de asistentes virtuales con audio y video.

**Campo de Acción:**

Los asistentes virtuales como sistemas de apoyo a personas con vulnerabilidades.

**Objetivo General:**

Integrar la investigación de "Herramienta digital para la construcción de conocimiento automático para un Asistente Virtual" del año 2022 y la mejora de la experiencia del usuario en la interacción con el asistente virtual al agregar el uso del audio (poder comunicarse en lenguaje natural) y el reconocimiento de las emociones faciales para poder dar respuestas y entender las necesidades de los usuarios.

**Objetivos Específicos:**

Analizar del estado del arte.

Estudiar de la investigación "Herramienta digital para la construcción de conocimiento automático para un Asistente Virtual" del año 2022.

Estudio de tecnologías para el análisis de texto y conversión de audio a texto (p.ej., Whisper).

Modificar del algoritmo de generación de preguntas y respuestas.

Integrar un modelo grande del lenguaje para mejorar el algoritmo de creación de preguntas y respuestas.

Diseñar el proceso de creación de la plataforma.

Implementación de la plataforma.

* Probar la plataforma.

**Hipótesis:**

Si se desarrollara un software que permita la integración de RASA, Whisper y FaceAPI para la inclusión del canal de voz y reconocimiento de emociones en un sistema que permita crear asistentes virtuales, esta herramienta permitiría una mayor facilidad a la hora de la creación de asistentes, mayor precisión en el reconocimiento de voz y una mejor interpretación de las emociones de los usuarios, lo que se traducirá en una mayor eficiencia en las respuestas ofrecidas por los asistentes virtuales creados.

**Métodos de investigación:**

* Método histórico: se aplicó al realizar el análisis de otros sistemas, tecnologías o herramientas que le pudieran dar solución a la problemática en cuestión.
* Método de análisis y síntesis: se aplicó al realizar el análisis de todo el

proceso llevado a cabo en el desarrollo del proyecto y sintetizar las ideas que fueron

surgiendo; extrayendo los elementos comunes al objeto de estudio.

**Estructura del informe:**

Capítulo 1: Muestra el marco teórico, en el cual se analizan los

diferentes aspectos teóricos y su fundamentación.

Capítulo 2: Plantea la propuesta del sistema, entre otros elementos que proporciona la metodología utilizada.

Capítulo 3: Muestra la implementación del sistema, las pruebas realizadas, los

posibles resultados a obtener.

Conclusiones, Recomendaciones, Referencias Bibliográficas y Anexos.

**Aportes de la investigación:**

* Desarrollo de asistentes virtuales con los cuales se pueda establecer una comunicación mediante canales de audio y video.
* Desarrollo de una API para el sistema de creación de asistentes virtuales lo cual permitiría su integración en plataformas web.

# MARCO TEÓRICO

En esta sección se examinarán algunas de las principales herramientas de construcción de asistentes virtuales, conceptos importantes del contexto en el que se está trabajando; por último, se explicará cuál es la mejor opción para utilizar en este proyecto y por qué.

## Procesamiento del Lenguaje Natural.

Procesamiento de Lenguaje Natural (PLN), una disciplina que fusiona elementos de la Inteligencia Artificial y la Lingüística para permitir que las computadoras comprendan y se relacionen con los idiomas humanos. El propósito fundamental del PLN es cerrar la brecha entre la comunicación humana y la interacción con computadoras, proporcionando a los usuarios la capacidad de interactuar con estas últimas mediante el uso de lenguaje natural.

El PLN se desglosa en dos componentes esenciales: la Comprensión del Lenguaje Natural y la Generación del Lenguaje Natural. La Comprensión del Lenguaje Natural implica la habilidad de la computadora para comprender el lenguaje humano, abarcando aspectos como la fonología, morfología, sintaxis, semántica y pragmática. Cabe destacar que figuras prominentes como el lingüista Noam Chomsky han realizado contribuciones significativas al estudio de la sintaxis en la lingüística.

Por otro lado, la Generación del Lenguaje Natural se enfoca en la capacidad de la computadora para producir texto coherente y significativo a partir de representaciones internas. Este componente implica la creación de frases, oraciones y párrafos en lenguaje humano. (Khurana et al., 2023).

## Asistente Personal Inteligente.

Un Asistente Personal Inteligente es una herramienta tecnológica que utiliza la inteligencia artificial para interactuar con los usuarios de manera natural y realizar diversas tareas diarias. Estos asistentes pueden entender el contexto de las preguntas y peticiones del usuario, y realizar acciones cada vez más complejas y útiles, como organizar la agenda del usuario, controlar dispositivos inteligentes, o informar sobre noticias diarias. Un ejemplo de asistente de voz, que es un tipo de asistente personal inteligente, es Google Assistant. Este asistente puede interactuar con el teléfono móvil y sus aplicaciones solo con comandos de voz, convirtiendo el dispositivo en el mando de un hogar inteligente. Google Assistant puede ajustar la temperatura y la iluminación, controlar los dispositivos inteligentes del usuario incluso cuando no está en casa, ofrecer información antes de que la pida y programar recordatorios para ayudarle en el momento oportuno. Además de los asistentes de voz, existen otros tipos de asistentes virtuales que pueden ayudar al usuario en tareas cotidianas relacionadas con el manejo de dispositivos electrónicos. Estos asistentes pueden entablar una conversación natural con el usuario, entender el contexto de sus preguntas y peticiones, y llevar a cabo acciones cada vez más complejas y útiles para el usuario (Asistente Virtual, s. f.).

## Reconocimiento Automático del habla.

Desde hace tiempo ha sido de gran interés para el ser humano la posibilidad de entablar comunicación verbal con las máquinas. La ciencia ficción nos ha dado muchos ejemplos y la realidad poco a poco nos acerca a ese futuro. Para que esto sea posible, es necesario que las máquinas escuchen y entiendan nuestro lenguaje, y la primera parte de ese proceso es el reconocimiento automático del habla.

Los sistemas de reconocimiento automático del habla (automatic speech recognition o speech-to-text) son aquellos que traducen expresiones y enunciados de su forma hablada a texto. Este reconocimiento no conlleva la interpretación y entendimiento del significado del texto, ya que esta tarea es realizada posteriormente, en caso de que la aplicación lo requiera, mediante técnicas de Procesamiento de Lenguaje Natural.

El reconocimiento automático del habla (ASR, Automatic Speech Recognition por sus siglas en inglés) es un proceso en el que un sistema convierte la señal acústica de una expresión hablada en texto escrito. El proceso utiliza un enfoque probabilístico para asignar una puntuación a cada secuencia de palabras candidatas, evaluando su propiedad fonética, conocimiento lingüístico, contigüidad de las palabras y gramática para seleccionar la mejor opción. El esquema típico del proceso de ASR incluye preprocesamiento, extracción de rasgos característicos, decodificación y posprocesamiento del resultado. El preprocesamiento incluye la segmentación de la señal, la reducción del ruido de fondo y la identificación de las partes de silencio entre frases. La extracción de rasgos característicos implica la división de la señal acústica en ventanas de tiempo y la aplicación de transformaciones matemáticas, filtros y procesos de normalización para obtener un vector de coeficientes representativo de la señal. En la etapa de decodificación, se calcula la secuencia de palabras más probable de corresponder a la señal representada por los vectores de rasgos característicos. (Sobrino, 2020).

## Estado del arte de las herramientas.

Actualmente, existen diversas herramientas y plataformas para la creación de chatbots y asistentes virtuales. En este estado del arte se presentan cuatro de las más utilizadas: Rasa marco de trabajo, DialogFlow, IBM Watson y Amazon Lex.

### DialogFlow



*Figura 1. Logotipo de DialogFlow*

DialogFlow, propiedad de Google, es otra herramienta popular para la creación de chatbots y asistentes virtuales. Ofrece una interfaz visual amigable para la creación de flujos de conversación y una gran cantidad de integraciones con otras herramientas y plataformas. DialogFlow también cuenta con la capacidad de integrar soluciones de inteligencia artificial y aprendizaje automático de Google, como Google Cloud Speech-to-Text y Google Cloud Translate (Dialogflow de Google cambia su modelo de pricing, 2017).

### Amazon Lex



*Figura 2. Logotipo de Amazon Lex*

Amazon Lex es un servicio de Amazon Web Services que permite a los desarrolladores construir interfaces de conversación para aplicaciones y bots. Con Amazon Lex, los desarrolladores pueden construir aplicaciones que pueden entender y responder a las solicitudes de los usuarios en lenguaje natural. Los bots de Amazon Lex pueden entender las entradas de los usuarios proporcionadas con texto o voz y conversar en lenguaje natural.

Amazon Lex permite a los desarrolladores crear bots que pueden realizar tareas automatizadas, como pedir una pizza, reservar un hotel, pedir flores, etc. Un bot de Amazon Lex cuenta con capacidades de reconocimiento automático de voz (ASR) y comprensión del lenguaje natural (NLU). Cada bot debe tener un nombre único en su cuenta.

Los bots de Amazon Lex están diseñados para una interacción de solicitud y respuesta o una conversación en streaming continua. Mediante la interacción de solicitud y respuesta, cada entrada (voz o texto) del usuario se procesa como una llamada API independiente. En una conversación en streaming, todas las entradas del usuario en diferentes turnos se procesan como una sola llamada a la API.

Amazon Lex admite varios idiomas, lo que permite a los desarrolladores crear bots que pueden interactuar con usuarios de todo el mundo.

En términos de costos, con Amazon Lex, no hay costos por anticipado ni cuotas mínimas. Solo se le cobrará por las solicitudes de texto o voz que se realicen. Los precios del pago por uso y el bajo costo de cada solicitud hacen que el servicio sea un método rentable para crear interfaces de conversación (Cómo Amazon Lex cionamiento cionamiento cionamiento cionamiento cionamiento cionamiento cionamiento cionamiento - Amazon Lex V1, s. f.)

### IBM Watson



*Figura 3. Logotipo de IBM Watson*

IBM Watson Assistant es una plataforma líder en el mercado, de inteligencia artificial conversacional diseñada para ayudarte a superar la fricción del soporte tradicional y ofrecer experiencias excepcionales a prospectos, clientes y empleados.

Impulsado por modelos de lenguaje de gran envergadura (LLMs) en los que puedes confiar y una interfaz de usuario intuitiva, Watson Assistant permite a tus equipos crear agentes de voz y chatbots impulsados por inteligencia artificial que brindan soporte automatizado de autoservicio a través de todos los canales y puntos de contacto, con una integración perfecta con las herramientas que impulsan tu negocio (IBM Watsonx Assistant Virtual Agent, s. f.).

### RASA marco de trabajo



*Figura 4. Logotipo de RASA*

Rasa es un marco de aprendizaje automático de código abierto utilizado para crear chatbots conversacionales. Se utiliza para automatizar asistentes de texto y voz.

Es capaz de mejorar las interacciones con los clientes y, por lo tanto, ayuda a las empresas. Se puede utilizar para abordar aspectos muy cruciales de las empresas, como la contratación, donde la inteligencia artificial se puede utilizar para construir chatbots que ofrezcan un excelente proceso de entrevista al proporcionar una gran experiencia al candidato. Para las empresas, los clientes que visitan su sitio web tienen una gran experiencia al conversar con una IA. La lista de implicaciones podría ser extensa.

Se compone de 2 partes:

• RASA NLU: Esta parte se encarga del lenguaje natural, para ello se nutre de los idiomas disponibles en la librería spaCy. Su misión es realizar el procesado del mensaje y su posteriormente transformarlo en datos con estructura concreta.

• RASA Core: Esta parte se encarga de la gestión del diálogo, se conecta con RASA NLU. El componente decide qué acciones tomar en cada momento (AskLua, 2020) .

### BotPress



*Figura 5. Logotipo de Botpress*

 Botpress es una herramienta que simplifica el proceso de creación de chatbots para los desarrolladores. La plataforma reúne el código y la infraestructura necesarios para poner en marcha un chatbot y ofrece una plataforma completa y fácil de usar que incluye todas las herramientas necesarias para crea, desplegar y gestionar chatbots en tiempo récord (Primeros pasos con Botpress | Botpress Blog, s. f.).

La plataforma incluye:

* Tareas integradas de procesamiento del lenguaje natural, como reconocimiento de intenciones, corrección ortográfica, extracción de entidades y etiquetado de ranuras (y muchas otras).
* Un visual conversation studio para diseñar conversaciones y flujos de trabajo multiturno
* Un emulador y un depurador para simular conversaciones y depurar tu chatbot
* Compatibilidad con canales de mensajería populares como Slack, Telegram, MS Teams, Facebook Messenger y un chat web integrable.
* Un SDK y un editor de código para ampliar las funciones
* Herramientas posteriores a la implementación, como paneles de [análisis de chatbot](https://botpress.com/es/blog/chatbot-analytics), traspaso humano y [más](https://botpress.com/es/pricing)...

