



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

2024

Santiago de Cuba,

“Año 66 de la Revolución”

AGRADECIMIENTOS

A mi mamá y mi abuela, mis dos madres que han estado a mi lado en todo momento apoyándome, guiándome y dándome todo su amor.

A mi pareja por su cariño, por ayudarme a continuar creciendo personal y profesionalmente y estar ahí en los momentos difíciles.

A mi tutor por su constante guía, paciencia y siempre mantenerme creyendo que puedo aprender más, porque sí, yo puedo aprender más.

Al resto de mi familia por hacerme sentir querida.

A mis compañeros de aula, que me hicieron disfrutar de buenos momentos en estos años.

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 investigación se da solución a varias carencias y trabajos futuros propuestos en la tesis en opción al título de Ingeniero Informático, nombrada “Herramienta digital para la construcción de conocimiento automático para un Asistente Virtual” discutida en 2022, sobre la creación un sistema que permita crear asistentes virtuales utilizando el marco de desarrollo de software RASA. La principal mejora es la implementación del uso de tecnologías de reconocimiento de voz para convertir audio a texto y permitir al asistente virtual interactuar con el usuario a través de la voz.

El presente trabajo demuestra que es posible crear asistentes virtuales que permitan una interacción hombre – máquina más natural usando RASA, procesamiento del lenguaje natural y reconocimiento de voz con Whisper. Se evaluaron y entrenaron los asistentes, obteniendo resultados prometedores con respecto a la aceptación de los usuarios y 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 use is expected to continue growing in the future. These artificial intelligence systems are capable of understanding and responding to user questions and requests in a natural way, making them a useful tool for automating tasks and improving efficiency in various areas.

This research addresses several shortcomings and future works proposed in the thesis entitled "Digital Tool for the Construction of Automatic Knowledge for a Virtual Assistant," discussed in 2022, regarding the creation of a system that allows the development of virtual assistants using the RASA software development framework. The main improvement is the implementation of voice recognition technologies to convert audio to text and enable the virtual assistant to interact with the user through voice.

This work demonstrates that it is possible to create virtual assistants that allow for more natural human-machine interaction using RASA, natural language processing, and voice recognition with Whisper. The assistants were evaluated and trained, obtaining promising results in terms of user acceptance and confirming the viability of this technology.

Keywords: Virtual Assistant, RASA, Voice Recognition, Whisper

ÍNDICE

INTRODUCCIÓN.....	7
CAPITULO 1 . MARCO TEÓRICO	13
Procesamiento del Lenguaje Natural.....	13
Asistente Personal Inteligente.....	14
Reconocimiento automático del habla.....	14
Estado del arte de las herramientas utilizadas para el desarrollo de la plataforma.....	15
Herramientas y plataformas para la creación de asistentes virtuales.....	15
DialogFlow.....	15
Amazon Lex.....	16
IBM Watson.....	16
RASA.....	17
BotPress.....	17
Herramienta de reconocimiento y transcripción de voz a texto.....	18
Herramientas y tecnologías empleadas en la plataforma propuesta en la investigación.	19
Marco de Trabajo utilizado en el módulo orientado al cliente (Interfaz de Usuario)...	19
Herramienta para el reconocimiento de emociones faciales.....	20
Marco de trabajo utilizado en el módulo de servicios.....	21
Lenguajes de programación utilizados.....	21
Python.....	21
Marco de trabajo Django.....	22
Herramientas para la creación del algoritmo de generación de preguntas y respuestas	23
Langchain.....	23
Base de datos Vectorial.....	24
Modelo de Lenguaje para la generación de preguntas y respuestas.....	25
Base de Datos utilizada.....	27
Entorno de desarrollo y herramientas utilizadas para la implementación de los módulos de la plataforma.....	28
Sistema de Control de Versiones GIT versión 2.44.0.....	28
Metodología de Desarrollo de Software XP.....	29
Conclusiones del capítulo.....	30
CAPITULO 2 . ORGANIZACIÓN Y DISEÑO	31

Propuesta de Sistema.....	31
Flujo del Sistema.	32
Historias de Usuario.....	34
Autenticar al Usuario en el Sistema.....	35
Crear asistente virtual con información inicial.....	35
Visualizar asistentes creados.	36
Requerimientos para el despliegue de la plataforma.....	37
Software.....	37
Hardware.	38
Diseño de Base de Datos.....	38
Usuarios del Sistema.....	39
Tabla de usuarios del sistema.	39
Diagrama de Secuencia del reconocimiento de emociones del usuario.....	39
Diagrama de Actividades del reconocimiento de voz.....	40
Diseño de la interfaz del Chat para probar el asistente.....	41
Conclusiones del capítulo.....	42
CAPITULO 3 . IMPLEMENTACIÓN Y PRUEBA.....	43
Patrones de Diseño utilizados.....	43
Diagrama de Clases.....	44
Algoritmos Importantes.....	44
Algoritmo para la creación de preguntas	44
Algoritmo para la creación de las respuestas a las preguntas previamente generadas	
.....	47
Algoritmo de Reconocimiento del habla y transcripción.....	48
Interfaz gráfica resultante	49
Pruebas al sistema	50
Encuestas a usuarios.....	52
Análisis económico del costo de producción del sistema.....	53
Estimación de costo y tiempo.....	53
Conclusiones del capítulo.....	56
CONCLUSIONES Y RECOMENDACIONES.....	57
REFERENCIAS BIBLIOGRÁFICAS.....	59

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 (Duvalón, 2022).

Los asistentes virtuales o también conocidos como chatbots 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: primero, comprensión del lenguaje natural (NLU; siglas en inglés) que obtiene las intenciones del usuario; segundo, administración de diálogo que monitorea el sistema actual y el estado de la conversación y tercero, generación de lenguaje natural que responde al usuario (Mindbowser 2018).

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” (Mindbowser 2018), donde se popularizó en gran medida su uso. Existen varias plataformas líderes en la creación de chatbots en la nube virtual (ed., servicios accesibles desde internet) como: 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.

Los asistentes virtuales fomentan una conversación bidireccional, mejorando la eficiencia temporal y la reducción de costos para las empresas. Estos sistemas optimizan procesos comerciales, reorganizando recursos y ahorrando potencialmente más de \$8 mil millones anuales en costos de soporte al cliente para 2022 (Caballero, 2021).

Es importante mencionar que existen dos tipos principales de chatbots: los de dominios específicos de conocimiento y los de dominios generales, los de dominios específicos de conocimiento son aquellos que han sido diseñados para trabajar en un área particular de conocimiento o campo de especialización. Por ejemplo, un chatbot de dominio específico de conocimiento en medicina podría responder preguntas sobre síntomas, enfermedades y tratamientos médicos (Caballero, 2021).

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).

En Cuba el uso de chatbots aumenta con la adopción de la transformación digital en varios sectores de la sociedad. Sin embargo, la mayoría de los servicios de chatbots existentes no permiten la interacción con usuarios de forma fluida; la creación y mejora de este tipo de tecnología requiere de recursos humanos y especialistas en dominios específicos que cada vez es más escasa. La utilidad para el país, empresas nacionales y la sociedad aumentan con el envejecimiento poblacional y la popularidad de esta tecnología.

La investigación “Herramienta digital para la construcción de conocimiento automático para un Asistente Virtual” (Duvalón, 2022) es una propuesta sobre la creación de un sistema que permite crear asistentes virtuales utilizando el marco de desarrollo de software RASA, y la integración de herramientas de procesamiento del lenguaje natural para mejorar la capacidad de los asistentes para comprender y responder a las preguntas y solicitudes del usuario. La propuesta de (Duvalón, 2022) revela limitaciones como:

- La falta de una arquitectura para acceso remoto (ed., clientes accediendo al servicio de chatbot a través de internet u otro protocolo de comunicación), siendo necesario el despliegue local para su uso (ed., un sistema informático instalado en la computadora personal).
- Carece de una interfaz gráfica amigable e intuitiva para el usuario.
- Además, los asistentes virtuales creados no permiten el envío de mensajes a través de la voz, no logrando una interacción hombre-máquina más fluida y excluyendo a usuarios con dificultades de escritura o discapacidades visuales, así como tampoco

logra el reconocimiento facial de emociones que permita adaptar las respuestas al posible estado de ánimo que tiene el usuario¹.

- El algoritmo de generación de preguntas y respuestas presente en este sistema ha quedado obsoleto debido a nuevos enfoques en el campo del procesamiento del lenguaje natural a partir de la aparición de los grandes modelos del lenguaje como GPT-3.5, entre otros.

Teniendo en cuenta estas carencias y propuestas de trabajo futuro de la investigación analizada, priorizando la interacción de los usuarios con un asistente virtual, se selecciona uno de estos como problema principal de esta investigación.

Problema de investigación:

La investigación (Duvalón, 2022) no permite que los asistentes virtuales (chatbots) creados puedan enviar y recibir mensajes a través de la voz y el reconocimiento de emociones faciales, lo que no permite una interacción más efectiva entre el usuario y el asistente virtual.

Para la formulación del objeto de la investigación, el objeto de estudio y el campo de acción, se propuso basándose en el material de (Pardo Gómez, 2021).

Objeto de la investigación:

La interacción hombre-máquina a través de asistentes virtuales empleando tecnologías de procesamiento del lenguaje natural.

Objeto de Estudio:

Las plataformas de creación automática de asistentes virtuales.

Campo de Acción:

Los asistentes virtuales con soporte para el envío y recepción de mensajes de audio y la compresión del estado emocional del usuario.

A pesar de que la tesis y sistema informático que tiene como base esta investigación (Duvalón, 2022) posee varias carencias, por la importancia para el usuario que interactúa con un asistente virtual se decidió seleccionar como

¹ e.d., no permite reconocer si un usuario está enojado por una respuesta y darle información según su estado o si está satisfecho y continuar el flujo de información

Objetivo General:

Diseñar e implementar una plataforma web para la creación automática y edición de asistentes virtuales con soporte para el envío y recepción de mensajes de voz y el reconocimiento de las emociones faciales enojo, alegría e ira que mejore las deficiencias de la propuesta de (Duvalón, 2022).

Objetivos Específicos²:

- Analizar el estado del arte sobre plataformas para la creación de asistentes virtuales.
- Analizar el estado del arte sobre recientes tecnologías para el procesamiento del lenguaje natural y el uso de inteligencia artificial generativa (ed., modelos grandes de lenguaje).
- Estudiar tecnologías para el análisis de audio y su conversión de audio a texto asociados al modelo computacional Whisper.
- Estudiar tecnologías para la captura y análisis del reconocimiento facial de usuarios que permitan detectar las emociones.
- Diseñar e implementar un nuevo algoritmo de generación de preguntas y respuestas que integre tecnologías y aportes científicos recientes como el uso de un modelo grande del lenguaje (ed., inteligencia artificial generativa).
- Implementar un asistente virtual con soporte para el envío y recepción de mensajes de voz y reconocimiento de las emociones faciales (enojo, alegría, ira).
- Diseñar una aplicación web como plataforma para la creación y gestión de asistentes virtuales con interfaz gráfica amigable e intuitiva.
 - Esta plataforma deberá incorporar un nuevo algoritmo para la generación de la base de conocimiento de un asistente virtual usando inteligencia artificial generativa.
 - Los asistentes virtuales creados deben tener soporte para el envío y recepción de mensajes de voz y reconocimiento de las emociones faciales enojo, alegría w ira de los usuarios.
- Implementar la plataforma.

² La presente investigación es una continuación de (Duvalón, 2022), de la cual se tomaron varias experiencias y código fuente. Los resultados de este trabajo fueron presentados en el Foro Nacional de Estudiantes de Ciencias Técnicas de 2023 alcanzando la categoría de relevante. Por orientación del tutor de esta investigación, se decidió incluir como objetivos específicos varias de las carencias de la investigación (Duvalón, 2022), para lo cual se ha llevado a cabo un proceso de aprendizaje de tecnologías, diseño e implementación y pruebas rigurosas.

- Desplegar y probar la plataforma.

Hipótesis:

La implementación de una plataforma web que facilite la generación automática de asistentes virtuales, capaces de soportar el envío y recepción de mensajes de voz y reconocimiento de las emociones enojo, alegría e ira, mejora las limitaciones identificadas en la propuesta de (Duvalón, 2022). Esta solución con el uso de tecnologías como el Procesamiento de Lenguaje Natural (NLP; siglas en inglés) y la inteligencia artificial generativa ofrece una experiencia más personalizada y eficiente para los usuarios de un asistente virtual.

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.
- Método de modelado: se empleó para realizar los diagramas necesarios para documentar el software.
- Método de entrevistas: Se realizó una entrevista al autor de la investigación previa para facilitar la comprensión del software desarrollado y su integración con la solución actual de esta investigación. Además, se llevaron a cabo entrevistas a usuarios potenciales para evaluar las opciones principales en la creación y gestión de un asistente virtual, incluyendo el soporte para mensajes de voz y el reconocimiento de emociones.

Estructura del informe:

En el primer capítulo se presenta el marco teórico, donde se examinan y fundamentan diversos aspectos teóricos relevantes. El segundo capítulo expone la propuesta del sistema, proporcionando detalles sobre la metodología empleada. El tercer capítulo se centra en la implementación del sistema, las pruebas llevadas a cabo y se discuten los resultados obtenidos. Finalmente, se incluyen las conclusiones y recomendaciones, seguidas de las referencias bibliográficas y los 8 anexos.

