

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 las investigaciones del estudiante Ernesto Duvalon 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, we address the continuation of student Ernesto Duvalon's research on creating a system that allows the development of virtual assistants using the RASA software development marco de trabajo and the integration of natural language processing tools, such as Spacy, to enhance the assistants' ability to understand and respond to user questions and requests. Additionally, voice recognition technologies were implemented to convert audio to text and enable the virtual assistant to interact with users 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 6](#_Toc129849672)

[CAPITULO 1 . MARCO TEÓRICO 11](#_Toc129849673)

[1.1 Procesamiento del Lenguaje Natural 11](#_Toc129849674)

[1.2 Asistente Personal Inteligente 11](#_Toc129849675)

[1.3 Reconocimiento Automático del Habla 12](#_Toc129849676)

[1.4 Estado del arte de las herramientas empleadas en el proceso de desarrollo 13](#_Toc129849677)

[1.4.1 DialogFlow 13](#_Toc129849678)

[1.4.2 Amazon Lex 14](#_Toc129849679)

[1.4.3 IBM Watson 15](#_Toc129849680)

[1.4.4 RASA marco de trabajo 16](#_Toc129849681)

[1.4.5 Conclusiones de las herramientas y marcos de trabajo analizados 17](#_Toc129849682)

[1.5 Herramientas y Lenguajes de Programación 18](#_Toc129849683)

[Python 18](#_Toc129849684)

[Visual Studio Code 19](#_Toc129849685)

[RASA 20](#_Toc129849686)

[Whisper 20](#_Toc129849687)

[Mongo DB 20](#_Toc129849688)

[FastAPI 21](#_Toc129849689)

[Git 22](#_Toc129849690)

[1.6 Metodología de desarrollo de Software 22](#_Toc129849691)

[SCRUM 22](#_Toc129849692)

[Conclusiones del capítulo 23](#_Toc129849693)

[CAPITULO 2 . ORGANIZACIÓN Y DISEÑO 24](#_Toc129849694)

[2.1 Sprint Backlog 24](#_Toc129849695)

[2.2 Arquitectura del Sistema 25](#_Toc129849696)

[2.3 Requisitos no Funcionales 25](#_Toc129849697)

[2.3.1 Requisitos de Software 25](#_Toc129849698)

[2.3.2 Requisitos de Hardware 25](#_Toc129849699)

[2.4 Diagrama de Secuencia 26](#_Toc129849700)

[2.5 Diagrama de Actividades del Asistente 26](#_Toc129849701)

[2.6 Diseño de la interfaz 27](#_Toc129849702)

[CAPITULO 3 . IMPLEMENTACIÓN Y PRUEBA 28](#_Toc129849703)

[3.1 Instalación de las herramientas empleadas 28](#_Toc129849704)

[CONCLUSIONES Y RECOMENDACIONES 30](#_Toc129849705)

[REFERENCIAS BIBLIOGRÁFICAS 31](#_Toc129849706)

[GLOSARIO DE TÉRMINOS 35](#_Toc129849707)

[ANEXOS 36](#_Toc129849708)

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 creciente población de personas ancianas, discapacitadas, con problemas visuales y niños enfrenta desafíos significativos en términos de atención y comunicación. La falta de sistemas de apoyo adaptados a las necesidades específicas de estos grupos ha llevado a dificultades en la obtención de información precisa y en la interacción efectiva con tecnologías digitales. Una consecuencia de esto es el aumento de la brecha en la comunicación, limitando la capacidad de estas personas para recibir respuestas personalizadas y adaptadas a su estado emocional.

**Objeto de estudio:**

Procesamiento del lenguaje natural en la creación de asistentes virtuales.

**Campo de Estudio:**

Asistentes Virtuales

**Campo de Acción:**

Abarca la investigación y desarrollo de algoritmos de procesamiento de lenguaje natural, reconocimiento de voz, análisis de video y sistemas de inteligencia artificial que permitan la interacción efectiva entre las personas vulnerables y los asistentes virtuales. Además, se busca diseñar interfaces de usuario accesibles y amigables que faciliten la interacción para aquellos con limitaciones visuales o motoras.

**Objetivo General:**

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

**Objetivos Específicos:**

Análisis del estado del arte.

Estudio de la investigación de José Ernesto Duvalón

Modificación del algoritmo de generación de preguntas y respuestas

Integración de un modelo largo del lenguaje

Análisis y diseño del proceso de integración y creación de la plataforma

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

Implementación de la plataforma

* Pruebas y despliegue de la plataforma

**Hipótesis:**

Se desarrollará 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, con el objetivo de mejorar la interacción con los usuarios y ofrecer una experiencia más personalizada y satisfactoria en la resolución de dudas y consultas. Esta herramienta permitirá 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

La creación de asistentes virtuales se ha convertido en una tendencia cada vez más frecuente para aquellos que buscan aprovechar la tecnología para ofrecer una mejor experiencia de usuario. Estas herramientas permiten a los usuarios interactuar con la maquina a través de una conversación lo que les permite obtener información de forma sencilla y eficaz. En esta sección se examinarán algunas de las principales herramientas de construcción de asistentes virtuales, conceptos importantes del contexto que se está trabajando, por último, se explicará cual es la mejor opción para utilizarse en este proyecto y el porqué.

## 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 destacadas como el lingüista Noah 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 tiempo atrás 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.

Rasa marco de trabajo es una herramienta de código abierto basada en Python que permite la creación de chatbots y asistentes virtuales con inteligencia artificial. Esta herramienta es altamente personalizable y ofrece una gran flexibilidad en cuanto a la implementación de algoritmos de procesamiento de lenguaje natural y la integración con otros sistemas. Además, Rasa marco de trabajo cuenta con una gran comunidad de desarrolladores que contribuyen constantemente a su mejora y actualización.