Al analizar las opciones, RASA es la mejor elección para este proyecto. DialogFlow tiene una versión gratuita, pero limita las solicitudes del usuario y requiere administración en la propia plataforma. Amazon Lex es descartado porque los asistentes deben desplegarse en Amazon Web Services, lo cual es gratuito por un año, pero luego requiere una tarjeta de crédito extranjera. IBM Watson es costoso, tiene documentación limitada y solo interactúa en inglés. Botpress, aunque fácil de aprender e integra con ChatGPT, no permite un host personalizado para el asistente y cobra después de 1000 solicitudes mensuales gratuitas.

## Herramienta de Reconocimiento automático del Habla y Transcripción de Voz a Texto.

Whisper es una herramienta de inteligencia artificial diseñada para convertir el habla en texto. Es una solución innovadora que ha revolucionado el campo de la transcripción, ya que puede manejar una amplia variedad de idiomas y ofrece un alto nivel de precisión en el reconocimiento de voz .

Este se define como un "sistema de reconocimiento automático de voz (ASR)" y ha sido entrenado con más de 680.000 horas de datos. Este modelo puede accederse a través de la web o de forma local y es abierto a todos los usuarios (He usado Whisper para transcribir una entrevista: es la herramienta que llevaba esperando desde hace años, s. f.). Además, Whisper puede trabajar con más de 100 idiomas y cuenta con varios modelos de diferentes tamaños para su uso en servidores o computadoras con menos recursos, es clave destacar que mientras más grande sea el modelo, más consumo de recursos tendrá pero tendrá una mayor precisión.

En comparación con otras herramientas de transcripción, Whisper destaca por su capacidad para manejar acentos, ruido de fondo y lenguaje técnico. Además, ha demostrado ser especialmente efectivo para la traducción de voz a texto, superando incluso a los modelos que se especializan en el rendimiento de LibriSpeech (¿Qué es y cómo funciona Whisper? | Código Espagueti, s. f.).

Después del análisis, se elige Whisper para transcribir audio a texto en este proyecto. Whisper se integra fácilmente con Python, ofrece alta precisión en transcripciones en varios idiomas, es de código abierto y muy sencillo de implementar.

## Lenguajes de Pogramación utilizados.

### Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general que se destaca por su simplicidad y legibilidad, lo que lo hace especialmente adecuado para principiantes en programación. Fue creado por Guido van Rossum y se lanzó por primera vez en 1991 (¿Qué es Python?, s. f.).

Python es versátil y se utiliza en una variedad de aplicaciones. Algunas de las principales aplicaciones de Python incluyen:

* **Desarrollo web**: Python se utiliza a menudo para desarrollar el back-end de un sitio web o una aplicación. Este proceso puede incluir el envío de datos hacia y desde los servidores, el procesamiento de datos y la comunicación con las bases de datos, el enrutamiento de URL y la garantía de seguridad. Python ofrece varios marcos de trabajo para el desarrollo web, los más utilizados son Django y Flask (Londoño, s. f.)
* **Desarrollo de software**: Python se implementa en el desarrollo de software desde que se lanzó y continúa siendo muy popular para ese propósito. Se utiliza para crear software en diferentes plataformas porque es compatible con muchos ámbitos de operación, sistemas operativos para computadoras y dispositivos móviles y en entornos de sistema en tiempo de ejecución (Londoño, s. f.) .
* **Ciencias de blockchain**: Python ha sido un usuario importante de la industria de blockchain, manejando desde siempre ciencias de blockchain. Su habilidad para manejar todos los aspectos de la tecnología blockchain lo convierte en un elemento primordial (Londoño, s. f.)
* **Aprendizaje automático y ciencia de datos**: Python es uno de los lenguajes de programación más populares en la ciencia de datos y el aprendizaje automático debido a su sintaxis clara y legible, y a su amplia gama de bibliotecas y marcos de trabajo especializados en estas áreas, como NumPy, Pandas, Matplotlib, Scikit-learn y TensorFlow (Londoño, s. f.)

La elección de este lenguaje para la programación del servidor y de los asistentes se basa en que RASA esta construido sobre él, también este es el lenguaje con el que se desarrolla el procesamiento del lenguaje natural.

### Marco de Trabajo de Interfaz de Usuario.

**React** 18 es una biblioteca de JavaScript para la construcción de interfaces de usuario, desarrollada por Facebook. React permite a los desarrolladores crear componentes reutilizables, lo que facilita el mantenimiento y la escalabilidad de las aplicaciones. React también utiliza un DOM virtual para mejorar el rendimiento de las aplicaciones. (React vs Vue vs Angular vs Svelte, 2020)

**Angular** 17 es una plataforma de desarrollo para la construcción de aplicaciones web móviles y de escritorio utilizando TypeScript/JavaScript y otros lenguajes. Angular fue desarrollado por Google y es conocido por su enfoque en la creación de aplicaciones de una sola página (SPA). Angular también proporciona una estructura completa para las aplicaciones, incluyendo enrutamiento, plantillas y pruebas de utilidad (React vs Vue vs Angular vs Svelte, 2020)

**Vue** 3 es un marco de trabajo de JavaScript que se centra en la facilidad de uso y la flexibilidad. A diferencia de Angular y React, Vue no está respaldado por una empresa de alto nivel, pero ha ganado una increíble popularidad en la comunidad de código abierto. Vue es conocido por su curva de aprendizaje más sencilla y su balance entre las fortalezas de otros marcos de trabajo y menos de sus debilidades. (Comparing React, Angular, Vue, and Svelte - Accelebrate, s. f.)

**Svelte** es una librería de JavaScript que compila el código a JavaScript en tiempo de compilación en lugar de en tiempo de ejecución. Esto resulta en aplicaciones más pequeñas y más rápidas. Svelte es conocido por su simplicidad y su enfoque en la eficiencia (Comparing React, Angular, Vue, and Svelte - Accelebrate, s. f.)

Para este proyecto se elige el desarrollo de la aplicación web con React JS, debido a que su enfoque en la creación de componentes reutilizables puede hacer que el desarrollo de aplicaciones sea más eficiente. También cabe resaltar que, gracias al virtual DOM, puede renderizar componentes extremadamente rápido, superando a Angular y a Vue en este aspecto, lo cual es importante debido a los requisitos de este proyecto. Por último, gracias a su alta popularidad, la mayoría de las librerías existentes están disponibles para su uso en React, como las librerías de reconocimiento facial y de habla utilizadas.

### Marco de trabajo del servidor

Al elegir un marco de trabajo, es importante tener en cuenta si se prefiere trabajar con una guía definida o tener más libertad en la configuración. También es importante tener en cuenta otras métricas, como la velocidad de respuesta, el soporte a la carga de peticiones y la concurrencia. Sin embargo, no existe un marco de trabajo que sea el mejor, ya que existen factores subjetivos como la productividad y las mejores prácticas que hacen imposible medir y comparar de manera objetiva (Hernández, 2023).

Para la elección, se tuvo en cuenta que el lenguaje a utilizar sería Python. Se realizó una investigación de cuál sería el más adecuado.

**Django**

Es un marco de trabajo de desarrollo web de alto nivel, de código abierto, escrito en Python. Es popular por su versatilidad, productividad y seguridad. Algunas de las empresas más grandes del mundo, como Instagram, Disqus, y la NASA, utilizan Django para sus proyectos (Qué es Django y por qué usarlo, 2018)

Ventajas:

El Django REST marco de trabajo es un marco de trabajo popular para APIs en Python. Ofrece una arquitectura personalizable y modular, ideal para APIs simples y complejas. Viene con operaciones CRUD y un navegador incorporado para probar los puntos finales de la API. Es especialmente útil para proyectos de Machine Learning, ya que Python es el lenguaje de programación estándar para los científicos de Machine Learning. Además, Django puede manejar grandes cantidades de tráfico y proporcionar uso de API a más de 450 millones de usuarios, lo que lo hace escalable y adecuado para soluciones empresariales de gran volumen (Team, 2022)

Desventajas:

Aunque es fácil de aprender, puede ser difícil de dominar debido a su gran cantidad de configuraciones y ajustes que se deben realizar. No es recomendable para proyectos pequeños debido a su intensivo uso de recursos del servidor . Además, es un marco de trabajo monolítico que no proporciona mucha flexibilidad en la arquitectura del proyecto, lo que puede ser un inconveniente para este proyecto (Team, 2022)

Flask

Es un micromarco de trabajo para Python que permite a los desarrolladores construir aplicaciones web de manera rápida y fácil. Fue desarrollado por Armin Ronacher, líder del Grupo Internacional de Pythonistas (POCCO), y se basa en el kit de herramientas WSGI y el motor de plantillas Jinja2. Es un marco de trabajo altamente usado, respaldado por el uso de compañias como Samsung, Netflix, Uber y Airbnb lo cual es un motivo de peso para tomarlo en cuenta al elegir un marco de trabajo para la creación del servidor de este proyecto (Condez, 2022)

Ventajas:

Es un marco de trabajo web que es fácil de entender, lo que lo hace ideal para principiantes. Su simplicidad permite a los desarrolladores tener un control total sobre el desarrollo web, tomando el control creativo de la aplicación y el desarrollo web. Aunque es simple, ofrece muchas características interesantes para usar en el marco. Además, viene con un motor de plantillas que permite usar la misma interfaz de usuario para múltiples páginas. Python puede insertar variables en las plantillas. También permite la prueba unitaria a través de su soporte integrado, servidor de desarrollo incorporado, depurador rápido y despacho de solicitudes RESTful. Es ligero para permitir transitar fácilmente a un marco de trabajo web con algunas extensiones (Condez, 2022).

Desventajas:

Este marco de trabajo es amigable para principiantes y aficionados, lo que puede llevar a un desarrollo de código de baja calidad. Maneja cada solicitud una tras otra, lo que puede ralentizar el servicio de múltiples solicitudes. La inclusión de módulos adicionales puede aumentar el riesgo de seguridad si se incluye un módulo malicioso. Aunque se pueden mitigar estos problemas utilizando un hosting especializado en Python, es crucial tener en cuenta estos factores al elegir Flask para el desarrollo web (Condez, 2022)

**FastAPI**

Es un marco de trabajo web de alto rendimiento para Python. Su rendimiento puede compararse con NodeJS y Go, y se considera uno de los marcos de trabajo más rápidos de Python disponibles (Build And Host Fast Data Science Applications Using FastAPI | by Farhad Malik | Towards Data Science, s. f.). Este marco de trabajo es conocido por su simplicidad y velocidad, lo que lo hace ideal para la construcción de APIs rápidas y eficientes.

Veloz. Solo esa palabra lo define. Es (y no solo en Python, sino hablando en comparación a todos los otros lenguajes de programación) una de las herramientas más veloces para construir un servidor, compitiendo con lenguajes como Go o Node.js

Algunas empresas que utilizan FastAPI incluyen a Netflix, Uber, y Microsoft [25]

Ventajas:

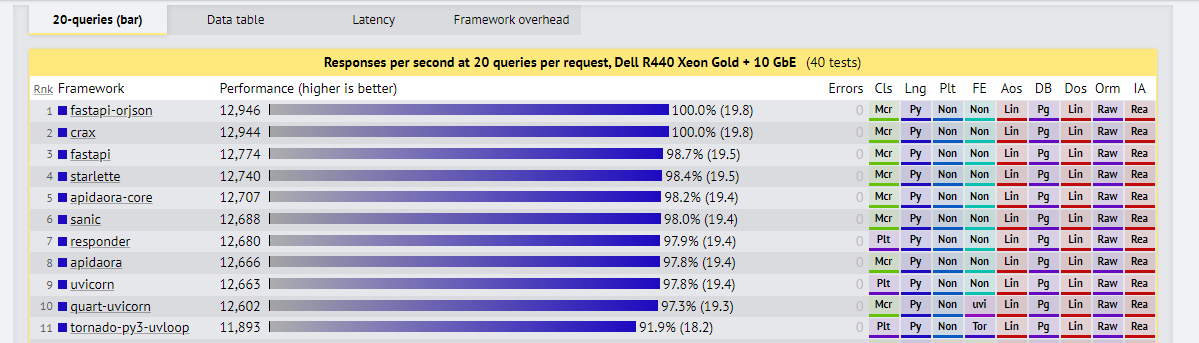
FastAPI es un marco de trabajo moderno y de alto rendimiento para construir APIs con Python 3.6+. Destaca por su documentación automática, que genera una interfaz interactiva para explorar y probar la API, y por el uso de declaraciones de tipos de Python 3.6 para proporcionar un soporte de editor excelente y autocompletado. Además, FastAPI incorpora soporte para seguridad y autenticación, lo cual es esencial para una API (Explicación de FastAPI en 5 minutos o menos - Geekflare, s. f.).