Aportes de la investigación:

- Desarrollo de asistentes virtuales con los cuales se pueda enviar y recibir mensajes de audio.
- El reconocimiento de las emociones faciales enojo, alegría e ira de los usuarios o clientes del asistente virtual.
- Desarrollo de una aplicación web con una interfaz de usuario amigable para la creación automática y edición de asistentes virtuales.
- Incorpora de un algoritmo que emplea inteligencia artificial generativa (ed., grandes modelos de lenguaje) en la creación de la base de conocimiento de los asistentes virtuales creados por la plataforma.

CAPITULO 1 . MARCO TEÓRICO

En esta sección se examinarán algunas de las principales herramientas de construcción de asistentes virtuales, conceptos importantes del contexto en el que se está trabajando, teniendo en cuenta que esta investigación toma como base los resultados y carencias de la investigación (Duvalón, 2022) varios de los cuales es objetivo mejorar. Del análisis crítico de las herramientas, tecnologías y conceptos se explicará cuál es la mejor opción para utilizar en este proyecto y por qué.

Procesamiento del Lenguaje Natural.

El Procesamiento de Lenguaje Natural (PLN), es un área de investigación que fusiona elementos de la inteligencia artificial y la lingüística para permitir que las computadoras comprendan el lenguaje y sea capaces de interactuar con los humanos. El propósito fundamental del PLN es cerrar la brecha entre la comunicación humana y la interacción con computadoras, proporcionando la capacidad de interactuar con estas últimas mediante el uso de lenguaje natural (Khurana et al., 2023).

El PLN se desglosa en dos componentes esenciales: la comprensión del lenguaje natural y la generación del lenguaje natural. La comprensión del lenguaje natural implica la habilidad de la computadora para comprender el lenguaje humano, abarcando aspectos como la fonología, morfología, sintaxis, semántica y pragmática. Cabe destacar que figuras prominentes como el lingüista Noam Chomsky han realizado contribuciones significativas al estudio de la sintaxis en la lingüística. Por otro lado, la generación del lenguaje natural se enfoca en la capacidad de la computadora para producir texto coherente y significativo a partir de representaciones internas. Este componente implica la creación de frases, oraciones y párrafos en lenguaje humano. (Khurana et al., 2023).

Los asistentes virtuales emplean técnicas de PLN, como el reconocimiento automático del habla y la interpretación semántica, para transcribir y comprender el significado de las palabras y las intenciones del hablante. Estos procesos mejoran la experiencia del usuario al proporcionar respuestas rápidas y precisas.

Asistente Personal Inteligente.

Un Asistente Personal Inteligente o asistente virtual 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 centro de 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.

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. La tecnología ASR es fundamental en los asistentes virtuales modernos (ej., Siri de Apple y Alexa) permitiendo que estos sistemas comprendan y respondan a comandos de voz de manera eficiente (Sobrinho, 2020).

Las herramientas y tecnologías que utilizan ASR permiten a los asistentes virtuales la conversión de la voz, de un usuario que interactúa con él, a texto y luego poder responder a estos mensajes, según la base de conocimiento que posea.

Estado del arte de las herramientas utilizadas para el desarrollo de la plataforma.

Por la gran cantidad de tecnologías, conceptos (p.ej.; Grandes Modelos de Lenguaje) empleados para el diseño e implementación de la plataforma, se considera importante la necesidad de dejar evidencias de la investigación, análisis comparativo con otras propuestas y las conclusiones que fueron tomadas para la selección de las tecnologías, lenguajes de programación, modelos computacionales para la inteligencia artificial generativa, etc.

Herramientas y plataformas para la creación de asistentes virtuales

En este epígrafe se analizarán las opciones existentes para la creación de asistentes virtuales con el objetivo de elegir una para la creación de los asistentes de este proyecto.

DialogFlow



Figura 1. Logotipo de DialogFlow

DialogFlow, propiedad de Google, es una plataforma popular para la creación de asistentes virtuales. Se destaca por su facilidad de uso, gracias a una interfaz intuitiva y plantillas predefinidas que permiten crear y entrenar chatbots sin necesidad de conocimientos avanzados de programación. Sin embargo, su dependencia del aprendizaje automático puede generar dificultades para comprender intenciones del usuario cuando se usa lenguaje coloquial o ambiguo, resultando en respuestas inexactas. Además, aunque ofrece una versión gratuita, esta tiene limitaciones y para acceder a funciones más avanzadas es necesario contratar una versión de pago, lo que puede ser costoso para algunas empresas. (ChatBot, P, 2022).

Amazon Lex

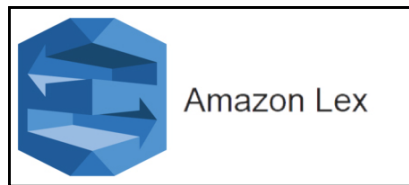


Figura 2. Logotipo de Amazon Lex

Amazon Lex es una plataforma de desarrollo de chatbots que utiliza tecnología de procesamiento de lenguaje natural y aprendizaje automático. Uno de los principales beneficios de Amazon Lex es su integración con otros servicios de Amazon Web Services (AWS), lo que permite a los usuarios crear bots sofisticados y escalables. Además, la plataforma cuenta con una interfaz fácil de usar y ofrece una amplia variedad de plantillas y ejemplos predefinidos, lo que facilita el proceso de creación de un asistente virtual. Sin embargo, uno de los desafíos de Amazon Lex es su precio, que puede ser bastante alto en comparación con otras plataformas, especialmente para empresas pequeñas o medianas. Además, aunque la plataforma utiliza tecnología avanzada de procesamiento de lenguaje natural, a veces puede tener dificultades para comprender las intenciones del usuario, lo que puede llevar a respuestas inexactas o poco útiles por parte del chatbot (Amazon Lex V1, s. f.)

IBM Watson

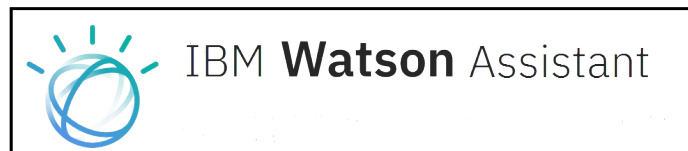


Figura 3. Logotipo de IBM Watson

IBM Watson es una plataforma de inteligencia artificial que ofrece una amplia variedad de servicios, incluyendo procesamiento de lenguaje natural, reconocimiento de voz y visión por computadora. Una de las principales ventajas de Watson es su capacidad para analizar grandes volúmenes de datos no estructurados y proporcionar información valiosa a partir de ellos. No obstante, uno de los desafíos de la plataforma es su complejidad y la necesidad de tener conocimientos técnicos avanzados para aprovechar al máximo sus capacidades. Además, su precio puede ser un obstáculo para algunas empresas, ya que algunos de sus servicios pueden ser costosos en comparación con otras opciones del mercado. (IBM Watsonx Assistant Virtual Agent, s. f.).

RASA

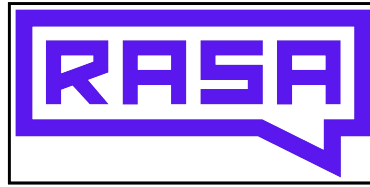


Figura 4. Logotipo de RASA

RASA es un marco de trabajo de código libre para el desarrollo de chatbots y asistentes virtuales. Una de sus principales ventajas es su flexibilidad y personalización, ya que permite a los desarrolladores crear modelos personalizados de procesamiento de lenguaje natural y diálogos adaptados a las necesidades específicas de cada proyecto. Sin embargo, esto también puede ser una desventaja, ya que requiere de conocimientos avanzados de programación y una curva de aprendizaje más pronunciada en comparación con otras plataformas. Además, la falta de una interfaz gráfica de usuario puede dificultar la creación y el entrenamiento de chatbots para usuarios sin experiencia en programación. En cuanto a su rendimiento, RASA puede ofrecer resultados precisos y efectivos si se configura y entrena adecuadamente, pero también puede requerir un mayor tiempo y esfuerzo en comparación con otras plataformas (AskLua, 2020) .

BotPress



Figura 5. Logotipo de Botpress

Botpress es una herramienta que facilita la creación de chatbots, al reunir el código y la infraestructura necesarios en una plataforma completa y fácil de usar (Primeros pasos con Botpress | Botpress Blog, s. f.). Esta plataforma incluye diversas funciones para simplificar el proceso de creación, despliegue y gestión de chatbots, como tareas integradas de procesamiento del lenguaje natural, un entorno de trabajo llamado “*visual conversation studio*”, un emulador y depurador, compatibilidad con canales de mensajería populares, un SDK (Kit de desarrollo de software, en español) y editor de código, y herramientas posteriores a la implementación. Pese a sus ventajas como son la edición en tiempo real del conocimiento del asistente mediante la creación de flujos, una gran desventaja es que no permite que se despliegan los asistentes fuera de dicha plataforma.

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

Herramienta de reconocimiento y transcripción de voz a texto.

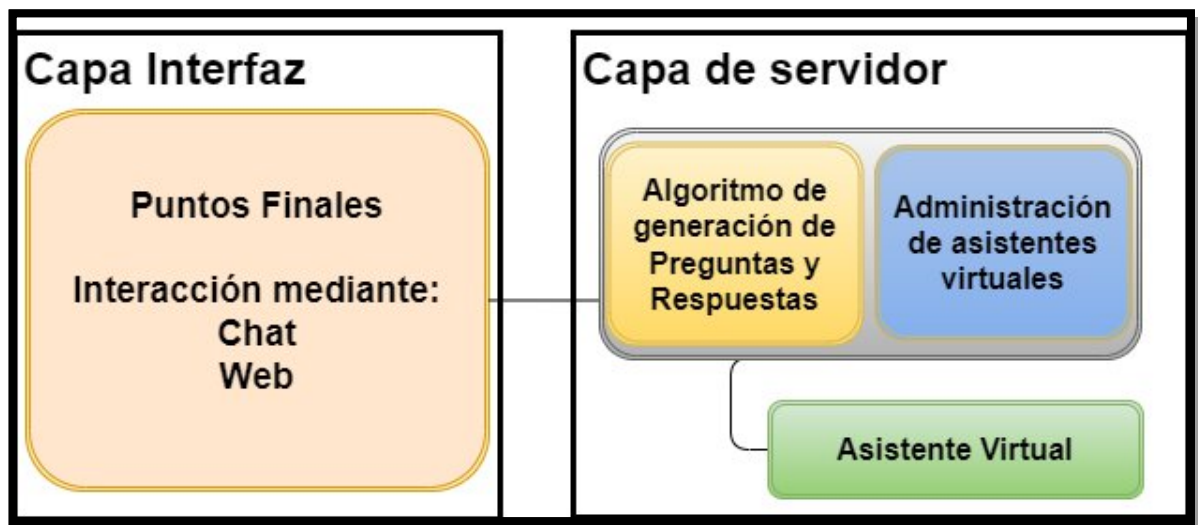
Esta investigación tiene como objetivo lograr el envío y recepción de mensajes de voz entre un usuario y un asistente virtual. Para esto se debe utilizar una herramienta capaz de convertir la voz de un usuario a texto y este sea procesado por un asistente virtual y las respuestas enviadas al usuario convertidas a voz.

Whisper es un modelo computacional de inteligencia artificial que convierte el habla en texto, revolucionando el campo de la transcripción con su capacidad para manejar múltiples idiomas y precisión en el reconocimiento de voz. Entrenado con más de 680.000 horas de datos, este sistema de reconocimiento automático de voz (ASR) es accesible en línea o localmente para todos los usuarios, y puede trabajar con más de 100 idiomas. Ofrece diferentes modelos de tamaños variados para adaptarse a servidores o computadoras con menos recursos (He usado Whisper para transcribir una entrevista, s. f.).

Whisper se destaca entre otras herramientas de transcripción por su habilidad para manejar acentos, ruido de fondo y lenguaje técnico, y por su efectividad en la traducción de voz a texto, superando a modelos que se especializan por su rendimiento (O'Sullivan et al. 2023). Debido a su fácil integración con Python, alta precisión en transcripciones multilingües, código abierto y facilidad de implementación se elige Whisper para transcribir audio a texto en esta investigación.

Herramientas y tecnologías empleadas en la plataforma propuesta en la investigación.

Partiendo de los aportes y carencias de la investigación (Duvalón, 2022) este trabajo propone una plataforma web para la creación y edición de asistentes virtuales. Esta plataforma está conformada por un módulo orientado al cliente (ed., interfaz gráfica y opciones del lado del cliente web) y otro módulo de servicios: para procesar la lógica del negocio, la persistencia, la creación, edición y prueba de asistentes virtuales y el uso de inteligencia artificial generativa.



Se analiza a continuación las herramientas y tecnologías empleadas en cada módulo de la plataforma teniendo en cuenta el análisis crítico de las posibles herramientas que existen en el estado del arte.

Marco de Trabajo utilizado en el módulo orientado al cliente (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.)

Para este proyecto se elige el desarrollo de la aplicación web con **React 18**, 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 (Modelo de Objetos del Documento, en español), puede renderizar componentes extremadamente rápido, superando a Angular y a Vue en este aspecto, lo cual es importante debido a los requisitos de este proyecto. Por último, gracias a su alta popularidad, la mayoría de las bibliotecas existentes están disponibles para su uso en React, como las bibliotecas de reconocimiento facial y de habla utilizadas en la investigación.