### 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)...

Si se realiza un análisis crítico sobre todas estas plataformas y cual de estas se ajusta a los requerimientos del proyecto en cuestión, RASA sería la mejor opción a elegir, dado que DialogFlow pese a que tiene una capa gratuita, limitaría el proyecto a la hora de que el usuario quisiera hacer peticiones sobre este, además la administración del asistente debe ser en la propia plataforma. En el caso de Amazon Lex se desecha la idea dado que los asistentes que se creen tienen que ser desplegados obligatoriamente en Amazon Web Services, lo cual estaría bien en su capa gratuita, pero esta solamente dura un año y pide tarjeta de crédito extranjera para poder registrarse en la plataforma AWS. Por parte de IBM Watson presenta desventajas en el proyecto dado que es una herramienta muy cara, la documentación no es tan abundante y este solo permite su interacción en inglés. Por último Botpress pese a no tener una gran curva de aprendizaje y tiene integración con ChatGPT, los bots creados con la plataforma se mantienen en la plataforma, es decir, no se podría hacer un host personalizado para el asistente, además solo permite la integración con ChatGPT y el precio de Botpress se basa en la cantidad de solicitudes mensuales, luego de exceder las 1000 gratuitas se debe pagar 0.005 dólares por cada una.

## 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 [13].

Whisper es una herramienta valiosa para realizar transcripciones con un alto nivel de precisión. Su capacidad para manejar acentos y ruido de fondo, así como su precisión en la transcripción y su uso de forma local, lo hacen especialmente útil para este proyecto.

## 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.

### HTML, CSS y Javascript

HTML, CSS y JavaScript son las tres tecnologías fundamentales que se utilizan para desarrollar y diseñar sitios web. Cada una de estas tecnologías tiene un propósito específico y se complementan entre sí para crear experiencias web completas y atractivas.

HTML (HyperText Markup Language) es el lenguaje de marcado estándar para documentos diseñados para ser mostrados en un navegador web. HTML proporciona la estructura básica de una página web, definiendo elementos como encabezados, párrafos, enlaces, imágenes y más. HTML se utiliza para estructurar el contenido de una página web, pero no controla la presentación visual de esa estructura (Learn Web Development Basics – HTML, CSS, and JavaScript Explained for Beginners, s. f.)

CSS (Cascading Style Sheets) es un lenguaje de hojas de estilo utilizado para describir la apariencia de un documento escrito en HTML. CSS se utiliza para controlar el diseño visual, incluyendo colores, fuentes, espaciado, y más. A diferencia de HTML, que se centra en la estructura y el contenido, CSS se centra en cómo se ve y se siente esa estructura y contenido

JavaScript es un lenguaje de programación que se utiliza para hacer que las páginas web sean interactivas. JavaScript puede cambiar el contenido y el estilo de una página web, reaccionar a las acciones del usuario (como clics o movimientos del mouse), y comunicarse con el servidor para recuperar o enviar datos. JavaScript se ejecuta en el navegador del usuario, lo que significa que puede hacer que las páginas web sean dinámicas y atractivas

### 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 para el desarrollo de la aplicación web 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 los requisitos de este proyecto. Por último, gracias a su alta popularidad, la mayoria 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**

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**

astAPI 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:

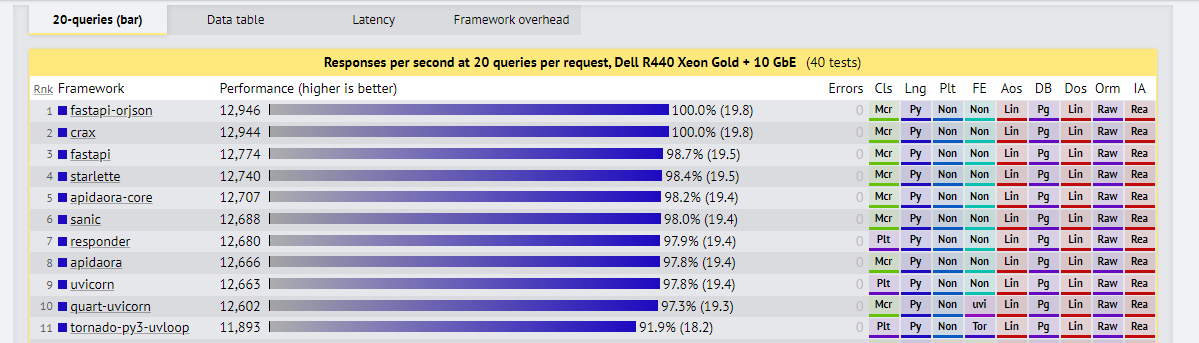
FastAPI es un marco de trabajo moderno y de alto rendimiento para construir APIs con Python 3.6+. Destaca por su documentación automática, que genera una interfaz interactiva para explorar y probar la API, y por el uso de declaraciones de tipos de Python 3.6 para proporcionar un soporte de editor excelente 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:

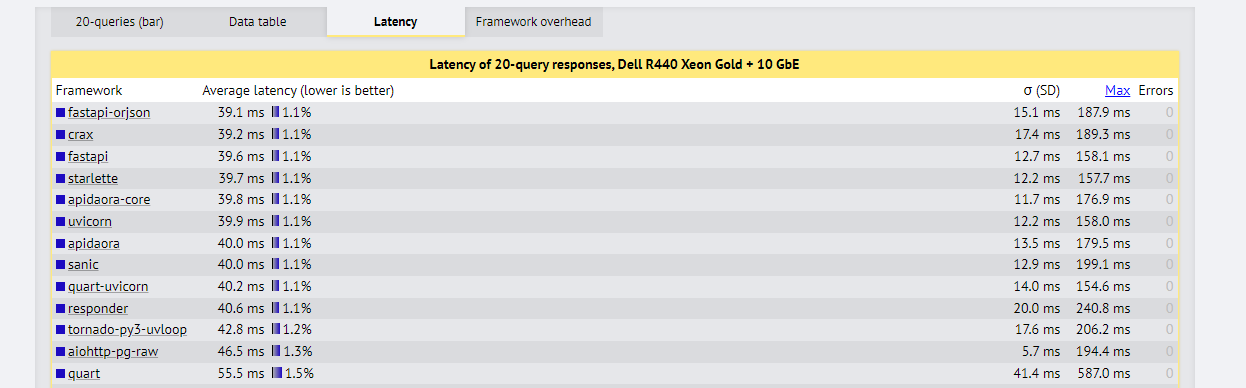
Pese a su gran robustez, es un marco de trabajo relativamente nuevo, dado esto es más propenso a que no se use con frecuencia. FastAPI aún está en desarrollo y hay muchas cosas que se pueden mejorar. Aunque su comunidad está creciendo, todavía no es comparable a las comunidades de otros marco de trabajos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