Desventajas:

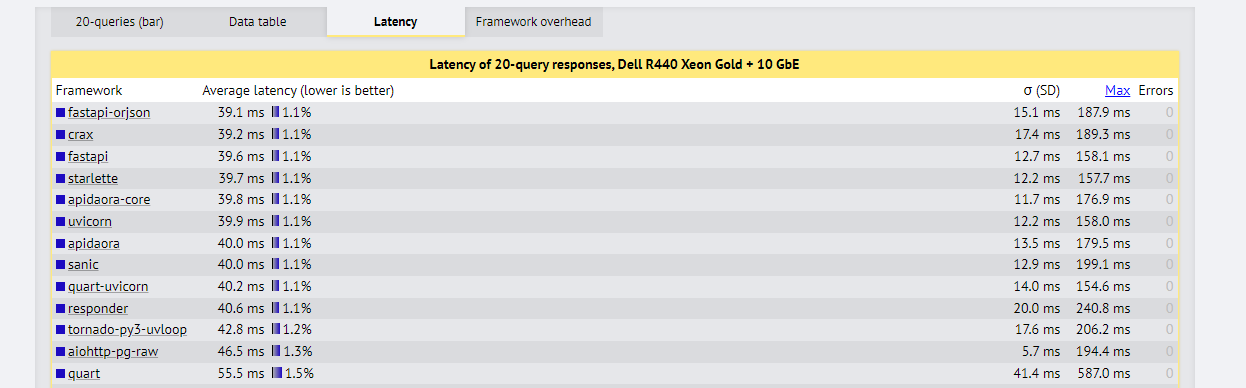
Pese a su gran robustez, es un marco de trabajo relativamente nuevo, dado esto es más propenso a que no se use con frecuencia. FastAPI aún está en desarrollo y hay muchas cosas que se pueden mejorar. Aunque su comunidad está creciendo, todavía no es comparable a las comunidades de otros marco de trabajos (Explicación de FastAPI en 5 minutos o menos - Geekflare, s. f.).

**Análisis de los marco de trabajos.**

A continuación se muestra una prueba de Benchmarking realizada en el año 2022 la cual consiste en realizar 20 consultas a un servidor por segundo (TechEmpower Web Framework Performance Comparison, 2023)



FastApi demuestra su alta velocidad en el manejo de concurrencia, ligereza y escasa latencia al realizar consultas, como se muestra en el primer y tercer lugar.



En esta imagen se puede apreciar la tasa de latencia al realizar las 20 consultas las cuales posicionan a FastAPI en primer y tercer lugar al igual que en la imagen anterior.

Luego de realizar un análisis crítico sobre las ventajas y desventajas de cada uno de los marco de trabajos y de revisar la prueba de punto de referencia que se plasmó previamente, se llega a la conclusión de que el marco de trabajo que cumple con todos los requisitos para desarrollar el servidor de la aplicación que generará los asistentes virtuales es FastAPI. Esto se debe a que la aplicación necesita tener un rendimiento óptimo en múltiples consultas en tiempo real, debe ser una solución sencilla con una mínima cantidad de configuraciones, y debe ser flexible y fácil de escalar para futuras mejoras.

### Langchain



*Logotipo de Langchain*

LangChain es una herramienta de código abierto que simplifica la creación de aplicaciones usando modelos de lenguaje a gran escala. Estos modelos, entrenados con vastos conjuntos de datos, pueden responder preguntas de usuarios o crear imágenes a partir de texto. LangChain mejora la personalización y relevancia de la información generada, permitiendo a los desarrolladores crear o adaptar cadenas de solicitudes y proporcionando componentes que facilitan el acceso de los modelos a nuevos datos sin necesidad de volver a entrenarlos. Destaca por su capacidad de ajustar los modelos de lenguaje a contextos empresariales específicos, optimizando así el desarrollo de aplicaciones que respondan a datos organizacionales (Topsakal & Akinci, 2023).

La elección de Langchain para este proyecto se debe a que es una herramienta que permitirá mantener una mejor comunicación con el modelo de lenguaje que se utilizará y facilitará el proceso de creación de indicaciones para que el modelo genere las preguntas correctamente. Además, su enfoque modular y su capacidad para manejar el contexto y la memoria a lo largo de interacciones multiturno mejoran la coherencia y relevancia de las respuestas generadas.

### Base de datos Vectorial.

Una base de datos vectorial es un tipo especializado de base de datos diseñada para almacenar y gestionar datos en forma de vectores multidimensionales, que representan características o cualidades de los datos. Estos vectores pueden ser el resultado de transformaciones de datos complejos, como texto, imágenes, audio o video, utilizando técnicas como modelos de aprendizaje automático, incrustaciones de palabras o técnicas de extracción de características. La principal ventaja de las bases de datos vectoriales es su capacidad para localizar y recuperar datos de manera rápida y precisa basándose en la proximidad o similitud de los vectores, permitiendo búsquedas basadas en relevancia semántica o contextual en lugar de depender únicamente de coincidencias exactas o criterios específicos (Han et al., 2023)

Qdrant es una base de datos vectorial y una herramienta para realizar búsquedas de similitud de vectores. Ofrece un servicio listo para producción con una API fácil de usar para buscar, almacenar y gestionar vectores. Qdrant es versátil y se adapta bien para coincidencias basadas en semántica o redes neuronales, soportando una amplia gama de criterios de consulta y tipos de datos (Singh et al., 2023).

Faiss es una biblioteca de Facebook AI Research (FAIR) diseñada para la búsqueda de vectores de alta dimensión. Aunque no se menciona explícitamente en los resultados de búsqueda, Faiss es conocida por su eficiencia en la búsqueda de vecinos más cercanos y su capacidad para manejar grandes volúmenes de datos de manera eficiente. Sin embargo, Faiss es más una biblioteca que una base de datos completa, lo que significa que puede requerir más trabajo de integración y gestión de datos en comparación con una base de datos vectorial dedicada (Douze et al., 2024).

Chroma es una base de datos vectorial de código abierto diseñada específicamente para almacenar y recuperar incrustaciones vectoriales. Su función principal es almacenar incrustaciones con metadatos asociados para su uso posterior por modelos de lenguaje extensos. Chroma se destaca por simplificar el proceso de creación de aplicaciones de procesamiento de lenguaje natural (NLP) al hacer que el conocimiento, los hechos y las habilidades sean pluggables para modelos de aprendizaje automático a gran escala. Además, ofrece capacidades sólidas como filtrado, agrupación inteligente y relevancia de consulta, lo que la hace ideal para aplicaciones de generación de preguntas y respuestas basadas en contexto (Dhungana, 2023).

Debido a su capacidad para manejar datos complejos como texto y su automatización de tareas relacionadas con la gestión de inserciones, ChromaDB es la mejor opción para este proyecto. Estas características, junto con su escalabilidad y rendimiento, lo hacen ideal para almacenar y recuperar información relevante de el documento para generar preguntas

y respuestas basadas en un contexto específico.

### Herramienta para el reconocimiento de rostros.

La biblioteca face-api.js es una herramienta de JavaScript que permite la detección y reconocimiento de rostros en el navegador. Esta biblioteca se construye sobre la API central de tensorflow.js y es compatible con la detección y reconocimiento de rostros, la identificación de expresiones faciales, la determinación de la edad y el género (10 principales API de Reconocimiento Facial - Infoteknico, 2023).

En términos de alternativas, existen varias API de reconocimiento facial disponibles, incluyendo las ofrecidas por Amazon, Microsoft, Google, IBM, Face++ de Megvii y Clarifai [29] Estas API ofrecen una amplia gama de funciones para la detección, verificación y reconocimiento facial.

En comparación con las alternativas, face-api.js se destaca por tener un modelo entrenado previamente, lo que puede facilitar su implementación y uso, además hay que tener en cuenta que los datos no quedan almacenados ni usados por terceros con Face Api y otro punto importante es que es una libreria gratuita a diferencia de las otras tecnologías mencionadas, que algunas pese a que tienen una face gratuita, luego tienen limitaciones o cobros por servicio.

## LLM a utilizar en el algoritmo de generación de preguntas y respuestas

**Llama 2.**

Llama 2 es un modelo de lenguaje de código abierto desarrollado por Meta. Es conocido por su eficiencia y velocidad, y es una opción atractiva para ciertas instancias en las que se necesita un modelo de generación de lenguaje. A pesar de ser un modelo más pequeño que otros modelos de lenguaje como GPT-3.5 y GPT-4, Llama 2 ha demostrado un rendimiento sólido en varios benchmarks (pruebas de referencia), lo que lo convierte en una opción viable para ciertas aplicaciones (Luzniak, 2023).

Llama 2 es especialmente útil para pequeñas empresas que buscan construir un asistente virtual, gracias a su naturaleza de código abierto y su adaptabilidad, que permite personalizaciones que se ajusten a las necesidades específicas de un negocio .

**GPT-3.5**

GPT-3.5, lanzado en marzo de 2022, es una versión actualizada de GPT-3. Es conocido por su excelente rendimiento en conversaciones complejas y su amplia compatibilidad con idiomas, lo que lo hace ideal para empresas con una base de clientes global (Llama 2 Vs GPT-3.5 Vs GPT-4, 2023).

**Falcon**

Falcon es otro modelo de lenguaje desarrollado por Meta. En comparación con Llama 2, Falcon ha mostrado un rendimiento inferior en la mayoría de los benchmarks, aunque su rendimiento específico puede variar dependiendo del caso de uso específico (Luzniak, 2023).

**GPT-4**

GPT-4 es el modelo de lenguaje más reciente de la familia GPT, desarrollado por OpenAI. Es conocido por superar a Llama 2 y GPT-3.5 en el benchmark 5-shot MMLU, lo que lo convierte en la mejor opción para tareas complejas y "misionales" que requieren un alto nivel de creatividad (Luzniak, 2023) .

**PaLM**

PaLM es una familia de modelos de lenguaje fundamentales desarrollada por Google. Según Google, PaLM 2 puede codificar, traducir y "razonar" de una manera que supera a GPT-4. Google ya utiliza PaLM 2 para alimentar 25 productos, incluyendo su asistente de IA conversacional Bard (The AI race heats up: Google announces PaLM 2, its answer to GPT-4 | Ars Technica, s. f.).

Se podría pensar que la elección de Llama se basa en que es Open Source, pero no, la realidad es que Llama frente a modelos como GPT-3.5, PALM y FALCON, tiene un rendimiento bastante adecuado y las métricas que se muestran a continuación lo demuestran.

Al comparar Llama 2 y Falcon, como se muestra en la foto, en los modelos de 7 billones de parámetros, Llama 2 tiene casi el doble de puntuación en MMLU. Esta métrica mide cuán bien un modelo de lenguaje a gran escala comprende el lenguaje y resuelve problemas usando el conocimiento adquirido durante su entrenamiento.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modelo | Cantidad de Parámetros | Puntuación en tareas de Código | Razonamiento común | Conocimiento del Mundo | Comprensión lectora |
| MPT | 7 Mil millones 30 Mil millones | 20,5 28,9 | 57,4 64,9 | 41,0 50,0 | 57,5 64,7 |
| Falcon | 7 Mil millones 40 Mil millones | 5,6 15,2 | 56,1 69,2 | 42,8 56,7 | 36,0 65,7 |
| Llama 1 | 7 Mil millones 13 Mil millones 33 Mil millones 65 Mil millones | 14,1 18,9 26,0 30,7 | 60,8 66,1 70,0 70,7 | 46,2 52,6 58,4 60,5 | 58,5 62,3 67,6 68,6 |
| Llama 2 | 7 Mil millones 13 Mil millones 34 Mil millones 70 Mil millones | 16,8 24,5 27,8 37,5 | 63,9 66,9 69,9 71,9 | 48,9 55,4 58,7 63,6 | 61,3 65,8 68,0 69,4 |

Tabla comparativa entre los modelos de Falcon y Llama (Tsang, 2023)

En la próxima imagen, se observa como pese a que el modelo Llama 2 70 billones pese a ser el que menor puntuación tiene de sus contrincantes PALM 1, PALM 2, GPT-3.5, GPT-4, se muestra muy cercano en sus métricas a estos modelos.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Marcos de Prueba (Intentos) | GPT-3,5 | GPT-4 | PaLM | PaLM-2-L | Llama 2 |
| MMLU (5-intentos) | 70.0 | 86.4 | 69.3 | 78.3 | 68.9 |
| TriviaQA (1-intento) | - | - | 81,4 | 83,1 | 85 |
| Preguntas Naturales (1-intento) | - | - | 29,3 | 37,5 | 33 |
| QSM8K (8-intentos) | 57,1 | 92 | 56,5 | 80,7 | 56,8 |
| EvalHumana (0-intentos) | 48,1 | 67 | 26,2 | - | 29,9 |
| BBH (3-intentos) | - | - | 52,3 | 65,7 | 51,2 |

Tabla comparativa entre Llama2, PaLM, PaLM2, GPT-3.5 y GPT-4

Se elige a Llama 2 para desarrollar esta tesis porque permite ajustes para mejorar la precisión en la generación de preguntas y respuestas, y se puede ejecutar localmente sin necesidad de una GPU. Estos aspectos son cruciales ya que otros modelos no permiten ajustes debido a su licencia o son costosos en términos de recursos de hardware.