Herramienta para el reconocimiento de emociones faciales.

La biblioteca face-api.js es una herramienta de JavaScript para la detección y reconocimiento facial, basada en la API (Interfaz para el desarrollo de aplicaciones; en español) principal de tensorflow.js. Ofrece diversas funcionalidades como detección de rostros, puntos de referencia faciales, reconocimiento facial, expresiones, estimación de edad y género. Su facilidad de uso se destaca por ofrecer APIs de alto y bajo nivel, adaptándose a las necesidades de los desarrolladores. Es compatible con el navegador y Node.js, lo que permite su uso en aplicaciones web y de servidor (Infoteknico, 2023). Sin embargo, presenta limitaciones como variabilidad en la precisión según las condiciones de iluminación y calidad de imagen, y requiere recursos computacionales significativos. Aunque existen alternativas como OpenCV, Dlib y Face++, face-api.js se distingue por su simplicidad y flexibilidad, aprovechando la red neuronal de tensorflow.js (Infoteknico, 2023). Para este proyecto se utilizó face-api.js para hacer el reconocimiento de las emociones faciales enojo, ira y alegría con lo cual se podrán preparar respuestas adaptadas para cada

una de estas emociones, lo cual permite brindar una experiencia más amena y personalizada para cada usuario.

Marco de trabajo utilizado en el módulo de servicios.

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).

Lenguajes de programación utilizados.

Python

Para el desarrollo de este módulo de la plataforma el lenguaje de programación base fue Python, un lenguaje de programación de alto nivel, interpretado y de propósito general, es reconocido por su simplicidad y legibilidad, lo que lo convierte en una opción ideal para los programadores novatos. Su creación, a cargo de Guido van Rossum, se remonta a 1991 (¿Qué es Python?, s. f.). Este lenguaje versátil se aplica en diversos campos, incluyendo el desarrollo web, donde se utiliza para crear el módulo de la lógica de negocios de sitios web y aplicaciones, con marcos de trabajo como Django y Flask (Londoño, s. f.); el desarrollo de software, donde su compatibilidad con múltiples plataformas y sistemas operativos lo hace una elección popular (Londoño, s. f.); y la ciencia de datos y el aprendizaje automático, donde su sintaxis clara y bibliotecas especializadas como NumPy, Pandas, Matplotlib, Scikit-learn y TensorFlow lo han convertido en uno de los lenguajes más utilizados (Londoño, s. f.).

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

Estudio del estado del arte de marcos de trabajo en Python para aplicaciones web.

En este epígrafe se realizara un estudio de las principales tecnologías que permitan el desarrollo de aplicaciones web con Python.

Marco de trabajo Django

Django es un marco de trabajo de desarrollo web de alto nivel, de código abierto, escrito en Python, conocido por su versatilidad, productividad y seguridad, utilizado por empresas como Instagram, Disqus y la NASA. Su componente Django REST, popular para APIs en Python, ofrece una arquitectura modular y personalizable, adecuada para APIs simples y complejas, y es especialmente útil para proyectos de Maquinas de Aprendizaje gracias a su escalabilidad y capacidad para manejar grandes volúmenes de tráfico. Sin embargo, Django puede ser difícil de dominar debido a su amplia gama de configuraciones y no es recomendable para proyectos pequeños debido a su intensivo uso de recursos del servidor, además de ser un marco monolítico que no ofrece mucha flexibilidad en la arquitectura del proyecto (Team, 2022).

Flask

Es un pequeño marco de trabajo creado en Python, desarrollado por el Grupo Internacional de Pythonistas, que utiliza WSGI y Jinja2, y es respaldado por empresas como Samsung, Netflix, Uber y Airbnb. Entre sus ventajas se incluyen su facilidad de uso, simplicidad, motor de plantillas, soporte para pruebas unitarias y despacho de solicitudes REST (Transferencia de Estado Representacional por sus siglas en español) . Sin embargo, también tiene desventajas como la posibilidad de un desarrollo de código de baja calidad, el manejo secuencial de solicitudes y el riesgo de seguridad al incluir módulos adicionales. Estos problemas se pueden mitigar con un hosting especializado en Python (Condez, 2022).

FastAPI

Es un marco de trabajo web de alto rendimiento para Python, conocido por su simplicidad y velocidad, comparable con NodeJS y Go. Es ideal para construir APIs rápidas y eficientes. Entre sus ventajas, destaca su documentación automática interactiva, el uso de declaraciones de tipos de Python 3.6 para autocompletado y soporte de editor, así como la incorporación de seguridad y autenticación (Geekflare, s.f.).

Sin embargo, como marco de trabajo relativamente nuevo, FastAPI puede no ser utilizado frecuentemente y aún está en desarrollo. Aunque su comunidad está creciendo, no es comparable a las de otros marcos de trabajo. Pese a su robustez, hay aspectos que se pueden mejorar (Geekflare, s.f.).

Análisis de los marcos de trabajos.

En la [figura 1](#), [figura 2](#) se muestra un marco de prueba realizado en el año 2022 la cual consiste en realizar 20 consultas a un servidor por segundo (TechEmpower Web

Framework Performance Comparison, 2023). En esta FastAPI demuestra su alta velocidad en el manejo de concurrencia, ligereza y escasa latencia al realizar consultas, como se muestra en el primer y tercer lugar (Ver [figura 2](#)).

Luego de realizar un análisis crítico sobre las ventajas y desventajas de los marcos de trabajos y de revisar los resultados de la prueba de punto de referencia que se plasmó anteriormente, 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.

Herramientas para la creación del algoritmo de generación de preguntas y respuestas

Uno de los objetivos específicos de esta investigación es crear un nuevo algoritmo de generación de preguntas y respuestas para la base de conocimientos de un asistente virtual. Esto se debe a que la solución propuesta por la investigación de (Duvalón 2022) ha quedado obsoleta debido a la aparición de nuevos aportes en el PLN como los grandes modelos de lenguaje y sus amplias capacidades generativas de información.

En este epígrafe se realiza un análisis del estado del arte de las herramientas que pueden ser útiles en el proceso de generación de preguntas y respuestas como núcleo de la base de conocimientos de un asistente virtual, así como los aspectos que se tuvieron en cuenta para la selección de una de ellas.

Langchain³

Es una herramienta de código abierto que simplifica la creación de aplicaciones usando modelos de lenguaje. Mejora la personalización y relevancia de la información generada, permitiendo a los desarrolladores crear o adaptar cadenas de solicitudes y proporcionando componentes que facilitan el acceso de los modelos a nuevos datos sin necesidad de volver a entrenarlos. Destaca por su capacidad de ajustar los modelos de lenguaje a contextos empresariales específicos, optimizando así el desarrollo de aplicaciones que respondan a datos organizacionales (Topsakal & Akinci, 2023).

³ [Introduction](#) |  [LangChain](#)

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

Base de datos Vectorial.

Una base de datos vectorial es un sistema especializado para almacenar y gestionar datos en forma de vectores multidimensionales, que representan características de los datos. Estos vectores pueden resultar de transformaciones de datos complejos como texto, imágenes, audio o video, usando técnicas como modelos de aprendizaje automático o extracción de características. Su principal ventaja es la capacidad de localizar y recuperar datos rápida y precisamente, basándose en la similitud de los vectores, permitiendo búsquedas semánticas o contextuales (Han et al., 2023).

Se empleará una base de datos vectorial que permita almacenar los documentos para que luego de que se generen las preguntas con el contenido de documento se le pasen estas preguntas como consultas a la base de datos vectorial para que a partir de la búsqueda basada en similitud se encuentre la respuesta para esa pregunta.

Qdrant es una base de datos vectorial versátil que ofrece una API fácil de usar y se adapta bien a coincidencias basadas en semántica o redes neuronales, soportando una amplia gama de criterios de consulta y tipos de datos. Sin embargo, su desventaja es la complejidad en la configuración y optimización del sistema. (Singh et al., 2023).

Faiss es una biblioteca de Facebook AI Research diseñada para la búsqueda de vectores de alta dimensión. Es eficiente en la búsqueda de vecinos más cercanos y puede manejar grandes volúmenes de datos. Su desventaja radica en que es más una biblioteca que una base de datos completa, lo que puede requerir más trabajo de integración y gestión de datos. (Douze et al., 2024).

ChromaDB es una base de datos vectorial de código abierto que facilita el almacenamiento y la recuperación de incrustaciones (embeddings; termino en inglés) vectoriales con metadatos asociados, lo que simplifica la creación de aplicaciones de PLN. Entre sus capacidades destacan el filtrado, la agrupación inteligente y la relevancia de consulta, siendo especialmente útil para aplicaciones de generación de preguntas y respuestas basadas en contexto. Sin embargo, al ser una tecnología de código abierto, puede presentar desventajas como la posible falta de soporte técnico dedicado. (Dhungana, 2023).

ChromaDB es la mejor opción para este proyecto debido a su habilidad para manejar datos complejos como texto y automatizar tareas de gestión de inserciones. Su escalabilidad y rendimiento lo hacen ideal para almacenar y recuperar información relevante de documentos, lo que facilita la generación de preguntas y respuestas basadas en contextos específicos.

Modelo de Lenguaje para la generación de preguntas y respuestas.

Un asistente virtual se auxilia de su base de conocimiento para responder a las preguntas que le hace un usuario. Al recibir una pregunta en formato texto u otro tipo se busca la respuesta más cercana y se responde al usuario.

En el caso de un dominio específico, como pudiera ser las consultas y especialidades médicas relacionadas al desarrollo neuro-infantil de un niño, se debe nutrir de información a esta base de conocimiento o usar un asistente virtual de propósito general como Chat-GPT.

Esta investigación propone como uno de sus resultados la generación de asistentes virtuales de dominio específico a partir de la información que le proporcione el usuario o cliente que desea la creación de este tipo de asistente. Para lograr esto es necesario un mecanismo inteligente para lograr la generación de la base de conocimiento del asistente en formato de posibles preguntas y respuestas del contenido o dominio deseado.

Para lograr la generación automática de la base de conocimiento de dominio específico de un asistente virtual, se propone en este trabajo, el uso de la inteligencia artificial generativa y en especial los grandes modelos de lenguaje (*Large Language Model*; LLM).

Un gran modelo de lenguaje es un sistema de inteligencia artificial avanzado, capaz de generar y comprender texto de manera similar a un humano, facilitando interacciones y tareas diversas. Para generar preguntas de calidad con una estructura semántica similar a las de los humanos, se seleccionó un gran modelo de lenguaje en lugar de otras opciones debido a su capacidad para comprender y generar texto de manera efectiva y natural.

En el estado del arte existen diversos LLM con diferentes propósitos, métodos generativos y formas de uso (ed., privado, público). Para seleccionar el LLM se realizó un estudio de las principales características de los más útiles para la plataforma:

Llama 2: Es un modelo de lenguaje de código abierto desarrollado por la empresa Meta⁴. 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 marcos de referencias (benchmark), lo que lo convierte en una opción viable (Luzniak, 2023).

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

GPT-3.5: 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: Es otro modelo de lenguaje desarrollado para el Instituto de Innovación Tecnológica (IIT) de los Emiratos Árabes Unidos (EAU). En comparación con Llama 2, Falcon ha mostrado un rendimiento inferior en la mayoría de los marcos de prueba, aunque su rendimiento específico puede variar dependiendo del caso de uso (Luzniak, 2023).

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 marco de pruebas 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) .

⁴ [Meta Research \(facebook.com\)](https://www.facebook.com/metaresearch)

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 (Ars Technica, s. f.).

Para la selección de un LLM no solo se tuvo en cuenta que sea de código abierto, además se analizó su rendimiento otras métricas, con respecto a otros modelos como GPT-3.5, PALM y FALCON.

Al comparar Llama 2 y Falcon, como se muestra en la [tabla 1](#), los modelos de 7 mil millones de parámetros como Llama 2 tiene mejores resultados en la métrica MMLU (casi el doble de puntuación en MMLU). Esta métrica mide cuán bien un modelo de lenguaje a gran escala comprende el lenguaje y resuelve problemas usando el conocimiento adquirido durante su entrenamiento.

En la [tabla 2](#), 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.

En esta investigación se elige a Llama 2 por los buenos resultados de desempeño que muestra, ser de libre acceso y permitir ajustes para mejorar la precisión en la generación de preguntas y respuestas (para la creación de la base de conocimiento de un asistente virtual), y se puede ejecutar localmente sin necesidad de una GPU. Estos aspectos son cruciales ya que otros modelos no permiten ajustes debido a su licencia o son costosos en términos de recursos de hardware.

Base de Datos utilizada.

En esta investigación para almacenar la información persistente de la plataforma se seleccionó MongoDB, una base de datos NoSQL, teniendo en cuenta los siguientes criterios para su selección:

- La capacidad de manejar grandes volúmenes de datos y tráfico gracias a su capacidad de distribución y sharding automático (Postgres vs. MongoDB, 2023).
- El modelo basado en documentos permite almacenar datos de diferentes estructuras en la misma colección, lo cual es beneficioso para un proyecto de asistentes virtuales con diversas necesidades de datos (MongoDB vs PostgreSQL: 15 Critical Differences, s. f. -a).