**Benchmark 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)



Como bien se puede observar en el primer y tercer lugar se encuentra FastApi mostrando su alta velocidad en el manejo de concurrencia, ligereza y escasa latencia al realizar las consultas.



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 benchmark 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.

### 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, 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 chatbot, 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.

Si comparamos a Llama 2 con Falcon, como bien se observa en la siguiente foto, en el caso de los modelos de 7Billones de parámetros Llama tiene casi el doble de puntuación en MMLU (Massive Multitask Language Understanding) métrica que se utiliza para medir cuán bien un modelo de lenguaje a gran escala (LLM) comprende el lenguaje y puede resolver problemas con 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

Por último, el motivo de peso por el cual elijo a Llama 2 para el desarrollo de esta tesis es que se le puede realizar fine-tuning para lograr que el modelo realice el proceso de generación de preguntas y respuestas con una mayor precisión. Además, este proceso se podría llevar a cabo de forma local, ya que los requisitos de hardware de este modelo permiten ejecutarlo sin necesidad de una GPU. Estos dos aspectos son importantes, ya que los modelos anteriormente mencionados carecen de ellos, algunos modelos no permiten el fine-tuning por su licencia, mientras que otros son muy costosos en cuanto a recursos de hardware

### Base de Datos utilizada

El uso de una base de datos NoSQL para este proyecto se justifica por su capacidad para manejar grandes volúmenes de datos de manera eficiente, su flexibilidad para adaptarse a cambios rápidos, su escalabilidad para soportar picos de tráfico y su tolerancia a fallas, por tanto se elige la base de datos MongoDB.

* **Escalabilidad**: MongoDB es una base de datos distribuida que soporta el sharding automático. Esto significa que puede manejar grandes cantidades de datos y tráfico de manera eficiente, lo cual es beneficioso para un proyecto de gestión de asistentes virtuales que podría crecer con el tiempo (Postgres vs. MongoDB, 2023).
* **Modelo de datos flexible**: MongoDB utiliza un modelo de datos basado en documentos, lo que permite almacenar datos no estructurados, semi-estructurados o incluso estructurados en la misma colección. Esto puede ser útil para tu proyecto, ya que los asistentes virtuales pueden tener diferentes tipos de preguntas y respuestas, y las necesidades de los usuarios pueden variar (MongoDB vs PostgreSQL: 15 Critical Differences, s. f.-a)
* **Rendimiento**: MongoDB utiliza BSON (Binary JSON) para almacenar datos, lo que facilita la lectura y escritura de datos. Además, MongoDB soporta la creación de índices en campos en documentos y subdocumentos, lo que puede acelerar las consultas de datos (MongoDB Vs PostgreSQL, s. f.).
* **Transacciones**: MongoDB ofrece transacciones multi-documento, lo que significa que puedes realizar cambios en múltiples documentos como una sola operación atómica. Esto puede ser útil para tu proyecto si necesitas realizar operaciones que involucran a múltiples asistentes virtuales o usuarios a la vez  (MongoDB Vs PostgreSQL, s. f.).
* **Costo**: MongoDB es una opción más económica en comparación con PostgreSQL, especialmente si tu proyecto crece y necesita más recursos. MongoDB también ofrece una opción de nube, MongoDB Atlas, que puede facilitar la administración de tu base de datos (MongoDB vs PostgreSQL: 15 Critical Differences, s. f.-b).
* **Facilidad de uso**: MongoDB tiene una sintaxis de consulta más intuitiva y fácil de usar en comparación con SQL, lo que puede facilitar el desarrollo de tu 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.

### 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 ultimo se centra en entregar constantemente software de alta calidad.

Conclusiones del capítulo

En conclusión, en este capítulo de marco referencial ofreció una base teórica y práctica sólida para el desarrollo del sistema de creación de asistentes, se han abordado diversos temas relacionados con el desarrollo del proyecto, desde el procesamiento del lenguaje natural hasta el estado del arte de las herramientas empleadas en su desarrollo. Además, se han presentado herramientas y lenguajes de programación como Python, Visual Studio Code, RASA, React JS, FastAPI y Mongo DB, entre otros, que son fundamentales para el desarrollo de este proyecto. Asimismo, se ha explicado la metodología de desarrollo de software SCRUM, que será utilizada para garantizar un proceso eficiente y efectivo en el desarrollo del proyecto.