### Base de Datos utilizada.

Para este proyecto, elegimos MongoDB, una base de datos NoSQL, por varias razones. En primer lugar, su capacidad de manejar grandes volúmenes de datos y tráfico gracias a su capacidad de distribución y sharding automático (Postgres vs. MongoDB, 2023). Además, su modelo basado en documentos permite almacenar datos de diferentes estructuras en la misma colección, lo cual es beneficioso para un proyecto de asistentes virtuales con diversas necesidades de datos (MongoDB vs PostgreSQL: 15 Critical Differences, s. f. -a). En términos de rendimiento, MongoDB utiliza BSON para almacenar datos, lo que agiliza la lectura y escritura, y la creación de índices en campos y subdocumentos mejora la velocidad de las consultas (MongoDB Vs PostgreSQL, s. f.). Ofrece transacciones multi-documento, útiles para operaciones que involucran múltiples asistentes virtuales o usuarios (MongoDB Vs PostgreSQL, s. f.). Además, es más económico que PostgreSQL, especialmente al escalar el proyecto, y su opción de nube, MongoDB Atlas, facilita la gestión de la base de datos (MongoDB vs PostgreSQL: 15 Critical Differences, s. f. -b). Finalmente, su sintaxis de consulta es más intuitiva y amigable en comparación con SQL, lo que simplifica el desarrollo del proyecto.

### Entorno de desarrollo.

**Visual Studio Code.**

Visual Studio Code (también llamado VS Code) es como la versión mini de Visual Studio. Es un editor de texto ligero y de código abierto disponible en Windows, Mac y Linux. También está disponible la versión web.

VS Code viene con soporte integrado para JavaScript, TypeScript y Node JS, pero puedes usarlo para codificar en cualquier idioma que desees. Todo lo que necesitas hacer es descargar las extensiones correspondientes.

Como es compatible con JavaScript, TypeScript y Node JS de forma predeterminada, también obtienes un depurador y detección inteligente (intellisence). Pero para obtener detección inteligente, un compilador y depuradores para otros lenguajes, debe descargar las extensiones correspondientes (Visual Studio vs Visual Studio Code, 2023).

Será de gran utilidad para el desarrollo del proyecto gracias a su alto nivel de personalización y facilidad a la hora de el desarrollo, permitiendo la integracion de herramientas como GIT para el control de versiones y permitiendo desarrollar en el entorno del servidor y la interfaz gráfica sin tener que cambiar de entorno de desarrollo.

### Sitema de Control de Versiones GIT.

Git es un sistema de control de versiones distribuido que permite a los desarrolladores trabajar juntos en proyectos de software. Fue creado por Linus Torvalds, el creador de Linux, y es ampliamente utilizado en el desarrollo de software. Git rastrea los cambios en el código fuente a lo largo del tiempo, lo que permite a los desarrolladores volver a versiones anteriores del código si es necesario. También facilita la colaboración entre desarrolladores al permitirles trabajar en el mismo código base de manera simultánea (Qué es Git | Atlassian Git Tutorial, s. f.)

### XP

La Metodología de Desarrollo de Software XP, también conocida como Programación Extrema, es una metodología ágil que se centra en la entrega de software de alta calidad en un tiempo reducido. Fue creada por Kent Beck en 1999 y se basa en la adaptabilidad y la comunicación constante entre el equipo de desarrollo y el cliente (Mancuzo, 2020).

La metodología XP se compone de varias fases y roles clave:

1. Planificación: Se identifican las historias de usuario, que son tarjetas donde se detallan las funcionalidades específicas del software a desarrollar. Estas historias se descomponen en mini-versiones y se priorizan según su importancia (Metodología XP o Programación Extrema, s. f.).
2. Diseño: En esta fase, se realiza el diseño del software, buscando hacer un código sencillo que sea lo mínimo necesario para que funcione. Para un diseño de software orientado a objetos, se crean tarjetas CRC (Clase-Responsabilidad-Colaboración) (Mancuzo, 2020)
3. Codificación: La programación se realiza en parejas en frente del mismo ordenador, asegurando que se realice un código más universal. Esto ayuda a garantizar que cualquier otro programador pueda trabajar y entender el código (Metodología XP o Programación Extrema, s. f.).
4. Pruebas: El código de una función se somete a una serie de pruebas unitarias continuas, con el objetivo de corregir fallas periódicamente (Mancuzo, 2020).
5. Lanzamiento: Si se han seguido de forma correcta las etapas anteriores, se ha logrado estructurar un software que cumple con las expectativas del cliente (Mancuzo, 2020)

La elección de esta metodología se basa en que permite una comunicación constante entre el equipo y el cliente, es altamente adaptable a los cambios como podrían ser nuevos requisitos en el proyecto, cambios de diseño y etc., tiene un alto enfoque en los resultados y por último se centra en entregar constantemente software de alta calidad.

Conclusiones del capítulo.

En el marco teórico de la tesis, se exploran y comparan diversas tecnologías y metodologías para respaldar el desarrollo de un proyecto de asistentes virtuales. Se destacan dos marcos de trabajo web, Flask y FastAPI, con sus respectivas ventajas y desventajas, optando finalmente por FastAPI debido a su alto rendimiento y documentación automática. Para el modelado de lenguaje, se elige Llama 2 por su eficiencia y adaptabilidad, aunque se consideran también GPT-3.5 y GPT-4 por su excelencia en conversaciones complejas. MongoDB se selecciona como la base de datos NoSQL ideal, gracias a su capacidad de manejar diversos tipos de datos y su rendimiento. En cuanto a las herramientas de desarrollo, Visual Studio Code y Git se presentan como opciones sólidas para la escritura de código y el control de versiones, respectivamente. Finalmente, se adopta la Metodología de Desarrollo de Software XP por su enfoque en la entrega rápida y de alta calidad, priorizando la comunicación constante y la adaptabilidad. Este análisis exhaustivo proporciona una base sólida para la implementación exitosa del proyecto de asistentes virtuales, asegurando un rendimiento óptimo, escalabilidad y calidad en todas las etapas del desarrollo.

# ORGANIZACIÓN Y DISEÑO

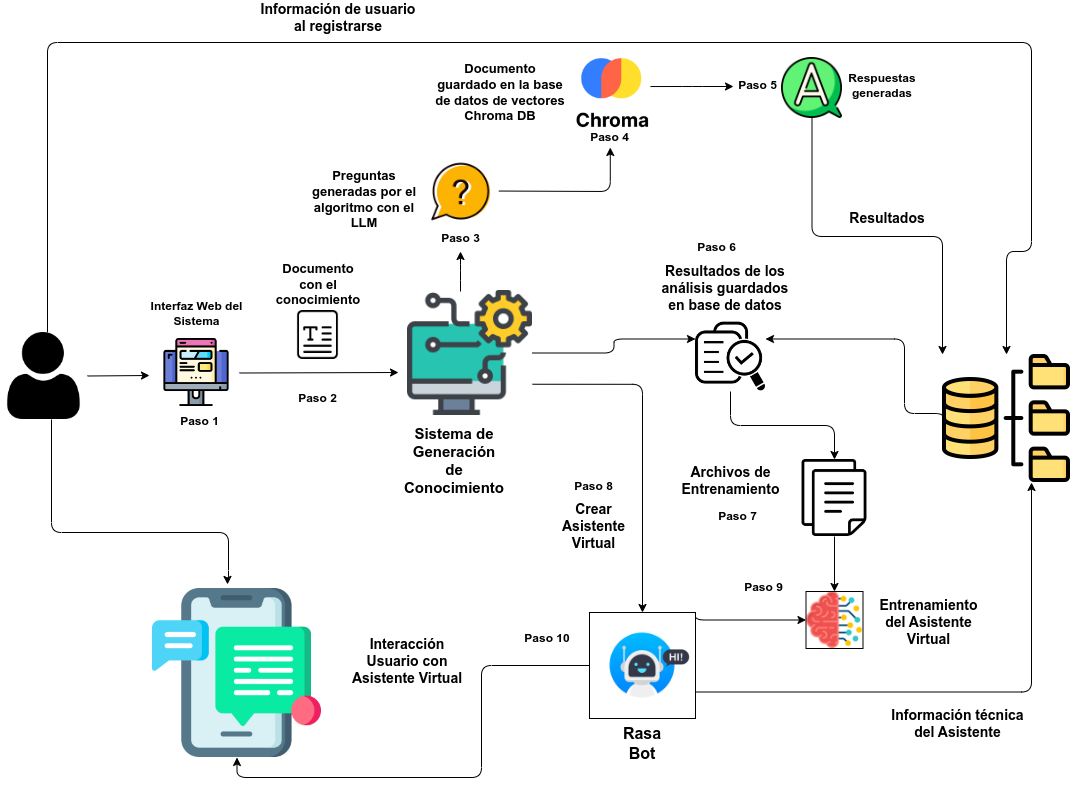
En este capítulo, se abordará el proceso de planificación y diseño del sistema, en el cual se han utilizado diferentes técnicas y herramientas para lograr el objetivo de esta investigación. Se presentarán los diagramas que se han utilizado para el modelado y documentación del sistema, así como la planificación para su desarrollo. Además, se discutirán los criterios que se han tenido en cuenta para la selección de las herramientas y técnicas utilizadas y se explicarán las ventajas de su uso en el proceso de desarrollo del sistema. El objetivo de este capítulo es proporcionar una visión general y detallada del proceso de planificación y diseño del sistema y servir como guía para el desarrollo de futuros proyectos de este tipo.

## Propuesta de Sistema.

Para darle solución a la problemática planteada se propone crear un sistema de gestión de asistentes virtuales basados en el marco de trabajo RASA, que le permitirá al usuario introducir un documento o una cierta cantidad de información para la creación de un asistente virtual. Esta información proporcionada por el usuario (e.d.; asociada al dominio u objetivo con el que se crea el asistente) se analizará por un modelo computacional (e.d., un *Gran Modelo de Lenguaje*) y se generará un asistente virtual de forma automática. Durante el proceso de edición del asistente virtual será posible modificar las respuestas que dará el asistente a posibles preguntas de un potencial usuario.

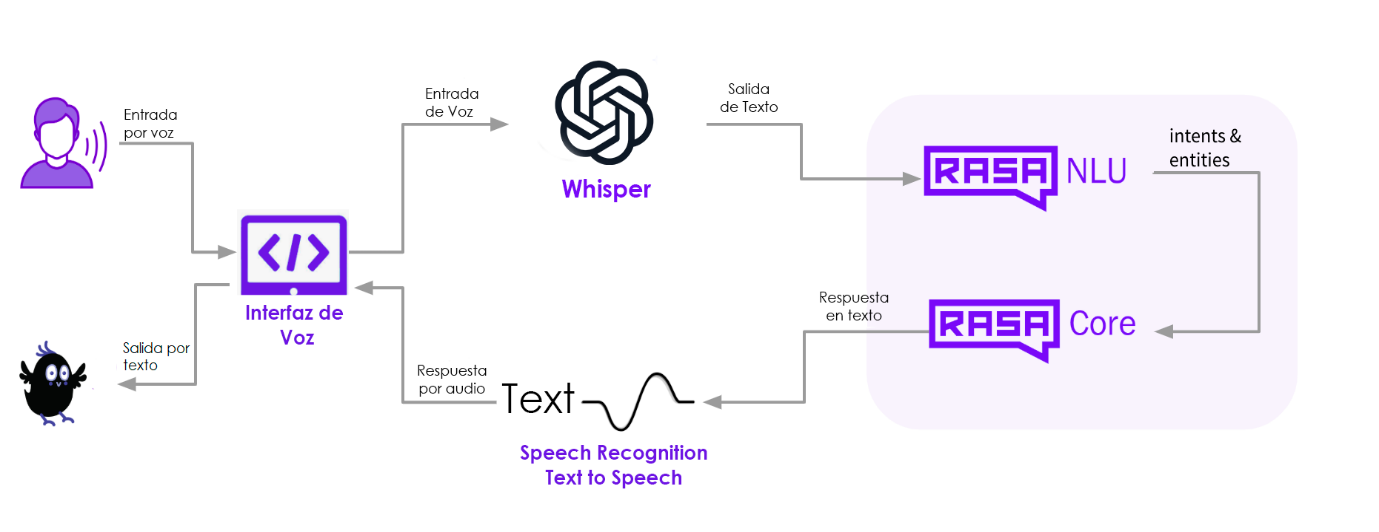
## Arquitectura del sistema.