- En términos de rendimiento, MongoDB utiliza BSON (JSON binario por sus siglas en español) para almacenar datos, lo que agiliza la lectura y escritura, y la creación de índices en campos y subdocumentos mejora la velocidad de las consultas (MongoDB Vs PostgreSQL, s. f.).
- Ofrece transacciones multi-documento, útiles para operaciones que involucran múltiples asistentes virtuales o usuarios (MongoDB Vs PostgreSQL, s. f.).
- Es más económico que PostgreSQL, especialmente al escalar el proyecto, y su opción de nube, MongoDB Atlas, facilita la gestión de la base de datos (MongoDB vs PostgreSQL: 15 Critical Differences, s. f. -b).
- Su sintaxis de consulta es más intuitiva y amigable en comparación con SQL, lo que simplifica el desarrollo del proyecto.

Entorno de desarrollo y herramientas utilizadas para la implementación de los módulos de la plataforma.

Visual Studio Code (versión 1.86)

También llamado VS Code, es una pequeña versión 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 Python, 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 (intellisense). Pero para obtener detección inteligente, un compilador y depuradores para otros lenguajes, debe descargar las extensiones correspondientes (Visual Studio vs Visual Studio Code, 2023).

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

Sistema de Control de Versiones GIT versión 2.44.0

Es un sistema de control de versiones distribuido que permite a los desarrolladores trabajar juntos en proyectos 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.)

Metodología de Desarrollo de Software XP

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

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

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

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

Conclusiones del capítulo.

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

CAPITULO 2 . ORGANIZACIÓN Y DISEÑO

En este capítulo, se abordará el proceso de planificación y diseño de la plataforma, 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.

Pese a que se utiliza la metodología XP para el desarrollo de este proyecto también se utilizaron artefactos de la metodología del Proceso Unificado de Modelado (RUP; siglas en inglés) como son el proceso de levantamiento de requisitos no funcionales de software y hardware y diagramas de actividades y secuencia los cuales contribuyen a hacer que la documentación del software sea mejor en términos de claridad, detalle y estructura.

Propuesta de Sistema.

Para abordar la problemática presentada, se sugiere el desarrollo de una aplicación web para la creación y edición de asistentes virtuales la cual permitiría que los clientes accedan de forma remota desde cualquier dispositivo que pueda conectarse a internet; esta aplicación tendrá un módulo o componente basado en el marco de trabajo RASA para la creación de asistentes virtuales. A continuación, se detalla la arquitectura propuesta para el sistema.

La aplicación web seguirá una arquitectura cliente-servidor, donde el cliente será la interfaz con la que interactúa el usuario y el servidor alojará la lógica de negocio, así como un módulo para la interacción con los modelos de inteligencia artificial y la creación de asistentes virtuales.

El módulo de Creación del Asistente Virtual estará estructurado en capas, con el fin de separar las diferentes funcionalidades y facilitar la integración de otros modelos de inteligencia artificial en el futuro. Las capas propuestas son las siguientes:

1. Capa de Interacción con el Usuario: Esta capa se encargará de gestionar la interfaz web a través de la cual el usuario introducirá el documento para la creación del asistente virtual. También permitirá la edición de las respuestas del asistente a posibles preguntas de los usuarios finales.
2. Capa de Procesamiento de Datos: Esta capa recibirá la información proporcionada por el usuario y realizará el preprocesamiento necesario para preparar los datos antes de enviarlos al modelo de inteligencia artificial.
3. Capa de Inteligencia Artificial Generativa: Esta capa contendrá el modelo computacional (actualmente, el Gran Modelo de Lenguaje Llama2) responsable de analizar la información proporcionada por el usuario y generar la base de conocimiento del asistente virtual de forma automática. Esta capa está diseñada para permitir la integración de otros modelos de inteligencia artificial en el futuro, si es necesario.
4. Capa de Persistencia: Esta capa se encargará de almacenar y recuperar la información relacionada con los asistentes virtuales creados, así como las modificaciones realizadas durante el proceso de edición.

De esta manera, la arquitectura propuesta busca proporcionar una solución clara y escalable para la creación y edición de asistentes virtuales, facilitando la integración de nuevos modelos de inteligencia artificial en el futuro.

Flujo del Sistema.

En la [figura 6](#) se muestra el diagrama de flujo para el marco de trabajo que permite generar un asistente virtual. Haciendo referencia a la información que se muestra en la figura, el proceso de creación de un asistente virtual comienza cuando el usuario llena un formulario para crear un asistente e introduce un documento con el conocimiento requerido en la interfaz web (Paso 1 y 2). Este documento se procesa utilizando un algoritmo generador de preguntas que está en el sistema de generación de conocimiento (Paso 3) auxiliándose las herramientas del marco de trabajo Langchain (el cual ofrece herramientas para la ingeniería de instrucciones (*prompts*; término en inglés) y estructuras de indicaciones, gestionando flujos de diálogo, contexto y memoria conversacional, lo que lo hace ideal para desarrollar aplicaciones conversacionales avanzadas) (Pillai y Thakur 2024), luego se almacena en la base de datos ChromaDB (Paso 4). Posteriormente, se consultan las preguntas generadas previamente para obtener respuestas (Paso 5), que son almacenadas en una base de datos MongoDB. Con estas preguntas y respuestas se crean los archivos de entrenamiento (base de conocimiento) para el asistente (Paso 6 y 7), el cual se crea (Paso), entrena (Paso 9) y

está listo para que el usuario descargue un comprimido que contiene el asistente virtual y pueda interactuar con él a través de texto, audio o video (Paso 10).

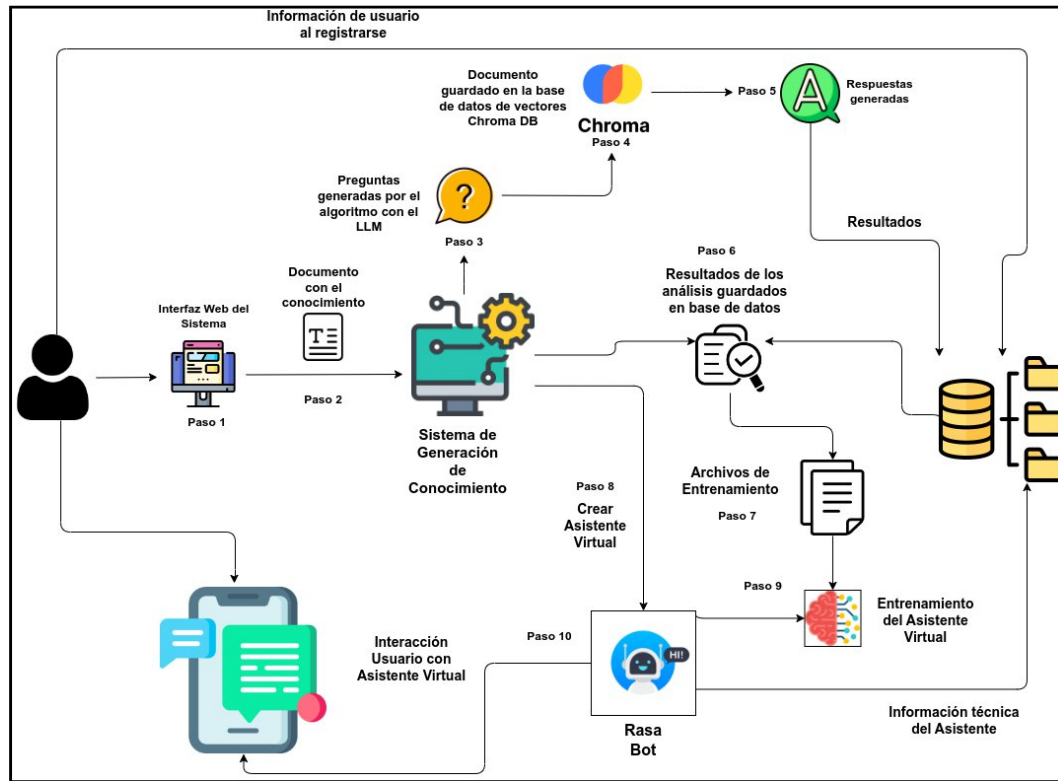


Figura 6 Diagrama de flujo de la generación de asistentes virtuales.

En la [figura 7](#), se muestra el flujo de la interacción por medio de comandos de voz de un asistente creado 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 chatbot de texto a voz.

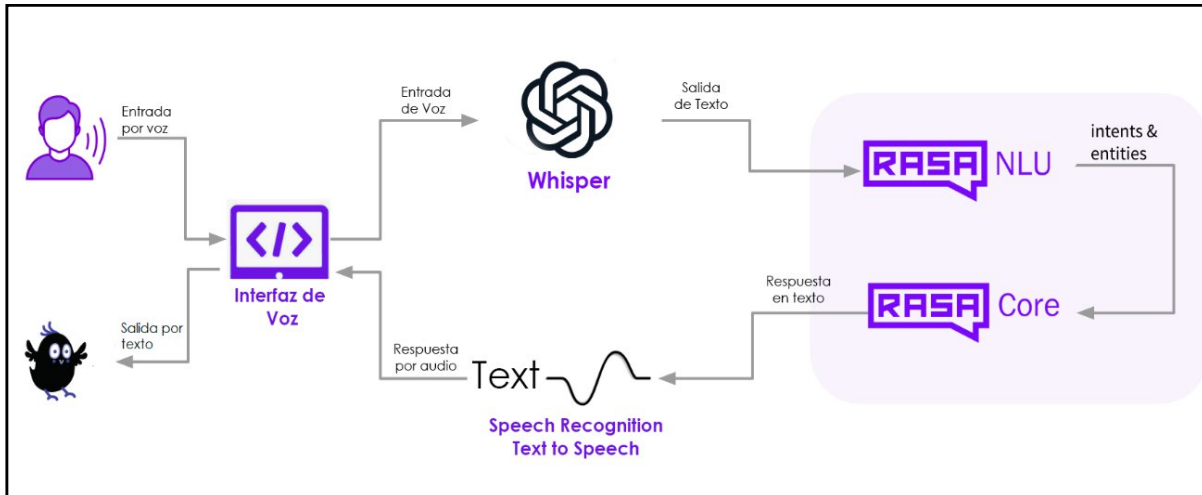


Figura 7 Diagrama de flujo de la interacción por medio de voz con un Asistente creado por el marco de trabajo propuesto.

Historias de Usuario.

Las historias de usuario son tareas de desarrollo expresadas como "persona + necesidad + propósito", enfocadas en solucionar problemas para usuarios reales, permitiendo colaboración, impulsando soluciones creativas y motivando al equipo de desarrollo (Williams, 2003). A continuación, se muestran todas las Historias de Usuarios propuestas y una breve descripción:

1. **Autenticar usuario en el sistema usando su email:** Como usuario, necesito ingresar mi correo electrónico y contraseña para acceder al sistema y utilizar sus funcionalidades de manera segura y personalizada.
2. **Registrar usuario en el sistema usando su email:** Como usuario nuevo, necesito registrarme en el sistema utilizando mi correo electrónico y creando una contraseña para poder acceder a las funcionalidades y servicios que ofrece la plataforma.
3. **Crear asistente virtual con información inicial:** Como usuario, se necesita poder crear un asistente virtual proporcionando un documento con el conocimiento y completando un formulario de creación para que el asistente pueda responder preguntas relevantes y proporcionar información valiosa a los usuarios finales.
4. **Editar asistente virtual:** El usuario, puede editar un asistente virtual existente, modificando el formulario y el archivo de conocimiento asociado para generar nuevas preguntas y respuestas, y así mejorar la precisión y eficacia del asistente.
5. **Visualizar asistentes creados:** Se mostrarán en una tabla donde se verán las opciones de editar, probar y eliminar el asistente: Como usuario, necesito ver una lista de los asistentes virtuales que he creado en una tabla, con opciones para

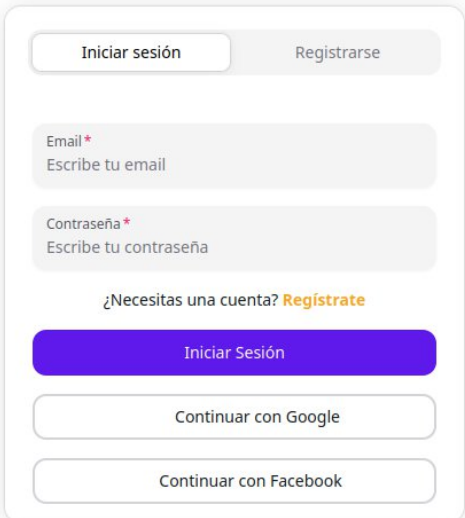
editarlos, probarlos o eliminarlos, para poder gestionar y mantener mis asistentes de manera eficiente.

6. **Eliminar asistente virtual:** En la tabla donde se muestran los asistentes creados, como usuario, se puede eliminar un asistente virtual de, para poder administrar los asistentes y deshacerse de aquellos que ya no son necesarios.

A continuación, se muestran de forma más detallada y con su respectivo diseño algunas de las principales historias de usuario.

Autenticar al Usuario en el Sistema.

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

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

Crear asistente virtual con información inicial.

En esta historia de usuario se rellena el formulario que permite obtener la información necesaria para analizar el documento y generar las preguntas y respuestas para posteriormente pasarlas al formulario de la derecha donde el usuario podrá editarlas en caso de que no esté satisfecho o encuentre un error en alguna. Por último, podrá generar

los archivos de entrenamiento para su asistente o guardar los resultados para posteriormente entrenar el asistente.