# ORGANIZACIÓN Y DISEÑO

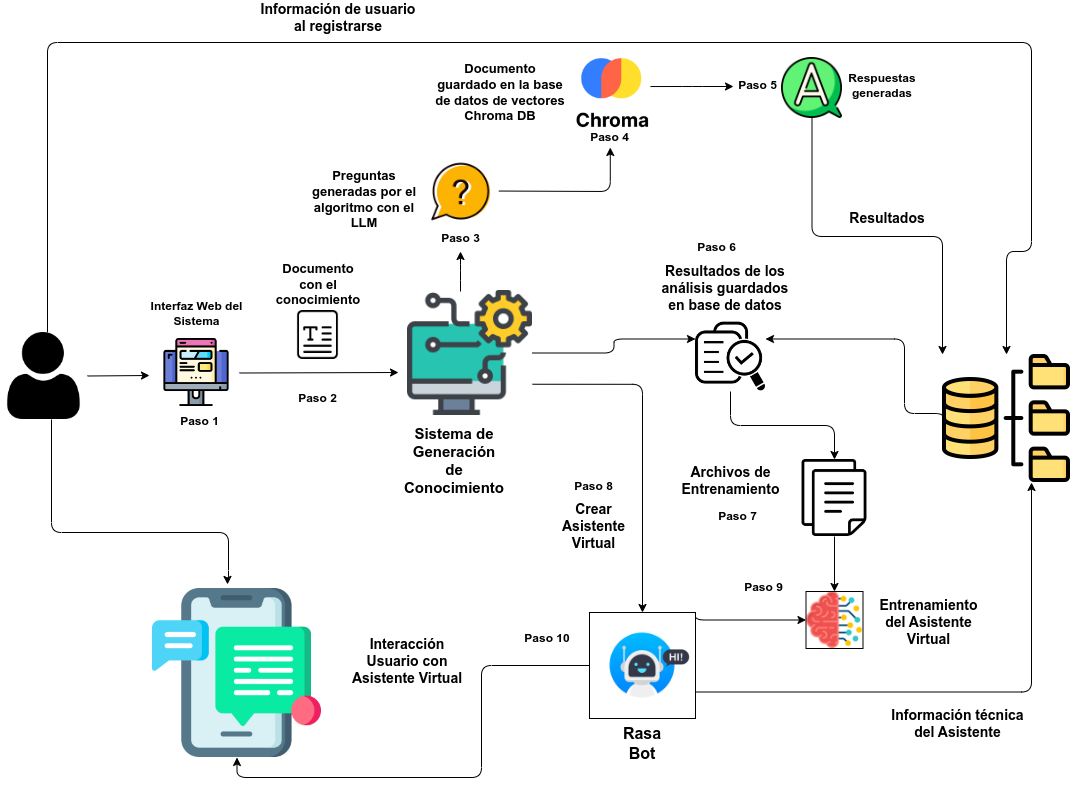
En este capítulo, se abordará el proceso de planificación y diseño del sistema, en el cual se han utilizado diferentes técnicas y herramientas para lograr el objetivo de esta investigación. Se presentarán los diagramas que se han utilizado para el modelado y documentación del sistema, así como la planificación 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.

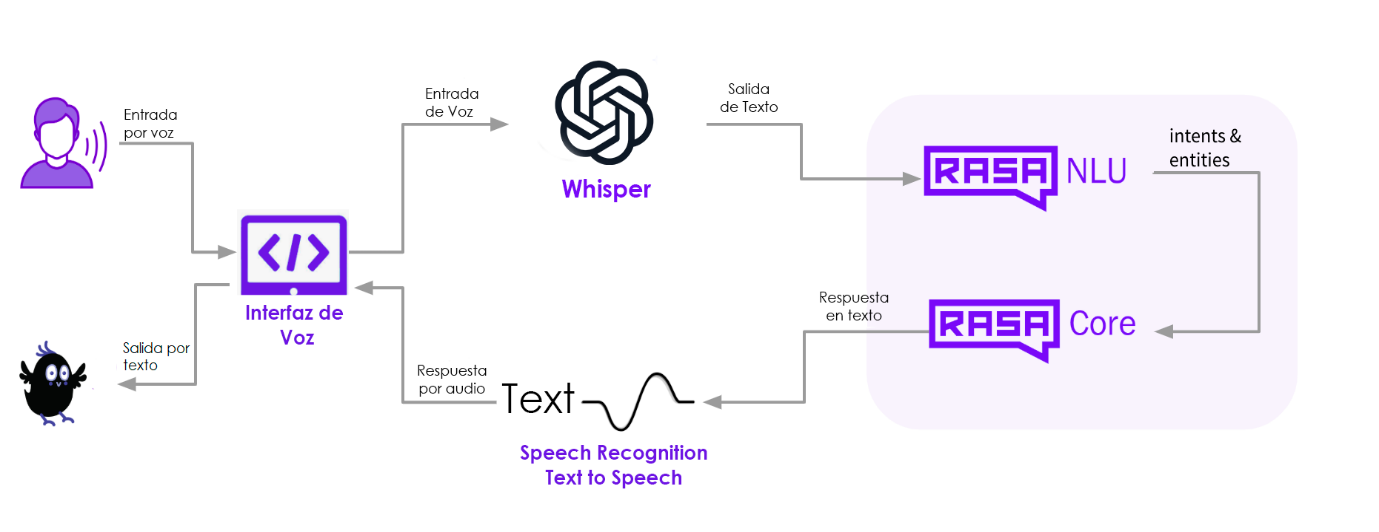
## Arquitectura del sistema

En la [figura 6](#_Diagrama_d) se muestra el diagrama de 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 usando Langchain (el cual ofrece herramientas para la ingeniería de prompts 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.



### Diagrama de la arquitectura de el 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.



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

## Requisitos Funcionales usando metodología RUP

Aunque el proyecto se adhiere a la metodología XP, la inclusión de Requisitos Funcionales en el proceso de documentación es una práctica altamente beneficiosa. Esta práctica no solo mejora la claridad y la comunicación dentro del equipo de desarrollo, sino que también contribuye a la entrega de un producto de alta calidad.

### Interfaz de Usuario

1. Registro e Inicio de Sesión: El sistema permitirá a los usuarios registrarse utilizando su correo electrónico y también iniciar sesión utilizando su correo electrónico, cuenta de Google o cuenta de Facebook.
2. Generación de Preguntas y Respuestas: El sistema generará preguntas y respuestas a partir de un contexto dado.
3. Gestión de Asistentes Virtuales: Los usuarios podrán crear, editar, visualizar, buscar, eliminar y probar asistentes virtuales.