En la [figura 6](#_Diagrama_d) se muestra el diagrama de la arquitectura del sistema de generación de asistentes virtuales. Este proceso comienza cuando el usuario llena un formulario para crear un asistente e introduce un documento con el conocimiento requerido. Este documento se procesa utilizando un algoritmo generador de preguntas auxiliándose las herramientas del marco de trabajo Langchain (el cual ofrece herramientas para la ingeniería de instrucciones (*prompts*; término en inglés) y estructuras de indicaciones, gestionando flujos de diálogo, contexto y memoria conversacional, lo que lo hace ideal para desarrollar aplicaciones conversacionales avanzadas), luego se almacena en la base de datos Chroma DB. Posteriormente, se consultan las preguntas generadas previamente para obtener respuestas, que son almacenadas en una base de datos MongoDB. Con estas preguntas y respuestas se crean los archivos de entrenamiento para el asistente, el cual se entrena y está listo para interactuar con el usuario a través de texto, audio o video.



### Figura 6 Diagrama de la arquitectura del sistema de generación de asistentes virtuales.

En la [figura 7](#_Diagrama_d_1), se muestra una vista de los componentes de la arquitectura de cada asistente creado por el usuario usando el marco de trabajo propuesto. El sistema presenta varios módulos, como la Interfaz de Voz, Whisper, que transcribe la pregunta del usuario de audio a texto para su comprensión, RASA\_NLU y RASA\_CORE para la comprensión del lenguaje natural y gestión de diálogos y por último Text to Speech que convierte la respuesta del bot de texto a voz.



### *Figura 7 Diagrama de la arquitectura de un Asistente creado por el marco de trabajo propuesto.*

## Historias de Usuario.

Las historias de usuario son tareas de desarrollo expresadas como "persona + necesidad + propósito", enfocadas en solucionar problemas para usuarios reales, permitiendo colaboración, impulsando soluciones creativas y motivando al equipo de desarrollo (Williams, 2003).

### Autenticar al Usuario en el Sistema.

Apartado donde el usuario puede iniciar sesión en el sistema con su correo electrónico o sus perfiles sociales, en caso de no tener cuenta en el sistema también puede registrarse en el apartado correspondiente.

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número: 1** | **Nombre de HU:** Autenticar al Usuario en el Sistema |
| **Usuario:** Cualquier usuario | |
| **Prioridad en negocio:** Alta | **Riesgo de desarrollo:** Bajo |
|  | |

### Crear asistente.

Donde se rellena el formulario que permite obtener la información necesaria para analizar el documento y generar las preguntas y respuestas para posteriormente pasarlas al fomulario de la derecha donde el usuario podrá editarlas en caso de que no este satisfecho o encuentre un error en alguna. Por ultimo podrá generar los archivos de entrenamiento para su asistente o guardar los resultados para posteriormente entrenar el asistente.

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número:2** | **Nombre de HU:** Crear asistente |
| **Usuario:** Usuario autenticado | |
| **Prioridad en negocio:** Alta | **Riesgo de desarrollo:** Alto |
|  | |

### Administrar asistentes.

Donde se edita, prueba y se eliminan asistentes, también se pueden buscar asistentes por cualquiera de los parámetros que tiene la tabla.

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número: 3** | **Nombre de HU:** Administrar asistentes |
| **Usuario:** Usuario autenticado | |
| **Prioridad en negocio:** Alta | **Riesgo de desarrollo:** Alto |
|  | |

## Requisitos No Funcionales.

Los requisitos no funcionales de RUP son criterios de calidad que especifican el rendimiento, la seguridad, la usabilidad, etc. del sistema. Incluso en metodologías ágiles donde la flexibilidad es crucial, incluirlos en su documento garantiza una comprensión completa de las expectativas del sistema y mejora la planificación y el diseño (Villanueva & Molina, s. f.).

### Software.

Los requisitos no funcionales del software son especificaciones que no se refieren directamente a las funciones del sistema, sino a sus características operativas, como rendimiento, seguridad, usabilidad y fiabilidad, que influyen en la experiencia del usuario y en la calidad del software (Dragos, 2021).

|  |  |
| --- | --- |
| **Requisito** | **Versión** |
| **Sistema Operativo** | El software es compatible con Windows 10 y la distribución Ubuntu 22.04. |
| **Node.js Versión** | 18 |
| **Python Versión** | 3.11.8 |
| **MongoDB Versión** | 5.0.x |

***Se recomienda realizar pruebas en otras distribuciones de linux y sistemas operativos basados en windows para comprobar la funcionalidad del proyecto.***

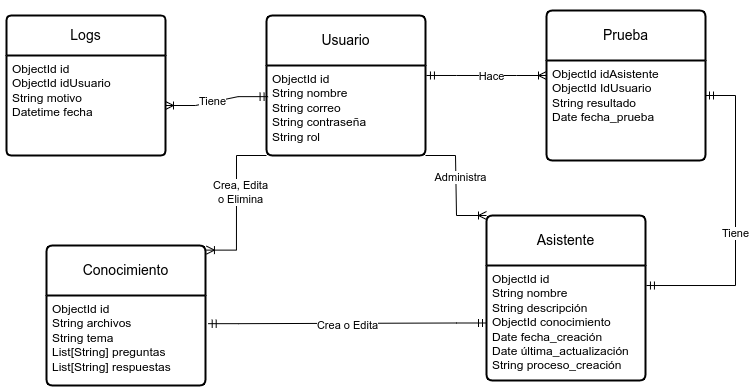
### Hardware.

Estos requisitos son importantes para garantizar que el sistema cumpla con los estándares de calidad y rendimiento deseados.

|  |  |
| --- | --- |
| **Componente** | **Requisitos mínimos** |
| Procesador | Procesador Intel Core i3 de 7ma generación con al menos 4 núcleos y 6 hilos. |
| RAM y GPU | Para optimizar el rendimiento de tareas como Whisper y modelos de lenguaje, se aconseja contar con al menos 8 GB de RAM. Además, se recomienda el uso de GPU, preferiblemente una tarjeta gráfica como la Nvidia GTX 960 con 4 GB de memoria DDR5, para una ejecución más eficiente. |
| Almacenamiento | Se necesita aproximadamente 20 GB de almacenamiento debido al peso de los modelos del lenguaje y de Whisper por su caché. |

## Diseño de Base de Datos.

El sistema emplea una base de datos NoSQL, específicamente MongoDB, que se centra en documentos. Este tipo de bases de datos están diseñadas para gestionar y manipular grandes volúmenes de datos a lo largo del tiempo. A diferencia de las bases de datos relacionales, las NoSQL no necesitan un esquema predefinido de tablas y columnas. En su lugar, utilizan colecciones, que son similares a las tablas en las bases de datos SQL, y documentos, que reemplazan a las filas de las bases de datos SQL. Los documentos en MongoDB no tienen un esquema fijo, lo que permite añadir campos de manera flexible según sea necesario.



## *Modelo de la Base de Datos NoSQL.*

## Usuarios del Sistema.

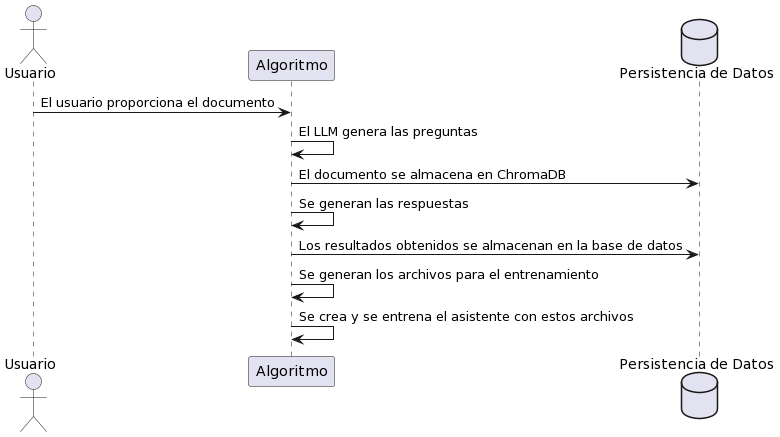
Analizar a los usuarios del sistema es crucial para garantizar la seguridad, eficiencia y personalización de la experiencia. Esta evaluación permite adaptar las políticas de acceso, identificar riesgos de seguridad y mejorar la usabilidad, optimizando así el rendimiento y la satisfacción del usuario (Villanueva & Molina, s. f.).

|  |  |
| --- | --- |
| Actor | Tareas que realiza |
| Cliente | Crea, edita, visualiza, elimina, prueba y entrena sus asistentes. |
| Administrador | Administra la base de datos del sistema, arregla fallos e implementa nuevas funcionalidades. |

### ***Tabla de usuarios del sistema.***

## Diagrama de Secuencia de creación de un asistente.

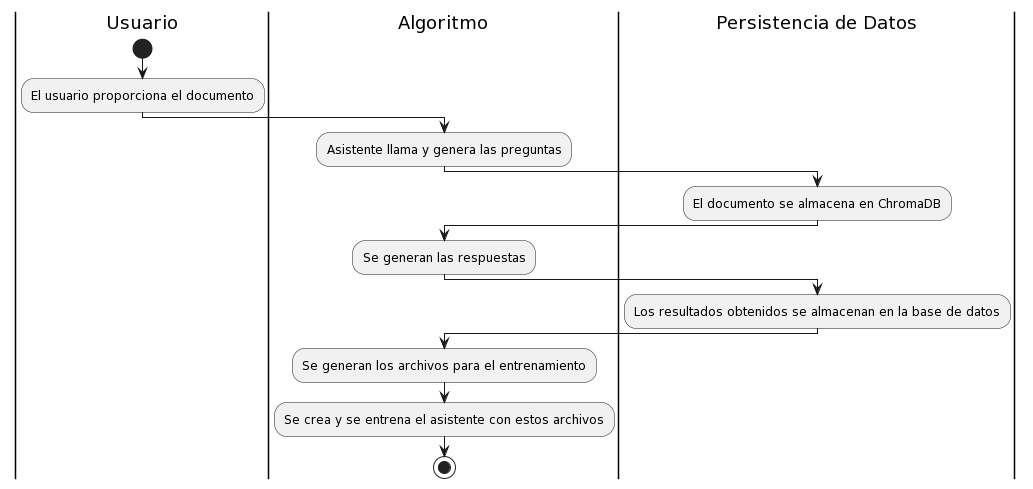
En la [figura 8](#_Fig_8._Dia) se observa el diagrama de secuencia que ilustra cómo el Usuario proporciona la entrada inicial, y luego el Algoritmo, con la ayuda del LLM, genera preguntas y respuestas, almacena datos en bases de datos, crea archivos de entrenamiento y, finalmente, entrena al asistente virtual. La Persistencia de Datos desempeña un papel crucial en el almacenamiento y recuperación de la información necesaria durante el proceso.



## *Fig 8. Diagrama de Secuencia de creación de un asistente.*

## Diagrama de Actividades de la creación de un asistente

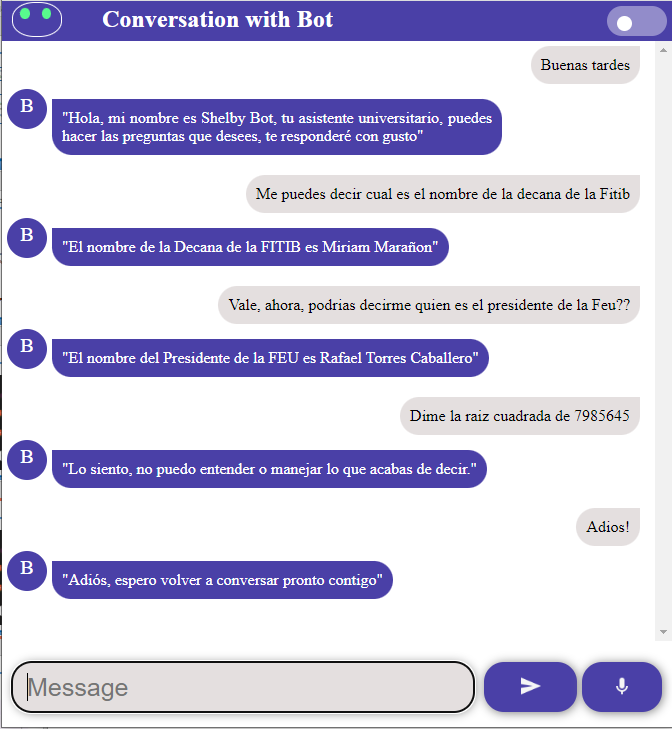
El proceso de creación de un asistente comienza cuando el usuario proporciona un documento para la generación de preguntas y respuestas. Luego, un algoritmo especializado, representado como "Algoritmo", se activa para analizar el contenido del documento y generar las preguntas pertinentes. Una vez que se generan las preguntas, el documento se almacena en Chroma DB, una base de datos especializada en este tipo de contenido. Posteriormente, se encarga de generar las respuestas adecuadas a partir del análisis del documento y las preguntas formuladas. Estas respuestas se registran en una base de datos para su posterior acceso y referencia. Luego, el asistente procede a generar archivos necesarios para el entrenamiento utilizando los datos recopilados. Finalmente, se lleva a cabo la creación y entrenamiento del asistente utilizando estos archivos, culminando así el proceso de creación del asistente inteligente como bien se observa en la [figura 9](#_Fig_9._Dia).