Historia de usuario	
Número:2	Nombre de HU: Crear asistente virtual con información inicial
Usuario: Usuario autenticado	
Prioridad en negocio: Alta	Riesgo de desarrollo: Alto

Datos del asistente

Nombre *

Escribe el nombre de tu asistente

Descripción

Escribe una descripción que te permita identificar a tu asistente

Conocimiento del asistente

Escribe conocimiento para tu asistente o carga un archivo

Cargar Archivo

Tipos de archivos permitidos: PDF, DOCX, TXT

Analizar

Preguntas y Respuestas generadas

Preguntas

Podrás editar este contenido cuando se analice el conocimiento.

Respuestas

Podrás editar este contenido cuando se analice el conocimiento.

Generar Archivos

Guardar resultados

Visualizar asistentes creados.

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

Historia de usuario	
Número: 3	Nombre de HU: 5. Visualizar asistentes creados
Usuario: Usuario autenticado	
Prioridad en negocio: Alta	Riesgo de desarrollo: Alto

Q Buscar...

Crear Asistente +

Total 7 assistants

Rows per page:5

	NOMBRE	DESCRIPCIÓN	CONOCIMIENTO	ESTADO CREACIÓN	FECHA CREACIÓN	ACCIONES
<input type="radio"/>	Teresa	Asistente para las tiendas TRD	atencion_cliente_trd.docx	● Creado	11 de junio de 2024	  
<input type="radio"/>	Mateo	Asistente para las bodegas	bodegas_bot.docx	● No creado	11 de junio de 2024	  
<input type="radio"/>	Teresa	Asistente para las tiendas TRD	atencion_cliente_trd.docx	● En proceso	11 de junio de 2024	  
<input type="radio"/>	Teresa	Asistente para las tiendas TRD	atencion_cliente_trd.docx	● Creado	11 de junio de 2024	  
<input type="radio"/>	Teresa	Asistente para las tiendas TRD	atencion_cliente_trd.docx	● Creado	11 de junio de 2024	  

< 1 2 >

0 of 5 selected

Requerimientos para el despliegue de la plataforma

Estos son criterios de calidad que especifican el rendimiento, la seguridad, la usabilidad, etc. del sistema. Incluso en metodologías ágiles donde la flexibilidad es crucial, incluirlos en su documento garantiza una comprensión completa de las expectativas del sistema y mejora la planificación y el diseño (Villanueva & Molina, s. f.).

Software.

Los requisitos no funcionales del software son especificaciones que no se refieren directamente a las funciones del sistema, sino a sus características operativas, como rendimiento, seguridad, usabilidad y fiabilidad, que influyen en la experiencia del usuario y en la calidad del software. Estos conceptos que definen cómo determinar los requisitos no funcionales del sistema guiaron la forma de estimar los requerimientos necesarios para el despliegue de la plataforma. (Dragos, 2021).

Requisito	Versión
Sistema Operativo	El software es compatible con Windows 10 o superior y la distribución Ubuntu 22.04 o superior ⁵ .
Node.js Versión	18

⁵ Se recomienda realizar pruebas en otras distribuciones de Linux y el sistema operativo MacOS para comprobar la funcionalidad del proyecto.

Python Versión	3.11.8
MongoDB Versión	5.0.x

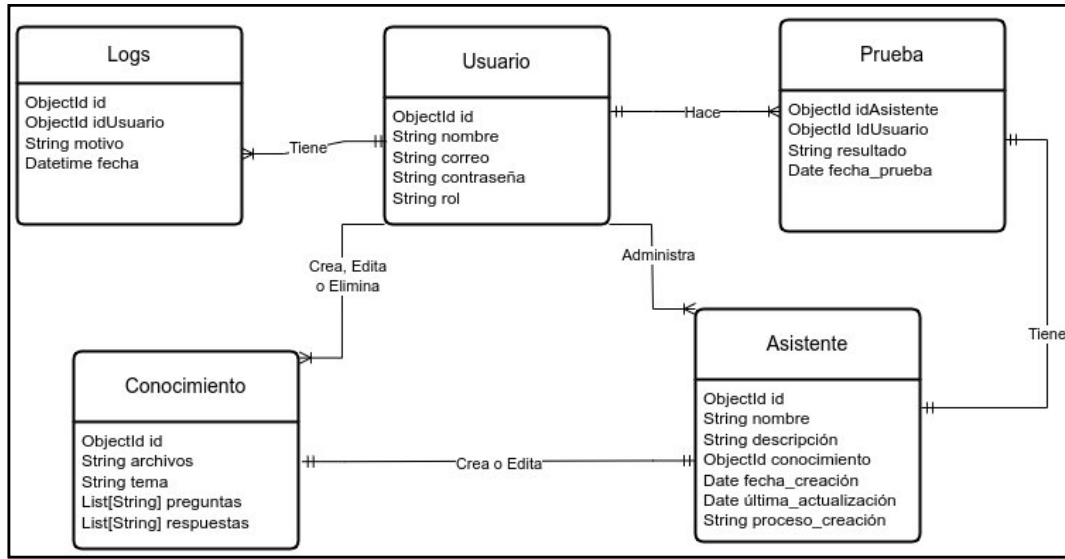
Hardware.

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

Componente	Requisitos mínimos
Procesador	Procesador Intel Core i3 de 7ma generación con al menos 4 núcleos y 6 hilos o superior.
RAM y GPU	Para optimizar el rendimiento y el tiempo de respuesta a tareas que usen Whisper y modelos de lenguaje, se aconseja contar con al menos 8 GB de RAM. Además, se recomienda el uso de GPU, y como requerimiento mínimo una tarjeta gráfica como la Nvidia GTX 960 con 4 GB de memoria DDR5, para una ejecución más eficiente o superior.
Almacenamiento	<p>Se necesita aproximadamente 20 GB de almacenamiento debido al tamaño en memoria de los modelos del lenguaje y de Whisper y al tamaño que usan estos modelos al ser cargados en memoria.</p> <p>Otro elemento para tener en cuenta es el tamaño mínimo para el almacenamiento de varios asistentes virtuales creados por la plataforma y la Base de Datos si reside en el mismo lugar que el módulo de generación y edición de asistentes virtuales.</p>

Diseño de Base de Datos.

El sistema emplea una base de datos NoSQL, específicamente MongoDB, que se centra en documentos. Este tipo de bases de datos están diseñadas para gestionar y manipular grandes volúmenes de datos a lo largo del tiempo. A diferencia de las bases de datos relacionales, las NoSQL no necesitan un esquema predefinido de tablas y columnas. En su lugar, utilizan colecciones, que son similares a las tablas en las bases de datos SQL, y documentos, que reemplazan a las filas de las bases de datos SQL. Los documentos en MongoDB no tienen un esquema fijo, lo que permite añadir campos de manera flexible según sea necesario. El empleo de un tipo de base de datos NoSQL no exige que el diseño de la base de datos este normalizado, aunque se trató de respetar en lo posible los requerimientos y restricciones asociadas a la normalización (Elmasri et al., 2007).



Diseño de la Base de Datos⁶.

Usuarios del Sistema.

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

Tabla de 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 del reconocimiento de emociones del usuario.

En la [figura 8](#) se observa el diagrama de secuencia que ilustra como el usuario luego de que interactúa con el asistente este analiza el rostro del mismo para detectar las emociones presentes en el, posteriormente estas se envían en formato de texto para el asistente virtual el cual analiza las emociones del usuario por medio del reconocimiento facial y le envía una respuesta en consecuencia de si esta triste o enojado.

⁶ Este diseño se lleva a cabo como una buena práctica para documentar la estructura de la base de datos de la plataforma de manera eficiente.

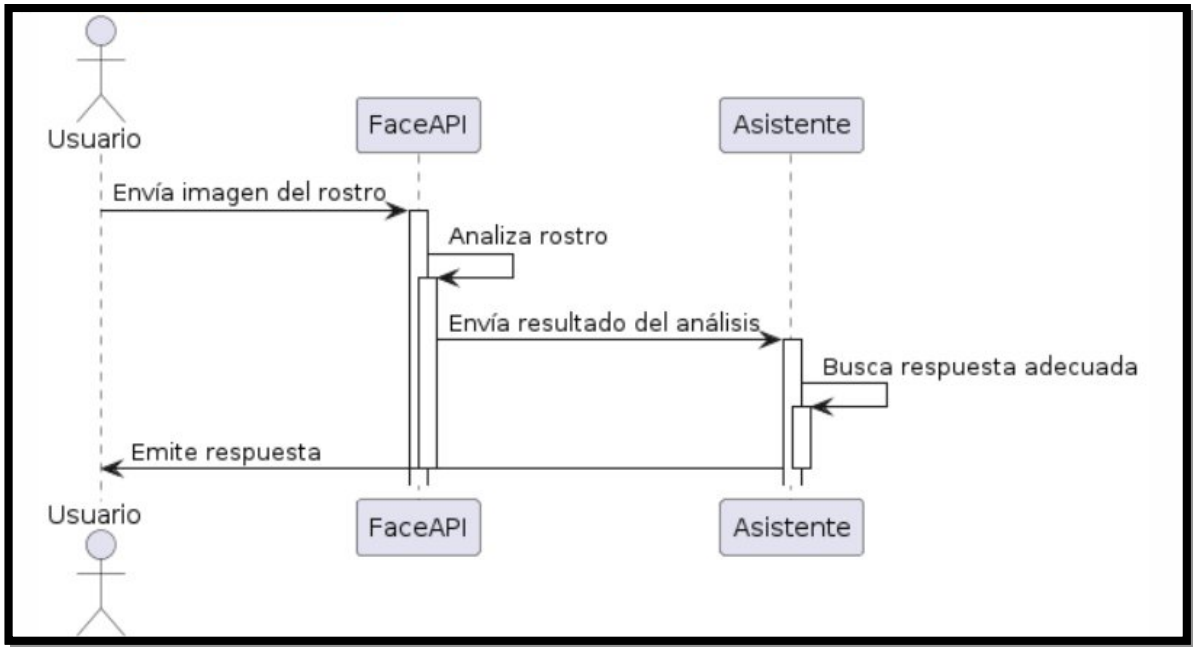


Fig 8. Diagrama de Secuencia del reconocimiento de emociones del usuario.

Diagrama de Actividades del reconocimiento de voz

Este proceso inicia cuando el usuario graba su voz a través de la interfaz web. El audio se envía al servidor para convertirse en texto, que luego se procesa para encontrar una respuesta a la solicitud del usuario. Finalmente, se devuelve la respuesta a la solicitud del usuario. Este diagrama es de vital importancia ya que facilita la comprensión del flujo y sirve para detectar insuficiencias en el mismo. Se puede observar este flujo en la [figura 9](#).

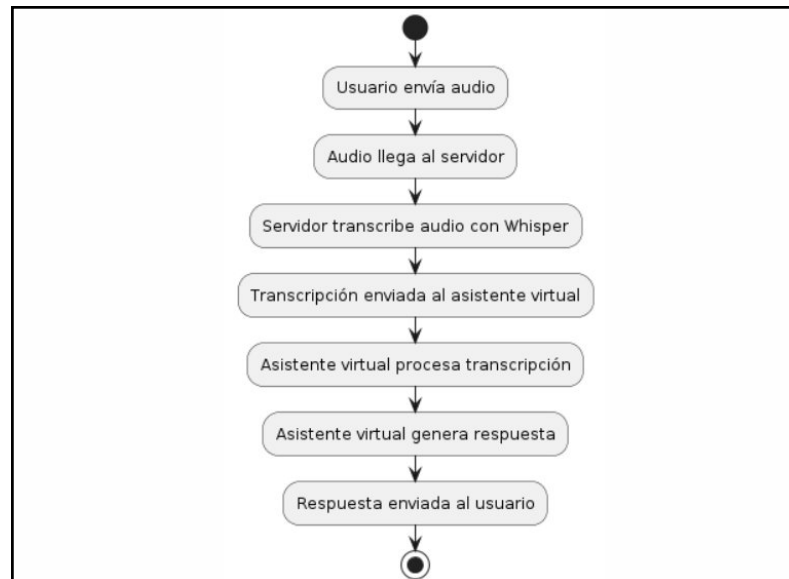


Fig 9. Diagrama de Actividades de Creación de un Asistente.

Diseño de la interfaz del Chat para probar el asistente

Cada asistente creado en la plataforma cuenta con una interfaz de chat que incluye una sección para reconocer las emociones mediante el uso de la webcam de la computadora o la cámara delantera del dispositivo móvil, tablet o laptop; así como reconocimiento de audio mediante el micrófono del dispositivo. Esto permite al usuario enviar solicitudes al asistente mediante mensajes de audio, facilitando una comunicación más fluida. Dado que existe un contraste entre sus colores y fuentes de letra, esta interfaz es accesible para cualquier usuario, independientemente de sus discapacidades. Además, se adapta a teléfonos inteligentes y tablets tal como se muestra en la [figura 10](#). La interfaz gráfica puede ser reemplazada por otra que el usuario estime conveniente, de esta puede tomar la lógica de conexión y aplicarla a otras interfaces.