### Servidor

1. Registro e Inicio de Sesión: El sistema permitirá a los usuarios registrarse utilizando su correo electrónico y también iniciar sesión utilizando su correo electrónico, cuenta de Google o cuenta de Facebook.
2. Generación y Evaluación de Preguntas: El sistema generará preguntas y evaluará las preguntas generadas.
3. Generación de Respuestas: El sistema generará respuestas a partir de las preguntas generadas.
4. Gestión de Asistentes Virtuales: Los usuarios podrán crear, editar, eliminar y probar asistentes virtuales.

La generación de preguntas y respuestas se considera una sola función porque está intrínsecamente relacionada. La generación de preguntas es el primer paso para obtener respuestas relevantes. Por lo tanto, agrupar estas funciones simplifica la estructura y evita la repetición de conceptos similares.

***Los requisitos funcionales antes descritos se convierten en historias de usuario***

## Requisitos No Funcionales

### Software

|  |  |
| --- | --- |
| **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***

### Hardware

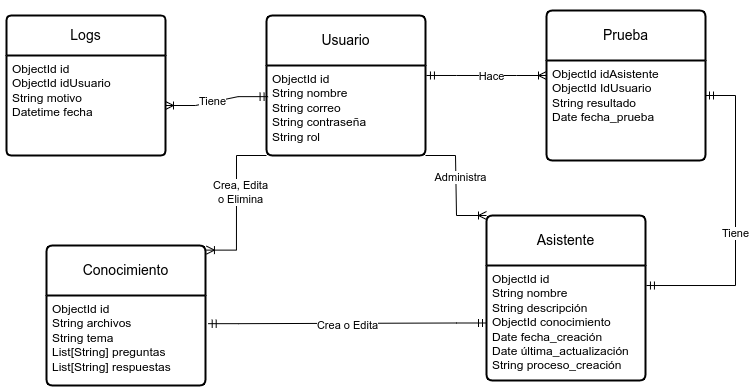
|  |  |
| --- | --- |
| **Componente** | **Requisito** |
| Procesador | Procesador Intel Core i3 de 7ma generación con al menos 4 núcleos y 6 hilos. |
| Memoria RAM | Se recomienda tener al menos 8 GB de RAM debido a los requisitos del modelo del lenguaje utilizado. |
| Almacenamiento | Se necesita aproximadamente 20 GB de almacenamiento debido al peso de los modelos del lenguaje y de Whisper por su caché. |

***Se recomienda para lograr un mejor rendimiento en la ejecución de las tareas de whisper y de los modelos del lenguaje el uso de GPU, con una tarjeta gráfica de al menos 4 GB DDR5 por ejemplo la tarjeta Nvidia GTX 960***

## Modelo Entidad Relación (MER)

Pese a que en este proyecto se va a utilizar una base de datos NoSQL, es buena práctica crear un modelo entidad relación que refleje la información que se va a manejar en la base de datos, esto es muy importante a la hora de realizar la documentación del sistema.

Esta base de datos se encuentra en la Tercera Forma Normal (3FN). Cada atributo depende directamente de la clave primaria de su entidad. No hay dependencias transitivas ni grupos repetidos de atributos.



## *Modelo Entidad Relación*

## Historias de Usuario

A continuación se describen 3 de las principales historias de usuario derivadas del proceso de levantamiento de requisitos funcionales.

|  |  |  |
| --- | --- | --- |
| **Historia de usuario** | | |
| **Número: 1** | **Nombre de HU:** Autenticación de Usuario en el Sistema | |
| **Usuario:** Cualquier usuario | | |
| **Prioridad en negocio:** Alta | **Riesgo de desarrollo:** Bajo | |
| **Descripción:** 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 | | |
|  | | **Interacciones del Usuario:**   * El usuario accede a la página de inicio de sesión. * Encuentra un formulario que le pide su dirección de correo electrónico y contraseña. * Ingresa su correo electrónico y contraseña. * Hace clic en el botón de "Iniciar sesión". * Si las credenciales son correctas, el sistema lo redirige a su cuenta personal. * Si las credenciales son incorrectas, el sistema muestra un mensaje de error y le permite al usuario intentarlo nuevamente o restablecer su contraseña. |

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número:1** | **Nombre de HU:** Creación de asistente |
| **Usuario:** Usuario autenticado | |
| **Prioridad en negocio:** Alta | **Riesgo de desarrollo:** Alto |
| **Descripción:** 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. | |
|  | |
| **Interacciones del Usuario:**   * Rellenar formulario inicial: El usuario completa un formulario para proporcionar la información necesaria para analizar un documento. * Generar preguntas y respuestas automáticas: Basado en la información del formulario, el sistema genera automáticamente preguntas y respuestas relacionadas con el documento. * Revisión y edición opcional: El usuario tiene la opción de revisar y editar las preguntas y respuestas generadas en un formulario de edición. * Generar archivos de entrenamiento: Una vez satisfecho con las preguntas y respuestas, el usuario puede generar archivos de entrenamiento para su asistente virtual. * Guardar resultados: El usuario tiene la opción de guardar los resultados para entrenar el asistente en otro momento. | |