## *Fig 9. Diagrama de Actividades de Creación de un Asistente.*

## Diseño de la interfaz del Chat para probar el asistente

La interfaz del sistema tiene una sección que reconoce rostros y expresiones faciales, así como reconocimiento de voz para que el usuario pueda enviar sus solicitudes al asistente mediante comandos de voz, lo que facilita una comunicación más fluida. Dado que existe un contraste entre sus colores y fuentes de letra, esta interfaz es accesible para cualquier usuario, independientemente de sus discapacidades. Además, se adapta a dispositivos móviles y tablets tal como se muestra en la [figura 10](#_Figura_10.). La interfaz gráfica puede ser reemplazada por otra que el usuario estime conveniente, de esta puede tomar la lógica de conexión y aplicarla a otras interfaces.



## *Figura 10. Interfaz de chat de cada asistente creado.*

IMPLEMENTACIÓN Y PRUEBA

En este capítulo se abordará la fase de implementación y prueba de la solución propuesta. Se presentarán los detalles de la implementación del sistema, los componentes utilizados y la configuración necesaria para su funcionamiento. Además, se describirá el proceso de prueba y validación de la solución, así como los resultados obtenidos en las pruebas realizadas.

En este capítulo es fundamental para demostrar la viabilidad y eficacia de la solución propuesta, y para comprobar que cumple con los objetivos y requisitos establecidos en esta tesis.

## Algoritmos Importantes

En este epígrafe se realizará una explicación de algunos de los algoritmos más importantes de este proyecto.

### Inicio de sesión de usuario

**@auth\_router.post("/login")**

***async*** ***def*** login(*form\_data*: **OAuth2PasswordRequestForm** = **Depends()**):

user = users\_collection.find\_one({"email": form\_data.username})

if not user:

return JSONResponse(

*content*="Correo electrónico o contraseña incorrectos",

)

if not verify\_password(form\_data.password, user["hashed\_password"]):

return JSONResponse(

*content*="Correo electrónico o contraseña incorrectos",

)

access\_token\_expires = timedelta(*minutes*=settings.ACCESS\_TOKEN\_EXPIRE\_MINUTES)

access\_token = create\_access\_token(

*data*={"sub": user["email"]}, *expires\_delta*=access\_token\_expires

)

return {"access\_token": access\_token, "token\_type": "bearer"}

Este código define un endpoint de inicio de sesión (`/login`) que verifica si las credenciales proporcionadas son válidas. Si el correo electrónico no coincide con ningún usuario o la contraseña es incorrecta, se devuelve un mensaje de error. Si las credenciales son correctas, se genera un token de acceso y se devuelve como respuesta.

### Algoritmo para la creación de preguntas

Los primero es definir dos plantillas de texto para generar preguntas de práctica basadas en dominio general. Se utiliza la clase PromptTemplate de langchain\_core.prompts.prompt para crear plantillas de texto dinámicas que aceptan variables de entrada para formatear el texto final. Estas plantillas se utilizan para preparar preguntas de práctica, ya sea creando nuevas preguntas a partir de un texto dado o refinando preguntas existentes con más contexto. La clase PromptTemplate permite especificar el formato de la plantilla (por defecto, f-string o jinja2) y las variables de entrada esperadas, facilitando la generación de prompts personalizados para modelos de lenguaje.

**prompt\_template\_questions** = **"""**Eres un experto en crear preguntas de práctica basadas en material de dominio general.

Tu objetivo es escribir preguntas y respuestas a partir de un contexto dado. Lo haces haciendo preguntas sobre el texto a continuación:

**{text}**

Crea preguntas que prepararán a una persona para responderlas.

Asegúrate de no perder ninguna información importante.

PREGUNTAS:**"""**

**refine\_template\_questions** = **"""**Eres un experto en crear preguntas de práctica basadas en material de estudio.

Tu objetivo es ayudar a una persona a prepararse para responder estas preguntas.

Hemos recibido algunas preguntas de práctica hasta cierto punto: **{existing\_answer}**.

Tenemos la opción de refinar las preguntas existentes o agregar nuevas, (solo si es necesario) con algo más de contexto a continuación.

**{text}**

Dado el nuevo contexto, refina las preguntas originales en español. Si el contexto no es útil, por favor proporciona las preguntas originales.

PREGUNTAS: **"""**

**PROMPT\_QUESTIONS** = **PromptTemplate**(

***template***=prompt\_template\_questions, ***input\_variables***=["text"]

)

REFINE\_PROMPT\_QUESTIONS = PromptTemplate(

*input\_variables*=["existing\_answer", "text"],

*template*=refine\_template\_questions,

)

Este código carga un documento PDF (el formato que se utilizará para aprender porque es un tipo de archivo más pequeño y no tendrá problemas al reconocer caracteres ya que los archivos Word se corrompen y cambian el formato de los caracteres a ASCII), divide su contenido en fragmentos y genera preguntas basadas en esos fragmentos utilizando un modelo de lenguaje. Primero, PyPDFLoader carga el documento PDF desde DATA\_PATH. Luego, se une el contenido de todas las páginas en una sola cadena de texto. Después, RecursiveCharacterTextSplitter divide esta cadena en fragmentos de 10000 caracteres con una superposición de 50 caracteres entre fragmentos (para mantener juntos los fragmentos que están semánticamente relacionados), este proceso lo que permite es que el modelo analice por fragmentos para poder pasarle grandes cantidades de información sin sobrepasar el número de tokens permitidos por el modelo como contexto. Estos fragmentos se convierten en documentos y se pasan a una cadena de procesamiento que utiliza un modelo de lenguaje para generar preguntas basadas en los documentos. Finalmente, las preguntas generadas se almacenan en la variable questions.

loader = **PyPDFLoader**(DATA\_PATH)

data = loader.**load()**

text\_question\_gen = ""

**for** page **in** data:

text\_question\_gen **+=** page.page\_content

text\_splitter\_question\_gen = **RecursiveCharacterTextSplitter**(

***chunk****\_****size***=10000, ***chunk\_overlap***=50

)

text\_chunks\_question\_gen= text\_splitter\_question\_gen.**split**\_**text**(

text\_question\_gen

)

docs\_question\_gen=[**Document**(***page\_content***=t)**for** t **in**  text\_chunks\_question\_gen]

question\_gen\_chain = **load\_summarize\_chain**(

***llm***=LlamaCpp(

***model\_path***=settings.LLM\_PATH,

***temperature***=0.75,

***top\_p***=1,

***verbose***=True,

***n\_ctx***=4096,

),

***chain\_type***="refine",

***verbose***=True,

***question\_prompt***=PROMPT\_QUESTIONS,

***refine\_prompt***=REFINE\_PROMPT\_QUESTIONS,

)

questions = question\_gen\_chain.**run**(docs\_question\_gen)

### Algoritmo para la creación de las respuestas a las preguntas previamente generadas

Este código genera respuestas a preguntas utilizando un modelo de lenguaje y una base de datos de vectores Chroma, que almacena documentos y sus embeddings de HuggingFace. Primero, inicializa los embeddings y la base de datos Chroma con documentos. Luego, crea una cadena de recuperación de preguntas y respuestas que utiliza el modelo de lenguaje y la base de datos para generar respuestas a las preguntas

proporcionadas.

embeddings = **HuggingFaceEmbeddings**(

***model\_name***="jaimevera1107/all-MiniLM-L6-v2-similarity-es",

***model\_kwargs***={"device": "cpu"},

)

vector\_store = **Chroma**.from\_documents(

docs\_question\_gen, embeddings

)

answer\_gen\_chain = **RetrievalQA**.from\_chain\_type(

***llm*** =LlamaCpp(

***model\_path***=settings.**LLM\_PATH**,

***temperature***=0.75,

***top\_p***=1,

***verbose***=True,

***n\_ctx***=4096,

),

***chain\_type***="stuff",

***retriever***=vector\_store.**as\_retriever**(***k***=2),

)

question\_list = questions.questions

answers = []

**for** idx, question **in enumerate**(question\_list):

answer = answer\_gen\_chain.**run**(question)

answers.**append**(*f*"{idx+1} - {answer}")

### Algoritmo de Reconocimiento del Habla y Transcripción

Los asistentes generados tienen una interfaz que permite que el usuario se comunique con ellos usando reconocimiento de audio, entonces este es el algoritmo que permite que esto se lleve a cabo usando Whisper.

Este código define un punto final para recibir archivos de audio en formato MP3 a través de una solicitud POST en la ruta "/whisper/audio". El archivo de audio se recibe como un objeto UploadFile de FastAPI. Luego, el código lee los bytes del archivo de audio y los guarda en un archivo temporal llamado "audio.mp3". A continuación, se carga el archivo de audio usando la biblioteca "whisper" y se ajusta su duración si es necesario para que tenga una duración fija. Después, se calcula el espectrograma logarítmico de mel del audio. Se configuran las opciones de decodificación, como el uso de punto flotante de 32 bits y el idioma español. Finalmente, se realiza la decodificación del audio utilizando un modelo preentrenado proporcionado por la variable "model" y se devuelve el texto resultante. El archivo de audio temporal se elimina al finalizar el proceso.

**@app.post**("/whisper/audio")

***async* *def*** **recive\_audio**(*file*: **UploadFile**=**File**(...)):

audio\_bytes = file.file.**read**()

**with open**("audio.mp3" , "wb") **as** f:

f.**write**(audio\_bytes)

audio = whisper.**load\_audio**("audio.mp3")

audio = whisper.**pad\_or\_trim**(audio)

mel = whisper.**log\_mel\_spectrogram**(audio).**to**(model.device)

options = whisper.**DecodingOptions**(***fp16***=False,***language***="es")

result = whisper.**decode**(model,mel,options)

os.**remove**("audio.mp3")

**print**(result.text)

return result.text

## Pruebas al sistema

Un caso de prueba representa un método para evaluar un sistema a través de ensayos, en los que se deben ingresar datos para verificar si el sistema produce los resultados previstos bajo condiciones de prueba específicas. Uno de los fundamentos de la metodología XP es el proceso de pruebas, el cual fomenta la realización de pruebas en la mayor medida posible, con el fin de reducir la cantidad de errores no detectados y acortar el tiempo transcurrido entre la ocurrencia de un error y su identificación (Mustafa et al., 2021) .

Las pruebas de funcionalidad y de aceptación se elaboran sobre la base de las historias de usuarios en cada ciclo de iteración del desarrollo del software, lo que permite corroborar que la historia ha sido implementada de manera adecuada. En caso de que varias pruebas fallen, deben señalar el orden de prioridad para su resolución. A lo largo de la implementación de la aplicación, se diseñó un conjunto de casos de prueba para verificar su funcionamiento de acuerdo con los requisitos descritos en las historias de usuario, las cuales fueron definidas en el capítulo 2.