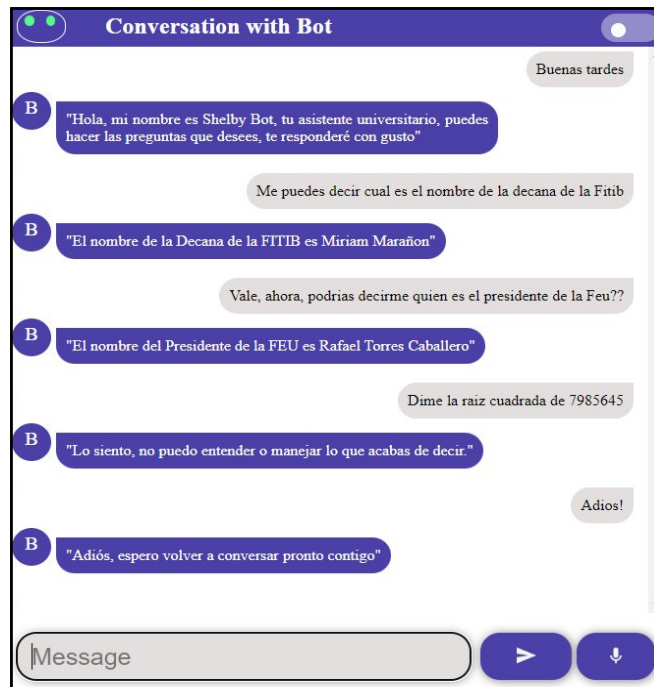


Figura 10. Interfaz de chat de cada asistente creado.

Conclusiones del capítulo

En este capítulo, se muestra el diseño de la plataforma virtual para la creación y edición de asistentes virtuales, utilizando una arquitectura cliente-servidor dividida en capas para facilitar la integración de nuevos algoritmos y mejoras constantes. Se presentaron historias de usuario e ilustraciones de diseño para proporcionar una idea clara de los resultados deseados. También se incluyeron los requisitos técnicos, el diseño de la base de datos y varios diagramas (flujo, secuencia y actividades) para documentar adecuadamente el proyecto. Al final, se presentó la interfaz de chat para la prueba de los asistentes, mostrando dos de sus funcionalidades (reconocimiento de voz y envío de mensajes de audio).

CAPITULO 3 . 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.

Los análisis y ejemplos de implementación mostrados son fundamentales para demostrar la viabilidad y eficacia de la solución propuesta, y para comprobar que cumple con los objetivos y requisitos establecidos en esta tesis.

Patrones de Diseño utilizados

Los patrones de diseño de software son estrategias de programación que resuelven problemas comunes, proporcionando soluciones estándar para garantizar la eficacia en la resolución de problemas específicos (Guerrero & Suárez, 2013).

En este proyecto se utilizaron varios patrones de diseño para mejorar la estructura y eficiencia del código. Estos incluyen el patrón Singleton, el patrón Factory y el patrón Mediator.

El patrón Singleton garantiza que una clase tenga solo una instancia y proporciona un punto de acceso global a esta instancia. Esto es especialmente útil en entornos multithreaded, donde se debe evitar la creación de múltiples instancias. Se usó en la clase Config la cual tiene todas las variables de configuración necesarias por ejemplo las credenciales de la base de datos.

El patrón Factory permite desacoplar la creación de objetos de su uso, facilitando la extensibilidad y la modularidad del código. Las clases Factory suelen tener un método estático Create() que decide qué tipo de objeto crear basándose en los parámetros que recibe. Este patrón se utilizó en la clase Assistant.

El patrón Mediator reduce la complejidad en la comunicación entre objetos introduciendo un objeto mediador que maneja la comunicación entre ellos. Esto ayuda a reducir el acoplamiento entre los objetos. Se empleó en la clase QAGeneration.

La implementación de estos patrones de diseño en este proyecto permitió una estructura de código más eficiente y sostenible, facilitando la escalabilidad y adaptabilidad del sistema a futuras necesidades o cambios.

Diagrama de Clases

Los diagramas de clases son una herramienta esencial en el desarrollo de software orientado a objetos, utilizada para visualizar la estructura estática de un sistema, facilitando la planificación, el diseño y la comunicación entre los miembros del equipo de desarrollo. Los diagramas de clases UML son un tipo específico de diagrama de clases que se utiliza en el desarrollo de software para documentar la arquitectura de un sistema, proporcionando una representación visual de las clases, sus atributos, métodos y las relaciones entre ellas (Cueto & Zuñiga, 2016).

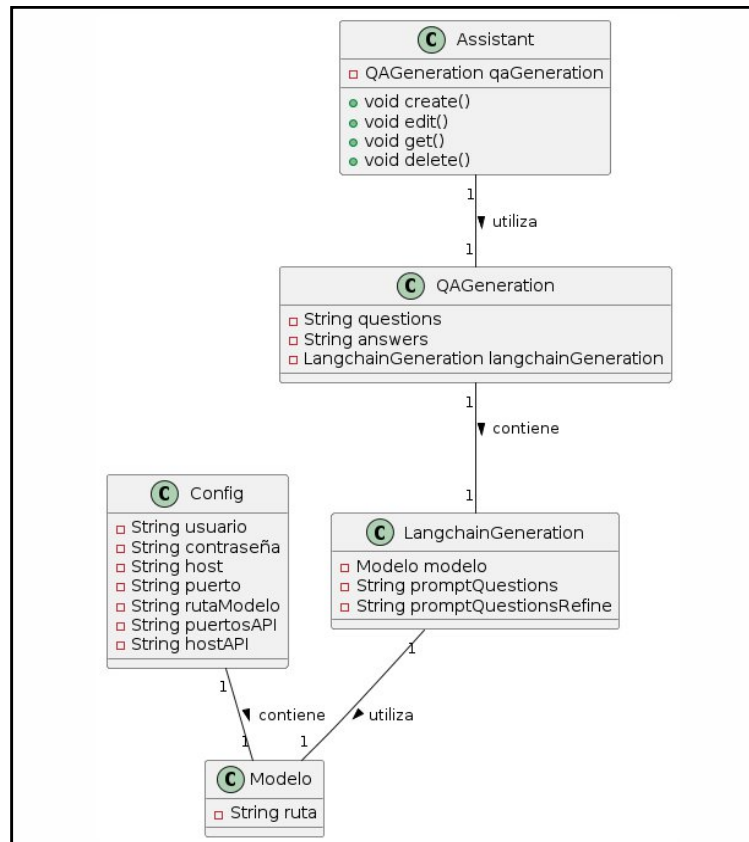


Diagrama de Clases

Algoritmos Importantes

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

Algoritmo para la creación de preguntas

Los primero es definir dos plantillas de texto para generar preguntas de prueba 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 **prueba**, 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-string⁷ o jinja2⁸) 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,
)
```

⁷ <https://docs.python.org/3/tutorial/inputoutput.html>

⁸ <https://jinja.palletsprojects.com/en/3.1.x/>

En el cuadro de texto se muestra un ejemplo de código para la carga un documento en formato PDF (será utilizado porque es más pequeño y no presenta problemas al reconocer caracteres, a diferencia de los archivos Word, que pueden corromperse y cambiar el formato de los caracteres a ASCII.) (Gil-Leiva et al., 2022), divide su contenido en fragmentos y genera preguntas basadas en esos fragmentos utilizando un modelo de lenguaje. Primero, *PyPDFLoader* carga el documento PDF desde `DATA_PATH`. Luego, se une el contenido de todas las páginas en una sola cadena de texto. Después, *RecursiveCharacterTextSplitter* divide esta cadena en fragmentos de 10000 caracteres con una superposición de 50 caracteres entre fragmentos (para mantener juntos los fragmentos que están semánticamente relacionados), este proceso permite que el modelo analice por fragmentos para poder pasarle grandes cantidades de información sin sobrepasar el número de tokens permitidos por el modelo como contexto. Estos fragmentos se convierten en documentos y se pasan a una cadena de procesamiento que utiliza un modelo de lenguaje para generar preguntas basadas en los documentos. Finalmente, las preguntas generadas se almacenan en la variable *questions*.

```
loader = PyPDFLoader(DATA_PATH)
data = loader.load()

text_question_gen = ""
for page in data:
    text_question_gen += page.page_content

text_splitter_question_gen = RecursiveCharacterTextSplitter(
    chunk_size=10000, chunk_overlap=50
)
text_chunks_question_gen= text_splitter_question_gen.split_text(
    text_question_gen
)
docs_question_gen=[Document(page_content=t) for t in
    text_chunks_question_gen]

question_gen_chain = load_summarize_chain(
    llm=LlamaCpp(
        model_path=settings.LLM_PATH,
        temperature=0.75,
        top_p=1,
        verbose=True,
        n_ctx=4096,
    ),
    chain_type="refine",
    verbose=True,
    question_prompt=PROMPT_QUESTIONS,
    refine_prompt=REFINE_PROMPT_QUESTIONS,
)

questions = question_gen_chain.run(docs_question_gen)
```

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

El ejemplo de código siguiente muestra como en la plataforma se generan respuestas a preguntas utilizando un modelo de lenguaje y una base de datos de vectores Chroma, que almacena documentos y los vectores de valores reales asociados (a partir de un modelo computacional se construcción de vectores de embeddings de HuggingFace).

Se inicia el proceso creando los vectores de embeddings y se indexa en la base de datos Chroma con el texto de los documentos y sus vectores asociados. Luego, crea una cadena

de recuperación de preguntas y respuestas que utiliza el modelo de lenguaje y la base de datos para generar respuestas a las preguntas proporcionadas.

```
embeddings = HuggingFaceEmbeddings(  
    model_name="jaimevera1107/all-MiniLM-L6-v2-similarity-es",  
    model_kwargs={"device": "cpu"},  
)  
vector_store = Chroma.from_documents(  
    docs_question_gen, embeddings  
)  
answer_gen_chain = RetrievalQA.from_chain_type(  
    llm=LlamaCpp(  
        model_path=settings.LLM_PATH,  
        temperature=0.75,  
        top_p=1,  
        verbose=True,  
        n_ctx=4096,  
    ),  
    chain_type="stuff",  
    retriever=vector_store.as_retriever(k=2),  
)  
  
question_list = questions.questions  
answers = []  
for idx, question in enumerate(question_list):  
    answer = answer_gen_chain.run(question)  
    answers.append(f"{idx+1} - {answer}")
```

Algoritmo de Reconocimiento del habla y transcripción

Los asistentes generados, por la plataforma, tienen una interfaz que permite que el usuario se comuniquen con ellos usando reconocimiento de audio. Cada asistente virtual permite la transcripción de voz a texto mediante un algoritmo diseñado en esta propuesta que tiene como base el uso de Whisper.

El ejemplo de código que se muestra a continuación, es parte de este algoritmo y, 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 de tipo *UploadFile* de FastAPI. Luego, el código lee los bytes del archivo de audio y los guarda en un archivo temporal llamado `audio.mp3`. A continuación, se carga el archivo de audio usando la biblioteca `"whisper"` y se ajusta su duración si es necesario para que tenga una duración fija. Después, se calcula el espectrograma logarítmico de Mel del audio. Se configuran las

opciones de decodificación, como el uso de punto flotante de 32 bits y el idioma español. Finalmente, se realiza la decodificación del audio utilizando un modelo preentrenado proporcionado por la variable "model" y se devuelve el texto resultante. El archivo de audio temporal se elimina al finalizar el proceso.