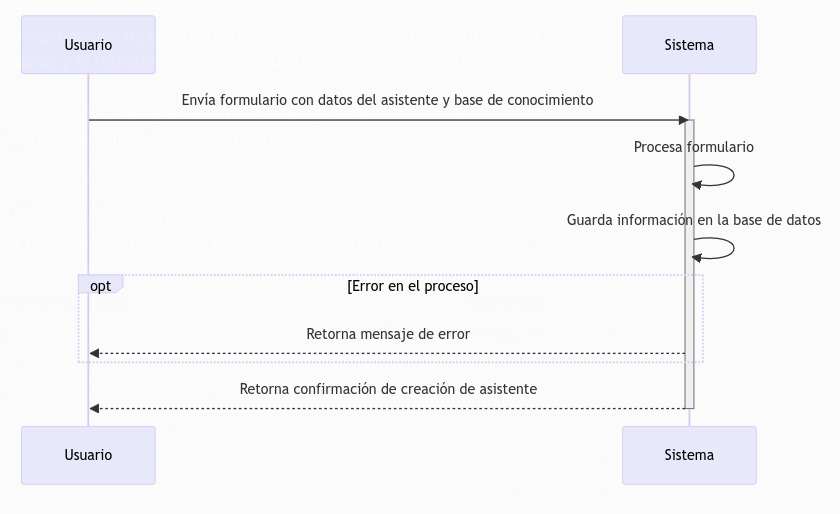
|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número:1** | **Nombre de HU:** Administrar asistentes |
| **Usuario:** Usuario autenticado | |
| **Prioridad en negocio:** Alta | **Riesgo de desarrollo:** Alto |
| **Descripción:** Donde se edita, prueba y se eliminan asistentes, también se pueden buscar asistentes por cualquiera de los parámetros que tiene la tabla | |
|  | |
| **Interacciones del Usuario:**   * Editar asistente: Modificar información de un asistente existente. * Probar asistente: Ejecutar pruebas o simulaciones con un asistente para verificar su funcionamiento. * Eliminar asistente: Eliminar un asistente de la lista. * Buscar asistente: Encontrar un asistente específico usando cualquiera de los parámetros disponibles en la tabla. | |

## Usuarios del Sistema

|  |  |
| --- | --- |
| 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. |

## Diagrama de Secuencia de creación de un asistente

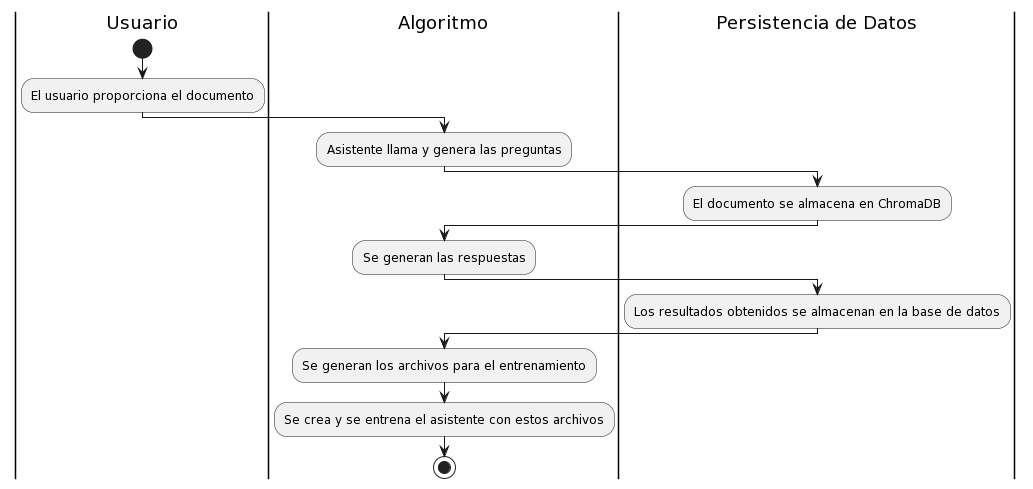
En el flujo el usuario envía inicialmente los datos del asistente a crear junto con la base de conocimientos correspondiente. Posteriormente, el sistema procesa estos datos, almacenándolos en la base de datos. Si surge algún error durante este proceso, se notifica al usuario; de lo contrario, se confirma la creación exitosa del asistente.



## *Diagrama de Secuencia de creación de un asistente*

## Diagrama de Actividades de Creación de un Asistente

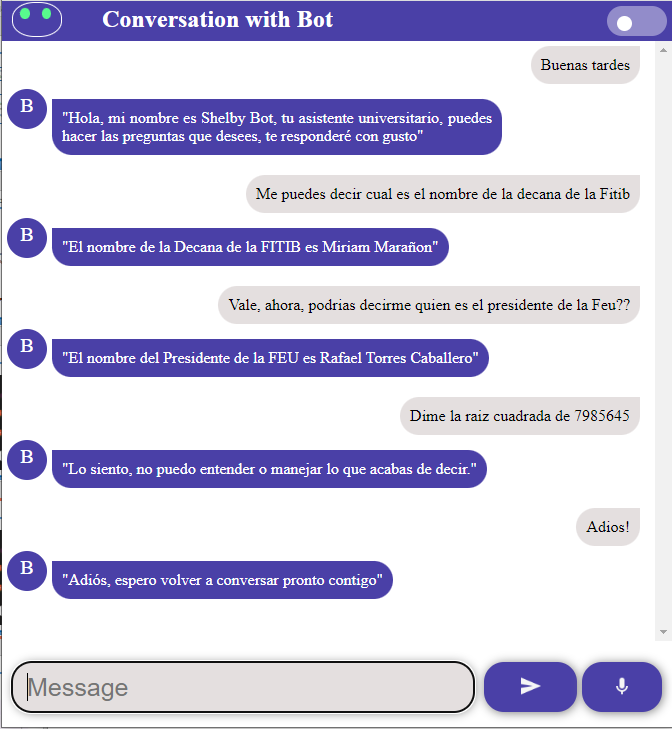
El proceso de creación de un asistente comienza cuando el usuario proporciona un documento relevante para la generación de preguntas y respuestas. Luego, un algoritmo especializado, representado como "Asistente", se activa para analizar el contenido del documento y generar las preguntas pertinentes. Una vez que se generan las preguntas, el documento se almacena en ChromaDB, una base de datos especializada en este tipo de contenido. Posteriormente, el mismo algoritmo 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.



## *Diagrama de Actividades de Creación de un Asistente*

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

Cada asistente tiene una interfaz gráfica para la interacción, que puede ser reemplazada por una creada por el usuario. La lógica de conexión puede aplicarse en otras interfaces.