**Casos de prueba ejecutados**

Tabla 3.7 **Caso de Prueba “Registrarse en el sistema usando su correo electrónico”**

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de Prueba** | **Entrada** | **Resultado Esperado** | **Condición** |
| CP1 | Correo electrónico válido | El sistema registra al usuario y muestra un mensaje de éxito | El correo electrónico no está asociado a otra cuenta |
| CP2 | Correo electrónico inválido | El sistema muestra un mensaje de error | El formato del correo electrónico es inválido |
| CP3 | Correo electrónico ya registrado | El sistema muestra un mensaje indicando que el correo electrónico ya está en uso | El correo electrónico ya está asociado a una cuenta existente |

Tabla 3.8 **Caso de Prueba “Crear asistentes virtuales rellenando los campos de nombre, descripción, archivos de conocimiento e imágenes del asistente.”**

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de Prueba** | **Entrada** | **Resultado Esperado** | **Condición** |
| CP1 | Documento proporcionado para generar preguntas y respuestas | Preguntas generadas y almacenadas en Chroma DB. Respuestas adecuadas generadas y almacenadas en la base de datos. Archivos de entrenamiento creados. Asistente inteligente creado y entrenado. | El algoritmo analiza correctamente el contenido del documento y genera preguntas pertinentes. Las respuestas generadas son precisas y relevantes. Los archivos de entrenamiento se crean correctamente. El asistente se entrena adecuadamente. |
| CP2 | Documento vacío o no válido | Mensaje de error indicando que el documento no es válido o está vacío. | El algoritmo detecta que el documento proporcionado no contiene información válida para generar preguntas y respuestas. |
| CP3 | Documento con contenido irrelevante o no estructurado | Preguntas generadas con base en el contenido relevante del documento. Respuestas adecuadas generadas. Archivos de entrenamiento creados. Asistente inteligente creado y entrenado. | El algoritmo es capaz de identificar y omitir el contenido irrelevante del documento, generando preguntas solo sobre la información pertinente. Las respuestas son precisas y relevantes, incluso si el documento no está estructurado correctamente. |
| CP4 | Documento con contenido técnico o especializado | Preguntas generadas que demuestren comprensión del contenido técnico. Respuestas precisas y detalladas generadas. Archivos de entrenamiento creados. Asistente inteligente creado y entrenado. | El algoritmo muestra capacidad para comprender y generar preguntas sobre contenido técnico o especializado. Las respuestas reflejan un nivel adecuado de detalle y precisión. |
| CP5 | Documento en un idioma diferente al predeterminado | Preguntas y respuestas generadas en el idioma del documento. Archivos de entrenamiento creados en el mismo idioma. Asistente inteligente creado y entrenado en el idioma correspondiente. | El algoritmo es capaz de procesar y generar preguntas y respuestas en diferentes idiomas, manteniendo la coherencia y precisión. |

Tabla 3.9 **Caso de Prueba “Probar los asistentes creados”**

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de Prueba** | **Entrada** | **Resultado Esperado** | **Condición** |
| CP1 | Interacción de el usuario con el asistente | El asistente responde a las consultas del usuario de manera adecuada | El asistente virtual está correctamente configurado y funcional |

## Análisis económico del costo de producción del sistema

En la realización de un proyecto se hace necesaria la planificación y el control del esfuerzo, costo y tiempo que tomará llevarlo a cabo. Con la utilización de métodos de estimación de costos, se puede determinar una aproximación de los recursos necesarios, así como el total de tiempo que gastaría una persona o un equipo, en el desarrollo de un producto de software específico. A continuación, se realiza un análisis de costos para el sistema de asistentes virtuales.

### Estimación de costo y tiempo

Para determinar los costos de los sistemas desarrollados se usará el método de puntos en casos de uso. Este es un método de estimación prometedor que se adapta bien al enfoque de caso de uso para la descripción de los requisitos. En sus bases yace el concepto de transacción de caso de uso, la unidad más pequeña de medición. Se realizará este análisis teniendo en cuenta las Historias de Usuario proporcionadas.

El método de punto de casos de uso consta de cuatro etapas, en las que se desarrollan los siguientes cálculos:

Ecuación 3.1: Cálculo de los Puntos de Historias de Usuarios sin ajustar

UUCP = UAW + UUCW

Donde:

UUCP: Puntos de Historias de Usuarios sin ajustar.

hUAW: Factor de Peso de los Actores sin ajustar.

UUCW: Factor de Peso de Historias de Usuarios sin ajustar.

Factor de Peso de los Actores sin ajustar (UAW)

Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos.

En la [tabla 3.1](#_Factor_de_) se presenta el Factor de Peso de los Actores sin ajustar.

## Total de Puntos de Actores sin ajustar (UAW): 7

**Cálculo del UUCW**

## Para calcular el UUCW (Unadjusted Use Case Weight) para tus historias de usuario, primero necesitamos determinar la complejidad de cada una de ellas según el número de transacciones que involucran. Luego multiplicaremos este valor por el peso asignado a cada tipo de historia de usuario tal como se muestra en la [tabla 3.2](#_Factor_de__1)

**Cálculo de los Puntos de Historias de Usuarios ajustadas.**

Una vez que se tienen los Puntos de Historias de Usuarios, se debe ajustar este valor como se muestra en la ecuación 3.2.

**Ecuación 3.2**: Cálculo de los Puntos de Historias de Usuarios ajustadas

**UCP = UUCP \* TCF \* EF**

Donde:

* **UCP**: Puntos de Historias de Usuarios ajustados.
* **UUCP**: Puntos de Historias de Usuarios sin ajustar.
* **TCF**: Factor de complejidad técnica.
* **EF**: Factor de ambiente.

**Factor de complejidad técnica** (**TCF**).

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante o nulo y 5 un aporte muy importante, tal como se observa en la [tabla 3.3](#_3.2._Facto).

**Para Calcular TCF:** Factor de complejidad técnica se muestra la ecuación 3.3 cálculo del factor complejidad técnica.

**Ecuación 3.3:** Cálculo del Factor de complejidad técnica

**TCF = 0.6 + 0.01 \* Σ (Pesoi\* Valori)**

**TCF = 0.6 + 0.01 \* 40**

**TCF = 1**

**Factor Ambiente (EF).**

El factor de ambiente está relacionado con las habilidades y entrenamiento del grupo de desarrollo. Cada factor se cuantifica con un valor desde 0 (aporte irrelevante) hasta 5 (aporte muy relevante).

En la [tabla 3.4](#_3.4__Facto) se muestra factor de ambiente con su significado y el peso de cada uno de estos factores:

**Para Calcular EF**: Factor de ambiente se muestra la ecuación 3.4 cálculo del factor ambiente.

**Ecuación 3.4**: Cálculo del Factor de ambiente

**EF** = 1.4 - 0.03 \* Σ (Peso**i** \* Valor**i**)

**EF** = 1.4 - 0.03 \* 9.5

**EF** = 0.965

Luego: UCP=UUCP \* TCF \* EF

**UCP = 42 \* 1 \* 0.965**

**UCP = 40.53**

## Estimación de esfuerzo a través de los Puntos de Historias de Usuarios.

**Ecuación 3.5**: Esfuerzo estimado en horas hombres.

E = UCP \* CF

Donde:

**E:** Esfuerzo estimado en horas hombres.

**UCP**: Punto de historias de usuarios ajustadas.

**CF**: Factor de conversión.

Para obtener el factor de conversión (CF) se cuentan cuántos valores de los que afectan el factor ambiente (E1 a E6) están por debajo de la media (**<3**), y los que están por encima (**>3**) para los restantes (E7 a E8). Si el total (nos da **0**) es 2 o menos se utiliza el factor de conversión 20 Horas- Hombre / Punto de historias de usuarios. Si el total es 3 o 4 se utiliza el factor de conversión 28 Horas-Hombre / Punto de historias de usuarios. Si el total es mayor o igual que 5 se recomienda efectuar cambios en el proyecto ya que se considera que el riesgo de fracaso es demasiado alto. En este caso:

CF= 20 Horas-hombre / Puntos de historias de usuarios.

Luego

**E = 40.53 \* 20 horas-hombre**

**E = 810.6 horas-hombre**

En la siguiente [tabla 3.5](#_3.5_Distri) se muestra distribución del esfuerzo por etapas.

Una vez estimado el tiempo de desarrollo del proyecto y conociendo la cantidad de desarrolladores y el pago que recibe cada uno de estos se puede llevar a cabo una estimación del costo total del proyecto referidos a los recursos humanos; existen otros costos como por ejemplo del equipamiento que se suman al anterior.

**K**: Coeficiente que tiene en cuenta los costos indirectos (1,5 y 2,0).

**THP:** Tarifa Horaria Promedio. El salario promedio mensual de los trabajadores en este caso es de $15 000 CUP dividido entre 176h.

**176 horas** (horas de trabajo para 1 mes, esto se toma a razón de 24 días, ya que no se cuentan los fines de semana ni sábados cortos).

**Tiempo**= 810.6 horas / 176 ≈ equivalente a 4 meses y 15 días, este es el tiempo que tomaría desarrollar el proyecto empleando una sola persona.

Como parte del proceso de investigación de este trabajo se desempeño un solo trabajador al cual le toma 4 meses y 15 días culminar el proyecto.

El tiempo para 3 trabajadores sería de 1 mes y 16 días

**Entonces el costo total del proyecto:**

**C = E (Total) \* K \* THP**

**C** = 810.6 \* 1.5 \* 15 000/176 = **$ 103 627**

Para el salario a los trabajadores se investigó como realizan el pago en diferentes organizaciones o sucursales en Santiago de Cuba de diferente sector ya sea estatal o privado, dada las nuevas regulaciones y tasas de cambio. Los datos se obtuvieron a través de trabajadores de las entidades y por anuncios laborales.

Para una información más detallada sobre la estimación de pago de trabajadores ver la [tabla 3.6](#_3.6_Situac)

## Conlusiones del capítulo

En este capítulo, se ha completado la implementación y prueba del sistema propuesto. Se han desarrollado y probado componentes clave, como el inicio de sesión de usuario, la generación de preguntas y respuestas, y el reconocimiento del habla. Además, se ha detallado la metodología de prueba utilizada, con casos específicos que abarcan desde el registro de usuarios hasta la interacción con los asistentes virtuales creados. Además, se ha realizado un análisis económico para estimar el costo del proyecto, considerando factores como la complejidad técnica y los recursos humanos necesarios.

CONCLUSIONES Y RECOMENDACIONES

**Conclusiones**

1. Se evidenció que la falta de sistemas adaptados a las necesidades de grupos específicos, como personas ancianas, discapacitadas, con problemas visuales y niños, ha generado dificultades significativas en la comunicación y acceso a la información.
2. Se intregró de la investigación "Herramienta digital para la construcción de conocimiento automático para un Asistente Virtual" del año 2022, junto con mejoras centradas en la experiencia del usuario, ha representado un avance crucial para abordar la brecha en la comunicación mencionada anteriormente.
3. Se estableció mediante el estudio del estado del arte una base sólida para la modificación del algoritmo de generación de preguntas y respuestas, así como para la integración de un modelo de lenguaje.
4. El análisis exhaustivo de tecnologías como Whisper para el procesamiento de texto y la transcripción de audio posibilitó una integración exitosa de la plataforma, promoviendo una experiencia de usuario más accesible e interactiva.
5. Las pruebas de la plataforma han validado la viabilidad de la solución propuesta, demostrando su capacidad para comprender y responder a las necesidades de los usuarios de manera más efectiva, especialmente en lo que respecta a su estado emocional y requisitos de comunicación específicos.

**Recomendaciones**

1. Mejorar la interfaz de usuario del sistema para mejorar la experiencia visual del usuario.
2. Mejorar el chat brindado para cada asistente, haciéndolo un poco más atractivo mediante el uso de efectos y colores.
3. Crear un sistema de tareas con Celery para la creación de los asistentes virtuales, ya que puede ser una tarea que pueda durar alrededor de 5 a 10 minutos, dependiendo de las prestaciones del servidor.
4. Agregar una base de datos para el seguimiento de los chats, lo cual le aportaría la capacidad de recordar formularios y entidades a los asistentes creados.
5. Incluir el idioma inglés para que el asistente pueda ser creado con conocimiento en este idioma.