```
@app.post("/whisper/audio")
async def recive_audio(file: UploadFile=File(...)):
    audio_bytes = file.file.read()
    with open ("audio.mp3" , "wb") as f:
        f.write(audio_bytes)
    audio = whisper.load_audio("audio.mp3")
    audio = whisper.pad_or_trim(audio)
    mel = whisper.log_mel_spectrogram(audio).to(model.device)
    options = whisper.DecodingOptions(fp16=False, language="es")
    result = whisper.decode(model,mel,options)
    os.remove("audio.mp3")
    print(result.text)
    return result.text
```

Interfaz gráfica resultante

A continuación, se muestra la interfaz de usuario del apartado de creación de asistente virtual en modo oscuro, en esta se observa que el usuario está utilizando dos archivos PDF para de ellos extraer el conocimiento para la creación de su asistente virtual.

The screenshot shows the 'EvoAssist' web interface in a dark theme. On the left, under 'Datos del asistente', there are three input fields: 'Nombre' (with a red asterisk), 'Descripción', and 'Conocimiento del asistente'. Below these is a 'Cargar Archivo' button and two file selection buttons labeled 'PoliticaPrivacidad enzona.pdf' and 'Orientación del seminario I de Redes.pdf'. A note specifies 'Tipos de archivos permitidos: PDF, DOCX, TXT'. At the bottom of this section is a large purple 'Analizar' button. On the right, under 'Preguntas y Respuestas generadas', there are two sections: 'Preguntas' and 'Respuestas', each with a text area and a note: 'Podrás editar este contenido cuando se analice el conocimiento.' At the bottom of the right section are two buttons: 'Generar Archivos' (purple) and 'Guardar resultados' (yellow).

Interfaz de usuario de la creación de un asistente virtual

Pruebas al sistema

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

Durante el desarrollo de la aplicación, se llevaron a cabo múltiples iteraciones, cada una de ellas acompañada de un conjunto de pruebas de funcionalidad y aceptación. Estas pruebas se elaboraron sobre la base de las historias de usuarios, lo que permitió corroborar que cada historia había sido implementada de manera adecuada. A lo largo del proceso, se diseñó y ejecutó un total de 9 casos de prueba, distribuidos en 4 iteraciones, con el objetivo de verificar el funcionamiento de la aplicación 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”

CP	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.”

CP	Entrada	Resultado Esperado	Condición
CP1	Documento proporcionado para generar preguntas y respuestas	Preguntas generadas y almacenadas en Chroma DB. Respuestas adecuadas generadas y almacenadas en la base de datos. Archivos de	El algoritmo analiza correctamente el contenido del documento y genera preguntas pertinentes. Las respuestas generadas son precisas y relevantes. Los

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

Tabla 3.9 Caso de Prueba “Probar los asistentes creados”

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

En el transcurso de estas pruebas, se identificaron y documentaron un total de 8 errores, los cuales fueron priorizados y solucionados. El tiempo dedicado a la ejecución de pruebas y la resolución de errores fue de aproximadamente 15 horas, distribuidas a lo largo del ciclo de desarrollo de la aplicación.

Encuestas a usuarios

En este proyecto, con el objetivo de validar el software y evaluar su aceptación, se recurrió a la utilización de encuestas. Esta herramienta de recolección de datos resulta altamente beneficiosa debido a su capacidad de capturar opiniones y percepciones de un grupo representativo de usuarios de manera eficiente y estructurada. Las encuestas permiten obtener información valiosa y cuantificable sobre la experiencia del usuario, facilitando la identificación de aspectos positivos y áreas de mejora en el software. Además, brindan la posibilidad de realizar análisis estadísticos y comparativos, lo que contribuye a tomar decisiones informadas y respaldadas por datos concretos (Joskowicz, J. 2008).

Esta encuesta se basará en las evaluaciones de calificación por 5 estrellas o puntos las cuales son un método popular para recopilar opiniones de clientes sobre productos o servicios. Estas evaluaciones funcionan como una escala de Likert y se basan en teorías como la Teoría de la Respuesta al Ítem, la Teoría de la Utilidad Esperada y la Teoría de la Satisfacción del Cliente. Estas teorías ayudan a comprender cómo las personas responden a las calificaciones, toman decisiones y evalúan su satisfacción (Luna, 2007).

Esta encuesta constituye una herramienta esencial para este proyecto, que busca comprender y satisfacer mejor las necesidades y expectativas de los usuarios, lo que se traduce en un mayor éxito y crecimiento a largo plazo.

En esta encuesta se propone alcanzar una puntuación de más de 4.0 lo cual significaría que la aplicación tiene un buen nivel de aceptación.

La encuesta tenía las siguientes preguntas:

1. Califica de 1 a 5 el diseño de la aplicación, considerando los colores y la disposición de elementos como botones, formularios y títulos.
2. Califica de 1 a 5 tu experiencia al crear un asistente, tomando en cuenta su velocidad.
3. Califica de 1 a 5 la relevancia y corrección de las preguntas y respuestas generadas por el asistente con relación a tu documento.
4. Califica de 1 a 5 la facilidad para crear un asistente (1 siendo muy difícil y 5 muy fácil).

5. Califica de 1 a 5 la facilidad y la **comodidad visual** para editar un asistente (1 no intuitivo, 5 muy fácil).
6. Califica de 1 a 5 tu experiencia al interactuar con un asistente por mensajes de texto y la satisfacción con las respuestas.
7. Califica de 1 a 5 la rapidez de respuesta del asistente cuando interactúas con él.
8. Califica de 1 a 5 qué tan natural y fluida es la interacción con el asistente virtual usando comandos de voz.
9. Califica de 1 a 5 la precisión del reconocimiento de emociones y si usualmente identificaba correctamente las tuyas.
10. Califica de 1 a 5 el diseño de la interfaz de los asistentes creados.

Estas preguntas se les hicieron a 23 usuarios, de ellos 5 personas de más de 60 años (2 hombres y 3 mujeres), 10 personas entre 30 y 59 años de edad (5 mujeres y 5 hombres), 8 personas entre 16 y 29 años de edad (4 hombres y 4 mujeres) y este es el resultado que se obtuvo una media de calificación final de 4.5, ver la [tabla de las encuestas](#), para visualizar cada una de las respuestas de cada usuario, esta prueba logra el objetivo propuesto el cual era obtener una puntuación de más de 4.0.

Análisis económico del costo de producción del sistema

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

Estimación de costo y tiempo

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

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

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

$$UUCP = UAW + UUCW$$

Donde:

UUCP: Puntos de Historias de Usuarios sin ajustar.

hUAW: Factor de Peso de los Actores sin ajustar.

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

Factor de Peso de los Actores sin ajustar (UAW)

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

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

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

Cálculo del UUCW

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

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

$$\text{UCP} = \text{UUCP} * \text{TCF} * \text{EF}$$

Donde:

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

Factor de complejidad técnica (TCF).

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

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

$$\text{TCF} = 0.6 + 0.01 * \Sigma (\text{Peso} * \text{Valor})$$

$$\text{TCF} = 0.6 + 0.01 * 40$$

$$\text{TCF} = 1$$

Factor Ambiente (EF).

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

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

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

Ecuación 3.4: Cálculo del Factor de ambiente

$$EF = 1.4 - 0.03 * \sum (\text{Peso}i * \text{Valor}i)$$

$$EF = 1.4 - 0.03 * 9.5$$

$$EF = 0.965$$

$$\text{Luego: } UCP = UUCP * TCF * EF$$

$$UCP = 42 * 1 * 0.965$$

$$UCP = 40.53$$

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:

$$CF = 20 \text{ Horas-hombre} / \text{Puntos de historias de usuarios.}$$

Luego

$$E = 40.53 * 20 \text{ horas-hombre}$$

$$E = 810.6 \text{ horas-hombre}$$

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

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

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

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

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

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

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

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

Entonces el costo total del proyecto:

$$C = E (\text{Total}) * K * \text{THP}$$

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

El costo estimado de este proyecto es de unos \$103 627 pesos cubanos. Para el salario a los trabajadores se investigó como realizan el pago en diferentes organizaciones o sucursales en Santiago de Cuba de diferente sector ya sea estatal o privado, dada las nuevas regulaciones y tasas de cambio. Los datos se obtuvieron a través de trabajadores de las entidades y por anuncios laborales. Para una información más detallada sobre la estimación de pago de trabajadores ver la [tabla 3.6](#).

Conclusiones del capítulo

En este capítulo, se finalizó e inspeccionó el sistema propuesto, incluyendo elementos fundamentales como el ingreso de usuarios, la creación de preguntas y respuestas, y el reconocimiento de voz. Se explicó la metodología de prueba, la cual incluye casos que van desde el registro de usuarios hasta la interacción con los asistentes virtuales lo cual arrojó en 4 iteraciones de 9 casos de prueba un total de 8 errores los cuales se solucionaron en un aproximado de 15 horas. También se utilizaron encuestas como método de prueba, obteniendo una aceptación de 4.5 de 5 puntos tras realizar una encuesta de 10 preguntas a 20 usuarios. Por último, se efectuó un análisis económico para calcular el costo del proyecto el cual fue de \$ 103 627 pesos cubanos, teniendo en cuenta aspectos como la complejidad técnica y los recursos humanos requeridos.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. Se evidenciaron las carencias que presentaba la investigación "Herramienta digital para la construcción de conocimiento automático para un Asistente Virtual" del año 2022, lo cual permitió determinar el objetivo general que persiguió esta investigación
2. Mediante el estudio del estado del arte se estableció una base sólida para la modificación del algoritmo de generación de preguntas y respuestas, así como para la integración de un modelo de lenguaje.
3. Se logró la creación de un asistente virtual con soporte para el envío y respuesta de mensajes de audio y el reconocimiento de las emociones faciales (enojo, alegría, ira) que permite una mejor interacción hombre-máquina. El análisis exhaustivo de tecnologías como Whisper para el procesamiento de audio y la transcripción a audio texto una integración exitosa de la plataforma, promoviendo una experiencia de usuario más accesible e interactiva.
4. Se mejoró la carencia de la propuesta anterior, asociada a la generación de la base de conocimiento de un asistente virtual mediante la implementación de un nuevo algoritmo de generación de preguntas y respuestas que utiliza inteligencia artificial generativa mediante un gran modelo del lenguaje para poder obtener mejores resultados.
5. Se diseñó e implementó una plataforma web para la gestión y edición de asistentes virtuales que permite brindar soluciones a diversas instituciones y sectores mediante la creación de asistentes virtuales a la medida de las necesidades de información.
6. Las pruebas de la plataforma han validado la viabilidad de la solución propuesta, demostrando su capacidad para comprender y responder a las necesidades de los usuarios de manera más efectiva, especialmente en lo que respecta a su estado emocional y requisitos de comunicación específicos.

Recomendaciones

1. Mejorar la interfaz de usuario del sistema para aumentar la experiencia visual positiva del usuario.
2. Realizar pruebas con el Gran Modelo de Lenguaje Llama3 y phi3 recientemente liberado por Meta y Microsoft de forma libre.

3. Continuar el proceso de pruebas y aumentar estas en cuanto a la escalabilidad a partir de contar con mayores recursos de cómputo.
4. Crear un sistema de tareas con Celery⁹ para la creación de los asistentes virtuales, ya que puede ser una tarea que pueda durar alrededor de 5 a 10 minutos, dependiendo de las prestaciones del servidor.
5. Agregar otra base de conocimiento al asistente virtual” que pueda funcionar como memoria temporal o cache.

⁹ <https://docs.celeryq.dev/en/stable/getting-started/introduction.html>

REFERENCIAS BIBLIOGRÁFICAS