## *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, 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. Estos fragmentos se convierten en documentos y se pasan a una cadena de procesamiento que utiliza un modelo de lenguaje (LlamaCpp) 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,

),

*# llm=llm,*

*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}")

return answers

### Algoritmo de Reconocimiento del Habla y Transcripción

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)

start = time.time()

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

## Análisis económico

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 se presenta el 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** |

## 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. A continuación se muestra la tabla con la clasificación de complejidad y el cálculo del peso:

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

**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. En la siguiente [tabla 3.3](#_Tabla_3.3_Factor) se muestra factor de complejidad técnica con su significado y el peso de cada uno de estos factores:

#### Tabla 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** |

**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 siguiente [tabla 3.4](#_Tabla_3.4_Factor) se muestra factor de ambiente con su significado y el peso de cada uno de estos factores:

#### Tabla 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** |

**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](#_Tabla_3.5_Distribución) se muestra distribución del esfuerzo por etapas.

#### Tabla 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** |

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.

En nuestro caso como empleamos 1 trabajador para el desarrollo del proyecto, el tiempo para su culminación quedará reducido a 30 días aproximadamente.

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 se mueve 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.

Tabla 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 123 CUP (Tasa de Cambio de CADECA) serían 61 500 cup |

## 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.

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 | Campos válidos y archivos de conocimiento proporcionados | El asistente virtual se crea correctamente y se muestra en la lista de asistentes | Todos los campos requeridos están completos y los archivos de conocimiento son válidos |
| CP2 | Campos incompletos | El sistema muestra un mensaje de error indicando los campos que faltan por completar | Al menos uno de los campos obligatorios no está lleno |
| CP3 | Archivos de conocimiento inválidos | El sistema muestra un mensaje de error indicando que los archivos de conocimiento no son válidos | Los archivos de conocimiento no tienen el formato correcto |

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

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

## Conlusiones del capítulo

En este capítulo se proporciona una visión detallada de la implementación y prueba del sistema propuesto, destacando los aspectos técnicos, económicos y de calidad del proyecto. La guía de instalación, la descripción de algoritmos clave y la validación del sistema a través de pruebas son elementos cruciales para demostrar la viabilidad y eficacia de la solución desarrollada.

CONCLUSIONES Y RECOMENDACIONES

**Conclusiones**

Las conclusiones deben estar en correspondencia con los objetivos planteados. Ellos deben referirse al problema de investigación planteado, de hecho la primera conclusión de un estudio es evaluar qué ocurrió con el planteamiento. Deben ser objetivas, breves, precisas y convincentes. Es importante destacar que las conclusiones no son un recuento de lo que se realizó en el trabajo. La conclusión constituye una generalización o comentario aplicado,.relativa a los resultados obtenidos

Para una mejor comprensión pueden ser numeradas y si algún caso lo requiere, pueden ser comentadas. Ejemplos

1. Se comprobó que el efecto del jitter sobre la BER no es significativo cuando la velocidad de transmisión no excede los 4 Mbps
2. Se verificó que …
3. Se demostró que …
4. Conclusión 3

**Recomendaciones**

Escriba lo que usted recomienda a otros investigadores para mejorar su trabajo o para determinar otros datos que usted no pudo lograr, sugerir qué hacer con sus resultados y aportes.

Recomendación 1

Recomendación 2

Recomendación 3

No debería haber muchas recomendaciones pues si uno recomienda muchas acciones la lectura sería que no se avanzó mucho en la investigación.

La extensión de estos aspectos no debe exceder de cinco páginas.