REFERENCIAS BIBLIOGRÁFICAS

1. M. (2018, April 24). *5 learnings from our ‘Chatbot Survey — 2017’ - Chatbots Journal*. Medium. <https://chatbotsjournal.com/5-learnings-from-our-chatbot-survey-2017-72a6a4fc209c?gi=21ae27bf4d12>
2. Argüello, F. (2024, March 22). *10 principales API de Reconocimiento Facial - Infoteknico*. Infoteknico. <https://www.infoteknico.com/principales-api-de-reconocimiento-facial/>
3. Asistente virtual: retos y posibilidades para la inteligencia artificial. (n.d.). Tableau. <https://www.tableau.com/data-insights/ai/ai-virtual-assistant>
4. A. (2021, December 16). What is RASA? — the open-source AI for building conversational chatbots. Medium. <https://medium.com/@asklua/what-is-rasa-the-open-source-ai-for-building-conversational-chatbots-8a86bef47ec4>
5. Malik, F. (2021, December 15). Build And Host Fast Data Science Applications Using FastAPI. Medium. <https://towardsdatascience.com/build-and-host-fast-data-science-applications-using-fastapi-823be8a1d6a0>
6. Caballero, Rosbel. (2021). Chatbot: a viable proposition for customer service in the UCI support center. <https://www.researchgate.net/publication/357689698_Chatbot_a_viable_proposition_for_customer_service_in_the_UCI_support_center>
7. Funcionamiento de Amazon Lex - Amazon Lex V1. (n.d.). <https://docs.aws.amazon.com/es_es/lex/latest/dg/how-it-works.html>
8. Payne, R. (2022, January 19). React, Angular, Vue, and Svelte: A Comparison. <https://www.accelebrate.com/blog/react-angular-vue-svelte-comparison>
9. Condez, A. (2022, February 13). Companies That Use Flask. Career Karma. <https://careerkarma.com/blog/companies-that-use-flask/>
10. Dhungana, K. (2023, noviembre 10). An Overview of ChromaDB: The Vector Database. Medium. <https://medium.com/@kbdhunga/an-overview-of-chromadb-the-vector-database-206437541bdd>
11. ChatBot, P., & ChatBot, P. (2022, June 6). Dialogflow de Google cambia su modelo de pricing: ¿cuánto cuesta la IA / NLU? - Planeta Chatbot. Planeta Chatbot - Comunidad De Expertos En IA Conversacional. <https://planetachatbot.com/dialogflow-google-cambia-modelo-pricing/>
12. Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P. E., Lomeli, M., Hosseini, L., & Jégou, H. (2024, January 16). The Faiss library. arXiv.org. <https://doi.org/10.48550/arXiv.2401.08281>
13. Dragos, P. (2021). Overview of the Agile Rational Unified Process (Rup) in the Context of Software Development Projects. Journal of Business and Economics, 12(6), 681-684. <https://repository.unpak.ac.id/tukangna/repo/file/files-20211015150215.pdf#page=107>
14. A. (2023, November 9). FastAPI explicado en 5 minutos o menos. Geekflare.

<https://geekflare.com/es/fastapi-explained/>

1. Han, Y., Liu, C., & Wang, P. (2023). A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv preprint arXiv:2310.11703*. <http://arxiv.org/abs/2310.11703>
2. Pérez, E. (2023, March 12). He usado Whisper para transcribir una entrevista: es la herramienta que llevaba esperando desde hace años. Xataka. <https://www.xataka.com/aplicaciones/he-usado-whisper-para-transcribir-entrevista-herramienta-que-llevaba-esperando-hace-anos>
3. Cómo elegir un framework para el backend. (n.d.). CódigoFacilito. <https://codigofacilito.com/articulos/elegir-framework-backend>
4. IBM watsonx Assistant Virtual Agent. (n.d.). <https://www.ibm.com/products/watsonx-assistant>
5. Khurana, D., Koli, A., Khatter, K., & Singh, S. (2022, July 14). Natural language processing: state of the art, current trends and challenges. Multimedia Tools and Applications, 82(3), 3713–3744. <https://doi.org/10.1007/s11042-022-13428-4>
6. Kumari, P. (2024, February 14). Llama 2 Vs GPT-3.5 Vs GPT-4: What, When &amp; How To Chose. Labellerr. <https://www.labellerr.com/blog/llama2-vs-gpt-3-5-vs-gpt-4/>
7. Londoño, P. (2023, April 4). Qu&eacute; es Python, para qu&eacute; sirve y c&oacute;mo se usa (+ recursos para aprender). <https://blog.hubspot.es/website/que-es-python>
8. Luzniak, K. (2023, November 17). Is Llama 2 Better Than GPT Models? 6 Main Differences Between Llama 2 vs. GPT-4 vs. GPT-3.5. Neoteric. <https://neoteric.eu/blog/6-main-differences-between-llama2-gpt35-and-gpt4/>
9. Mancuzo, G. (2024, March 8). Metodología XP: La Mejor Vía para el Desarrollo de Software. Blog - ComparaSoftware. <https://blog.comparasoftware.com/metodologia-xp/>
10. Canive, T. (2020, May 27). Metodología XP o Programación Extrema: ¿Qué es y cómo aplicarla? Gestor De Proyectos Online. <https://www.sinnaps.com/blog-gestion-proyectos/metodologia-xp>
11. Mustafa, A., Wan-Kadir, W. M., Ibrahim, N., Shah, M. A., Younas, M., Khan, A., ... & Alanazi, F. (2021). Automated test case generation from requirements: A systematic literature review. Computers, Materials and Continua, 67(2), 1819-1833. <https://www.researchgate.net/profile/Ahmad-Mustafa-7/publication/349096514_Automated_Test_Case_Generation_from_Requirements_A_Systematic_Literature_Review/links/601f6f924585158939892bfc/Automated-Test-Case-Generation-from-Requirements-A-Systematic-Literature-Review.pdf>
12. de Oliveira, G., Venson, R., & Marcelino, R. (2018). REDES NEURAIS APLICADAS NO DESENVOLVIMENTO DE CHATBOTS: UMA ANÁLISE BIBLIOMÉTRICA. ARTEFACTUM-Revista de estudos em Linguagens e Tecnologia, 17(2). <https://www.academia.edu/72166314/Redes_Neurais_Aplicadas_No_Desenvolvimento_De_Chatbots_Uma_An%C3%A1lise_Bibliom%C3%A9trica>
13. Primeros pasos con Botpress | Botpress Blog. (n.d.). <https://botpress.com/es/blog/getting-started-with-botpress>
14. Bueno, P. C. (2023, December 18). Qué es Django y por qué usarlo. OpenWebinars.net. <https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/>
15. Qué es Git | Atlassian Git Tutorial. (n.d.). Atlassian. <https://www.atlassian.com/es/git/tutorials/what-is-git>
16. ¿Qué es Python? - Explicación del lenguaje Python - AWS. (n.d.). Amazon Web Services, Inc. <https://aws.amazon.com/es/what-is/python/>
17. Chantiri, L. (2023, March 14). ¿Qué es y cómo funciona Whisper? | Código Espagueti. Código Espagueti. <https://codigoespagueti.com/noticias/tecnologia/que-es-como-funciona-whisper/>
18. Boisdequin, H. (2020, November 30). React vs Vue vs Angular vs Svelte. DEV Community. <https://dev.to/hb/react-vs-vue-vs-angular-vs-svelte-1fdm>
19. Singh, P. N., Talasila, S., & Banakar, S. V. (2023, December). Analyzing Embedding Models for Embedding Vectors in Vector Databases. In 2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG) (pp. 1-7). IEEE. <https://ieeexplore.ieee.org/abstract/document/10455990/>
20. Sobrino, D. C. (2020, May 15). ¿Cómo funciona el reconocimiento automático del habla? Medium. <https://medium.com/soldai/c%C3%B3mo-funciona-el-reconocimiento-autom%C3%A1tico-del-habla-eb038ecfe72e>
21. Team, P. (2022, January 14). Advantages of Django | Disadvantages of Django. Python Geeks. <https://pythongeeks.org/advantages-disadvantages-of-django/>
22. TechEmpower Framework Benchmarks. (n.d.). <https://www.techempower.com/benchmarks/#section=datar20&hw=ph&test=fortune&l=zijzen-sf>
23. Edwards, B. (2023, May 11). The AI race heats up: Google announces PaLM 2, its answer to GPT-4. Ars Technica. <https://arstechnica.com/information-technology/2023/05/googles-top-ai-model-palm-2-hopes-to-upstage-gpt-4-in-generative-mastery/>
24. Topsakal, O., & Akinci, T. C. (2023, July). Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In International Conference on Applied Engineering and Natural Sciences (Vol. 1, No. 1, pp. 1050-1056). <https://www.researchgate.net/profile/Oguzhan-Topsakal/publication/372669736_Creating_Large_Language_Model_Applications_Utilizing_LangChain_A_Primer_on_Developing_LLM_Apps_Fast/links/64d114a840a524707ba4a419/Creating-Large-Language-Model-Applications-Utilizing-LangChain-A-Primer-on-Developing-LLM-Apps-Fast.pdf>
25. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288. <https://sh-tsang.medium.com/brief-review-llama-2-open-foundation-and-fine-tuned-chat-models-6666eb8b56b7>
26. Villanueva, E., & Molina, D. F. Aporte a RUP para el Desarrollo de Aplicaciones Web Transaccionales–Requerimientos no Funcionales y Administración de Riesgos. <https://www.academia.edu/download/32737879/paper_rup.pdf>
27. Acosta, E. V. (2023, March 3). Visual Studio vs Visual Studio Code: ¿Cuál es la diferencia entre estos editores de código IDE? freeCodeCamp.org. <https://www.freecodecamp.org/espanol/news/visual-studio-vs-visual-studio-code-cual-es-la-diferencia-entre-estos-editores-de-codigo-ide/>
28. Williams, L. (2003). The xp programmer: The few-minutes programmer. IEEE software, 20(3), 16. <https://collaboration.csc.ncsu.edu/laurie/Papers/fewMinutes.pdf>
29. Xu, B. (2023, February 23). Domain-Specific Bots vs. Generic Bots - DealerAI. DealerAI. <https://dealerai.com/domain-specific-bots-vs-generic-bots/>

**ANEXOS**

### 3.1. Factor de Peso de los Actores sin ajustar.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo** | **Descripción** | **Peso** | **Cant \* peso** |
| Simple | Otro sistema que interactúa mediante una interfaz de programación de aplicaciones. (API) | 1 | 1\*1 |
| Medio | Otro sistema que interactúa mediante un protocolo o una persona interactuando con una interfaz basada en texto. | 2 | 2\*2 |
| Complejo | Una persona que interactúa con el sistema mediante una interfaz gráfica. (GUI) | 3 | 1\*3 |
| Total |  |  | 1+4+3= **7** |

### 3.2. Factor de Peso de Historias de Usuarios sin ajustar

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo | Descripción | Peso | Cant \* peso |
| Simple | La HU contiene de 1 a 3 transacciones. | 5 | 5\*5 |
| Medio | La HU contiene de 4 a 7 transacciones. | 10 | 1\*10 |
| Complejo | La HU contiene más de 8 transacciones. | 15 | 0\*15 |
| Total |  |  | 25+10=35 |

### 3.3. Factor de Complejidad Técnica

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Factor | Descripción | **Peso** | **Valor** | **(Peso-i \*Valor-i)** |
| T1 | Sistema distribuido | 2 | 2 | 4 |
| T2 | Rendimiento o tiempo de respuesta | 1 | 2 | 2 |
| T3 | Eficiencia del usuario final | 1 | 2 | 2 |
| T4 | Procesamiento interno complejo | 2 | 3.5 | 7 |
| T5 | El código debe ser reutilizable | 1 | 3 | 3 |
| T6 | Facilidad de instalación | 0.5 | 2 | 1 |
| T7 | Facilidad de uso | 1 | 5 | 5 |
| T8 | Portabilidad | 1 | 2 | 2 |
| T9 | Facilidad de cambio | 1 | 3 | 3 |
| T10 | Concurrencia | 1 | 4 | 4 |
| T11 | Incluye objetivos especiales de seguridad | 1 | 3 | 3 |
| T12 | Acceso directo a terceras partes | 1 | 2 | 2 |
| T13 | Se requieren facilidades especiales de entrenamiento a los usuarios | 1 | 2 | 2 |
| Total |  |  |  | **40** |

### 3.4 Factor de ambiente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Factor** | **Descripción** | **Peso** | **Valor** | **(Peso-i \* Valor-i)** |
| E1 | Familiaridad con el modelo de proyecto utilizado | 1 | 3 | 3 |
| E2 | Experiencia en la aplicación | 1 | 2 | 2 |
| E3 | Experiencia en orientación a objetos | 1 | 4 | 4 |
| E4 | Capacidad del analista líder | 1 | 3 | 3 |
| E5 | Motivación | 1 | 3 | 3 |
| E6 | Estabilidad de los requerimientos | 1 | 2 | 2 |
| E7 | |  |  | | --- | --- | | Personal part-time |  | | -1 | 0 | 0 |
| E8 | Dificultad del lenguaje de programación | -1 | 1 | -1.5 |
| **Total** |  |  |  | **14.5** |

### 3.5 Distribución del esfuerzo por etapas

|  |  |  |
| --- | --- | --- |
| **Actividad** | **% esfuerzo** | **Valor esfuerzo** |
| Planificación | 10 | 81.06 |
| Diseño | 20 | 121.59 |
| Codificación | 40 | 324.24 |
| Prueba | 15 | 121.59 |
| Sobrecarga | 15 | 121.59 |
| Total | 100 | **810.6** |

### 3.6 Situación actual de pago.

|  |  |
| --- | --- |
| **Organización (Sector)** | **Pago/mensual (Moneda Nacional)** |
| XETID (Estatal) | 15 000 |
| DATYS (Estatal) | 18 000 |
| DESOFT (Estatal) | 15 000 |
| MYPIMES (Privado) | 50 000 |
| Freelancer o persona autónoma (Privado) | 500 MLC a un camvbio de 123 CUP (Tasa de Cambio de CADECA) serían 61 500 cup |