- ¿Qué es Python? - Explicación del lenguaje Python - AWS. (n.d.). Amazon Web Services, Inc (accedido 21 de noviembre del 2023). <https://aws.amazon.com/es/what-is/python/>
- Gil-Leiva, I., Fujita, M. S. L., Redigolo, F. M., & Saran, J. F. (2022). Extracción de información de documentos PDF para su uso en la indización automática de e-books. *Transinformação*, 34, e210069.
- A. (2021, December 16). What is RASA? — the open-source AI for building conversational chatbots. Medium. <https://medium.com/@asklua/what-is-rasa-the-open-source-ai-for-building-conversational-chatbots-8a86bef47ec4>
- A. (2023, November 9). FastAPI explicado en 5 minutos o menos. Geekflare.
- Acosta, E. V. (2023, March 3). Visual Studio vs Visual Studio Code: ¿Cuál es la diferencia entre estos editores de código IDE? freeCodeCamp.org. <https://www.freecodecamp.org/espanol/news/visual-studio-vs-visual-studio-code-cual-es-la-diferencia-entre-estos-editores-de-codigo-ide/>
- Argüello, F. (2024, March 22). *10 principales API de Reconocimiento Facial - Infoteknico*. Infoteknico. <https://www.infoteknico.com/principales-api-de-reconocimiento-facial/>
- Asistente virtual: retos y posibilidades para la inteligencia artificial. (n.d.). Tableau. <https://www.tableau.com/data-insights/ai/ai-virtual-assistant>
- Boisdequin, H. (2020, November 30). React vs Vue vs Angular vs Svelte. DEV Community. <https://dev.to/hb/react-vs-vue-vs-angular-vs-svelte-1fdm>
- Bueno, P. C. (2023, December 18). Qué es Django y por qué usarlo. OpenWebinars.net. <https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/>
- Caballero, Rosbel. (2021). Chatbot: a viable proposition for customer service in the UCI support center. https://www.researchgate.net/publication/357689698_Chatbot_a_viable_proposition_for_customer_service_in_the_UCI_support_center
- Canive, T. (2020, May 27). Metodología XP o Programación Extrema: ¿Qué es y cómo aplicarla? Gestor De Proyectos Online. <https://www.sinnaps.com/blog-gestion-proyectos/metodologia-xp>

- O'Sullivan, J., Bogaarts, G., Kosek, M., Ullmann, R., Schoenenberger, P., Chatham, C., ... & Lipsmeier, F. Automatic Speech Recognition for ASD Using the Open-Source Whisper Model from OpenAI.
- ChatBot, P., & ChatBot, P. (2022, June 6). Dialogflow de Google cambia su modelo de pricing: ¿cuánto cuesta la IA / NLU? - Planeta Chatbot. Planeta Chatbot - Comunidad De Expertos En IA Conversacional. <https://planetachatbot.com/dialogflow-google-cambia-modelo-pricing/>
- Pillai, M., & Thakur, P. (2024, January). Developing a Website to Analyze and Validate Projects Using LangChain and Streamlit. In 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT) (pp. 1493-1501). IEEE.
- Cómo elegir un framework para el backend. (n.d.). CódigoFacilito. <https://codigofacilito.com/articulos/elegir-framework-backend>
- Condez, A. (2022, February 13). Companies That Use Flask. Career Karma. <https://careerkarma.com/blog/companies-that-use-flask/>
- de Oliveira, G., Venson, R., & Marcelino, R. (2018). REDES NEURAIAS APLICADAS NO DESENVOLVIMENTO DE CHATBOTS: UMA ANÁLISE BIBLIOMÉTRICA. ARTEFACTUM-Revista de estudos em Linguagens e Tecnologia, 17(2). https://www.academia.edu/72166314/Redes_Neuraais_Aplicadas_No_Desenvolvimento_De_Chatbots_Uma_An%C3%A1lise_Bibliom%C3%A9trica
- Dhungana, K. (2023, noviembre 10). An Overview of ChromaDB: The Vector Database. Medium. <https://medium.com/@kdbhunga/an-overview-of-chromadb-the-vector-database-206437541bdd>
- Joskowicz, J. (2008). Reglas y prácticas en eXtreme Programming. Universidad de Vigo, 22.
- Luna, S. M. M. (2007). Manual práctico para el diseño de la Escala Likert. *Xihmai*, 2(4).
- Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P. E., Lomeli, M., Hosseini, L., & Jégou, H. (2024, January 16). The Faiss library. arXiv.org. <https://doi.org/10.48550/arXiv.2401.08281>
- Dragos, P. (2021). Overview of the Agile Rational Unified Process (Rup) in the Context of Software Development Projects. Journal of Business and Economics, 12(6), 681-684. <https://repository.unpak.ac.id/tukangna/repo/file/files-20211015150215.pdf#page=107>
- Edwards, B. (2023, May 11). The AI race heats up: Google announces PaLM 2, its answer to GPT-4. Ars Technica. <https://arstechnica.com/information->

technology/2023/05/googles-top-ai-model-palm-2-hopes-to-upstage-gpt-4-in-generative-mastery/

- Funcionamiento de Amazon Lex - Amazon Lex V1. (n.d.). https://docs.aws.amazon.com/es_es/lex/latest/dg/how-it-works.html
- Elmasri, R., Navathe, S. B., Castillo, V. C., Pérez, G. Z., & Espiga, B. G. (2007). Fundamentos de sistemas de bases de datos. Pearson educación.
- Han, Y., Liu, C., & Wang, P. (2023). A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv preprint arXiv:2310.11703*. <http://arxiv.org/abs/2310.11703>
- IBM watsonx Assistant Virtual Agent. (n.d.). <https://www.ibm.com/products/watsonx-assistant>
- Khurana, D., Koli, A., Khatter, K., & Singh, S. (2022, July 14). Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, 82(3), 3713–3744. <https://doi.org/10.1007/s11042-022-13428-4>
- Kumari, P. (2024, February 14). Llama 2 Vs GPT-3.5 Vs GPT-4: What, When & How To Chose. *Labellerr*. <https://www.labellerr.com/blog/llama2-vs-gpt-3-5-vs-gpt-4/>
- Londoño, P. (2023, April 4). ¿Qué es Python, para qué sirve y cómo se usa (+ recursos para aprender). <https://blog.hubspot.es/website/que-es-python>
- Luzniak, K. (2023, November 17). Is Llama 2 Better Than GPT Models? 6 Main Differences Between Llama 2 vs. GPT-4 vs. GPT-3.5. *Neoteric*. <https://neoteric.eu/blog/6-main-differences-between-llama2-gpt35-and-gpt4/>
- Malik, F. (2021, December 15). Build And Host Fast Data Science Applications Using FastAPI. *Medium*. <https://towardsdatascience.com/build-and-host-fast-data-science-applications-using-fastapi-823be8a1d6a0>
- Mancuzo, G. (2024, March 8). Metodología XP: La Mejor Vía para el Desarrollo de Software. *Blog - ComparaSoftware*. <https://blog.comparasoftware.com/metodologia-xp/>
- Mindbowser. (2018, April 24). 5 learnings from our ‘Chatbot Survey — 2017’ - *Chatbots Journal*. *Medium*. <https://chatbotsjournal.com/5-learnings-from-our-chatbot-survey-2017-72a6a4fc209c?gi=21ae27bf4d12>
- Mustafa, A., Wan-Kadir, W. M., Ibrahim, N., Shah, M. A., Younas, M., Khan, A., ... & Alanazi, F. (2021). Automated test case generation from requirements: A systematic literature review. *Computers, Materials and Continua*, 67(2), 1819-1833. https://www.researchgate.net/profile/Ahmad-Mustafa-7/publication/349096514_Automated_Test_Case_Generation_from_Requirements_A

[Systematic_Literature_Review/links/601f6f924585158939892bfc/Automated-Test-Case-Generation-from-Requirements-A-Systematic-Literature-Review.pdf](#)

- Pardo Gómez, M. E. (2021). La Lógica del proceso de investigación científica. Recuperado de <https://eva.uo.edu.cu/mod/folder/view.php?id=123721>
- Payne, R. (2022, January 19). React, Angular, Vue, and Svelte: A Comparison. <https://www.accelebrate.com/blog/react-angular-vue-svelte-comparison>
- Pérez, E. (2023, March 12). He usado Whisper para transcribir una entrevista: es la herramienta que llevaba esperando desde hace años. Xataka. <https://www.xataka.com/aplicaciones/he-usado-whisper-para-transcribir-entrevista-herramienta-que-llevaba-esperando-hace-anos>
- Primeros pasos con Botpress | Botpress Blog. (n.d.). <https://botpress.com/es/blog/getting-started-with-botpress>
- Qué es Git | Atlassian Git Tutorial. (n.d.). Atlassian. <https://www.atlassian.com/es/git/tutorials/what-is-git>
- Singh, P. N., Talasila, S., & Banakar, S. V. (2023, December). Analyzing Embedding Models for Embedding Vectors in Vector Databases. In 2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG) (pp. 1-7). IEEE. <https://ieeexplore.ieee.org/abstract/document/10455990/>
- Sobrino, D. C. (2020, May 15). ¿Cómo funciona el reconocimiento automático del habla? Medium. <https://medium.com/soldai/c%C3%B3mo-funciona-el-reconocimiento-autom%C3%A1tico-del-habla-eb038ecfe72e>
- Team, P. (2022, January 14). Advantages of Django | Disadvantages of Django. Python Geeks. <https://pythongeeks.org/advantages-disadvantages-of-django/>
- Guerrero, C. A., Suárez, J. M., & Gutiérrez, L. E. (2013). Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. Información tecnológica, 24(3), 103-114.
- TechEmpower Framework Benchmarks. (n.d.). <https://www.techempower.com/benchmarks/#section=datar20&hw=ph&test=fortune&l=zijzen-sf>
- Topsakal, O., & Akinci, T. C. (2023, July). Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In International Conference on Applied Engineering and Natural Sciences (Vol. 1, No. 1, pp. 1050-1056). https://www.researchgate.net/profile/Oguzhan-Topsakal/publication/372669736_Creating_Large_Language_Model_Applications_Utili

[zing_LangChain_A_Primer_on_Developing_LLM_Apps_Fast/links/64d114a840a524707ba4a419/Creating-Large-Language-Model-Applications-Utilizing-LangChain-A-Primer-on-Developing-LLM-Apps-Fast.pdf](https://www.fast.ai/2020/07/14/langchain-a-primer-on-developing-llm-apps-fast/)

- Cueto, M. J. J. F., & Zuñiga, C. B. Diagrama de clases en UML. Diagrama de clases en UML.[En línea][Citado el: 25 de febrero de 2016.] <http://es.scribd.com/doc/31096724/Diagrama-de-Clases-en-UML#scribd>.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288. <https://sh-tsang.medium.com/brief-review-llama-2-open-foundation-and-fine-tuned-chat-models-6666eb8b56b7>
- Villanueva, E., & Molina, D. F. Aporte a RUP para el Desarrollo de Aplicaciones Web Transaccionales–Requerimientos no Funcionales y Administración de Riesgos. https://www.academia.edu/download/32737879/paper_rup.pdf
- Williams, L. (2003). The xp programmer: The few-minutes programmer. IEEE software, 20(3), 16. <https://collaboration.csc.ncsu.edu/laurie/Papers/fewMinutes.pdf>
- Xu, B. (2023, February 23). Domain-Specific Bots vs. Generic Bots - DealerAI. DealerAI. <https://dealerai.com/domain-specific-bots-vs-generic-bots/>
- Duvalón, J. E. (2022). Herramienta digital para la construcción de conocimiento automático para un Asistente Virtual [Pregrado, Universidad de Oriente]. https://github.com/Neto99d/Gen_file_train_Rasa/blob/main/Tesis/Tesis.zip

ANEXOS

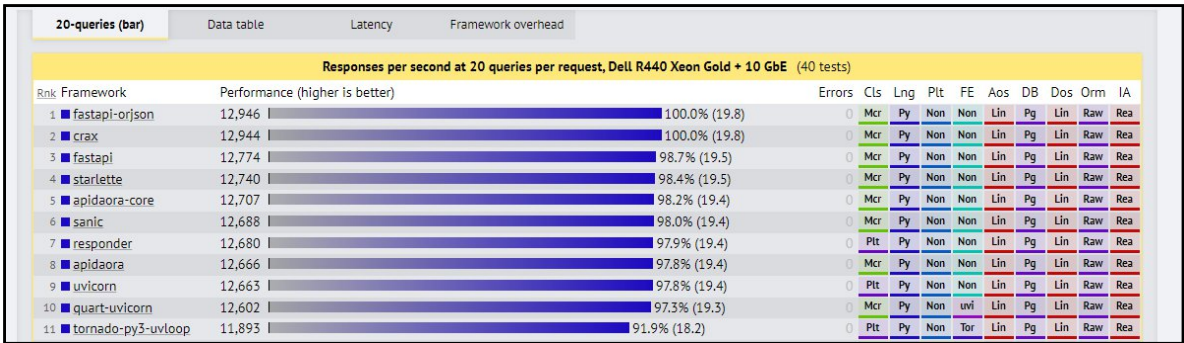


Figura 1 Prueba de marco de evaluación para 20 peticiones.

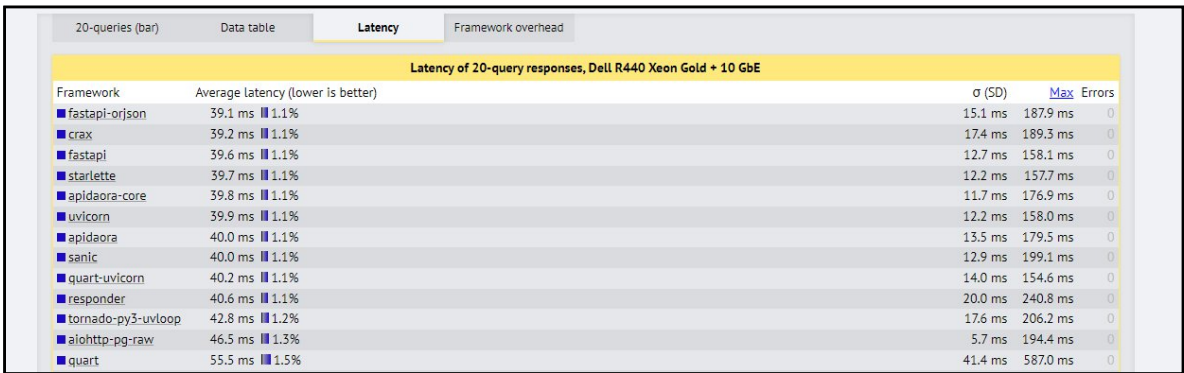


Figura 2 Prueba de marco de evaluación de la latencia con 20 respuestas.

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

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

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

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 de encuestas a 20 usuarios

[illegible]

Usuario	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Media Usuario
15	5	5	5	5	5	5	5	5	5	5	5.0
16	5	5	5	5	5	5	5	5	5	5	5.0
17	4	4	4	4	4	4	4	4	4	4	4.0
18	5	5	5	5	5	5	5	5	5	5	5.0
19	5	5	5	5	5	5	5	5	5	5	5.0
20	4	4	4	4	4	4	4	4	4	4	4.0

3.1. Factor de Peso de los Actores sin ajustar.

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

3.2. Factor de Peso de Historias de Usuarios sin ajustar

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

3.3. Factor de Complejidad Técnica

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

3.4 Factor de ambiente

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

3.5 Distribución del esfuerzo por etapas

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

3.6 Situación actual de pago.

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

Aval del proyecto

8 de abril de 2024

"Año del 66 aniversario del triunfo de la Revolución"

A quién pueda interesar:

A través de la siguiente carta quiero reconocer el aporte y los resultados satisfactorios de la estudiante Claudia Queipo García (4to de Ingeniería Informática) y su investigación de pregrado "Plataforma de creación automática de asistentes virtuales con audio y video. Esta investigación permitió el desarrollo de un Asistente Virtual que interactúa con la plataforma web WebIND para la gestión de información del Servicio de Neurodesarrollo Infantil y Discapacidad del Hospital Infantil Sur de la ciudad de Santiago de Cuba, por parte de pacientes (y cuidadores) y personal médico de ese servicio.

Esta investigación es parte de los resultados del proyecto "Digitalización del Servicio del Neurodesarrollo Infantil y Discapacidad en la provincia Santiago de Cuba (DiSeNID)", código PS_008 del MINCOM.



Dr.C Sergio Daniel Cano Ortiz
Jefe del Proyecto

Resultados en el Fórum Nacional de Ciencias Técnicas



XXVI Fórum Nacional de Estudiantes Universitarios de Ciencias Técnicas

El comité organizador otorga la categoría

RELEVANTE

A: Claudia Queipo García y Luis Andrés Licea Berenguer.

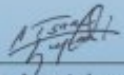
Con la ponencia:

Creación de Asistentes conversacionales para lograr
una atención más eficaz.

*“Solo el camino del trabajo, de la técnica y de la ciencia
es lo que hace progresar a la humanidad”*

Fidel Castro Ruz, 24 de febrero de 1963

*Dado en Santiago de Cuba, a los 29 días del mes de junio de 2023
“Año 65 de la Revolución”*


Carlos Antonio Leyva Isaac
Presidente FEU-UO


Dra. C. Diana Sedal Yanes
Rectora UO