REFERENCIAS BIBLIOGRÁFICAS

5 learnings from our ‘Chatbot Survey — 2017’ | by Mindbowser | Chatbots Journal. (s. f.). Recuperado 2 de noviembre de 2023, de https://chatbotsjournal.com/5-learnings-from-our-chatbot-survey-2017-72a6a4fc209c?gi=21ae27bf4d12  
 10 principales API de Reconocimiento Facial - Infoteknico. (2023, octubre 6). https://www.infoteknico.com/principales-api-de-reconocimiento-facial/  
 Asistente virtual: retos y posibilidades para la inteligencia artificial | Tableau. (s. f.). Recuperado 2 de noviembre de 2023, de https://www.tableau.com/data-insights/ai/ai-virtual-assistant  
 AskLua. (2020, noviembre 6). 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  
 Build And Host Fast Data Science Applications Using FastAPI | by Farhad Malik | Towards Data Science. (s. f.). Recuperado 2 de noviembre de 2023, de https://towardsdatascience.com/build-and-host-fast-data-science-applications-using-fastapi-823be8a1d6a0  
 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.). Recuperado 2 de noviembre de 2023, de https://rcci.uci.cu/?journal=rcci&page=article&op=view&path%5B%5D=2289  
 Cómo Amazon Lex cionamiento cionamiento cionamiento cionamiento cionamiento cionamiento cionamiento cionamiento - Amazon Lex V1. (s. f.). Recuperado 2 de noviembre de 2023, de https://docs.aws.amazon.com/es\_es/lex/latest/dg/how-it-works.html  
 Comparing React, Angular, Vue, and Svelte - Accelebrate. (s. f.). Recuperado 2 de noviembre de 2023, de https://www.accelebrate.com/blog/react-angular-vue-svelte-comparison  
 Condez, A. J. (2022, febrero 13). Top 10 Big Companies Using Flask. Career Karma. https://careerkarma.com/blog/companies-that-use-flask/  
 Dialogflow de Google cambia su modelo de pricing: ¿cuánto cuesta la IA / NLU? - Planeta Chatbot. (2017, diciembre 12). https://planetachatbot.com/dialogflow-google-cambia-modelo-pricing/  
 Explicación de FastAPI en 5 minutos o menos - Geekflare. (s. f.). Recuperado 7 de octubre de 2023, de https://geekflare.com/es/fastapi-explained/  
 He usado Whisper para transcribir una entrevista: es la herramienta que llevaba esperando desde hace años. (s. f.). Recuperado 7 de octubre de 2023, de https://www.xataka.com/aplicaciones/he-usado-whisper-para-transcribir-entrevista-herramienta-que-llevaba-esperando-hace-anos  
 Hernández, U. (2023, febrero 1). Cómo elegir un framework para el backend. CódigoFacilito. https://codigofacilito.com/articulos/elegir-framework-backend  
 IBM watsonx Assistant Virtual Agent. (s. f.). Recuperado 2 de noviembre de 2023, de https://www.ibm.com/products/watsonx-assistant  
 Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). 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  
 Learn Web Development Basics – HTML, CSS, and JavaScript Explained for Beginners. (s. f.). Recuperado 2 de noviembre de 2023, de https://www.freecodecamp.org/news/html-css-and-javascript-explained-for-beginners/  
 Llama 2 Vs GPT-3.5 Vs GPT-4: What, When & How To Chose. (2023, octubre 5). Labellerr. https://www.labellerr.com/blog/llama2-vs-gpt-3-5-vs-gpt-4/  
 Londoño, P. (s. f.). Qué es Python, para qué sirve y cómo se usa (+ recursos para aprender). Recuperado 2 de noviembre de 2023, de https://blog.hubspot.es/website/que-es-python  
 Luzniak, K. (2023, agosto 2). 6 main differences between Llama 2, GPT-3.5 & GPT-4. Neoteric. https://neoteric.eu/blog/6-main-differences-between-llama2-gpt35-and-gpt4/  
 Mancuzo, G. (2020, agosto 8). Metodología XP: La Mejor Vía para el Desarrollo de Software. Blog - ComparaSoftware. https://blog.comparasoftware.com/metodologia-xp/  
 Metodología XP o Programación Extrema: ¿Qué es y cómo aplicarla? (s. f.). Gestor de proyectos online. Recuperado 5 de diciembre de 2023, de https://www.sinnaps.com/blog-gestion-proyectos/metodologia-xp  
 MongoDB Vs PostgreSQL. (s. f.). MongoDB. Recuperado 2 de noviembre de 2023, de https://www.mongodb.com/compare/mongodb-postgresql  
 MongoDB vs PostgreSQL: 15 Critical Differences. (s. f.-a). Recuperado 2 de noviembre de 2023, de https://kinsta.com/blog/mongodb-vs-postgresql/  
 MongoDB vs PostgreSQL: 15 Critical Differences. (s. f.-b). Recuperado 2 de noviembre de 2023, de https://kinsta.com/blog/mongodb-vs-postgresql/  
 Oliveira, G. de. (2018). Redes Neurais Aplicadas No Desenvolvimento De Chatbots: Uma Análise Bibliométrica. https://www.academia.edu/72166314/Redes\_Neurais\_Aplicadas\_No\_Desenvolvimento\_De\_Chatbots\_Uma\_An%C3%A1lise\_Bibliom%C3%A9trica  
 Postgres vs. MongoDB: a Complete Comparison in 2023. (2023, julio 20). Bytebase. https://www.bytebase.com/blog/postgres-vs-mongodb/  
 Primeros pasos con Botpress | Botpress Blog. (s. f.). Recuperado 2 de noviembre de 2023, de https://botpress.com/es/blog/getting-started-with-botpress  
 Qué es Django y por qué usarlo. (2018, agosto 3). OpenWebinars.net. https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/  
 Qué es Git | Atlassian Git Tutorial. (s. f.). Recuperado 2 de noviembre de 2023, de https://www.atlassian.com/es/git/tutorials/what-is-git  
 ¿Qué es Python? - Explicación del lenguaje Python - AWS. (s. f.). Amazon Web Services, Inc. Recuperado 2 de noviembre de 2023, de https://aws.amazon.com/es/what-is/python/  
 React vs Vue vs Angular vs Svelte. (2020, noviembre 29). DEV Community. https://dev.to/hb/react-vs-vue-vs-angular-vs-svelte-1fdm  
 Sobrino, D. C. (2020, mayo 15). ¿Cómo funciona el reconocimiento automático del habla? SoldAI. https://medium.com/soldai/c%C3%B3mo-funciona-el-reconocimiento-autom%C3%A1tico-del-habla-eb038ecfe72e  
 Team, P. (2022, enero 14). Advantages of Django | Disadvantages of Django. Python Geeks. https://pythongeeks.org/advantages-disadvantages-of-django/  
 TechEmpower Web Framework Performance Comparison. (2023, febrero 1). Www.Techempower.Com. https://www.techempower.com/benchmarks/#section=data-r20&hw=ph&test=fortune&l=zijzen-sf  
 The AI race heats up: Google announces PaLM 2, its answer to GPT-4 | Ars Technica. (s. f.). Recuperado 2 de noviembre de 2023, de https://arstechnica.com/information-technology/2023/05/googles-top-ai-model-palm-2-hopes-to-upstage-gpt-4-in-generative-mastery/  
 Tsang, S.-H. (2023, agosto 5). Brief Review — Llama 2: Open Foundation and Fine-Tuned Chat Models. Medium. https://sh-tsang.medium.com/brief-review-llama-2-open-foundation-and-fine-tuned-chat-models-6666eb8b56b7  
 Visual Studio vs Visual Studio Code: ¿Cuál es la diferencia entre estos editores de código IDE? (2023, marzo 3). freeCodeCamp.org. https://www.freecodecamp.org/espanol/news/visual-studio-vs-visual-studio-code-cual-es-la-diferencia-entre-estos-editores-de-codigo-ide/  
 Xu, B. (2020, junio 9). Domain-Specific Bots vs. Generic Bots. DealerAI. https://dealerai.com/domain-specific-bots-vs-generic-bots/