

Universidad de Oriente

Sede “Julio Antonio Mella”

Facultad de Ingeniería en Telecomunicaciones, Informática y Biomédica

Trabajo de Diploma

En opción al título de Ingeniera en Informática

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

**Autora:** Claudia Queipo García.

**Tutor:** Dr. C. Dionis López Ramos.

Santiago de Cuba,

“Año de la Revolución”

Resumen

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

En esta tesis, se aborda la creación de asistentes virtuales utilizando el marco de desarrollo de software RASA y la integración de herramientas de procesamiento del lenguaje natural, como Spacy, para mejorar la capacidad del asistente para comprender y responder a las preguntas y solicitudes del usuario. Además, se implementaron tecnologías de reconocimiento de voz para convertir audio a texto y permitir al asistente virtual interactuar con el usuario a través de la voz.

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

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

*ABSTRACT*

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

This thesis addresses the creation of virtual assistants using the RASA software development framework and the integration of natural language processing tools, such as Spacy, to enhance the assistant's ability to understand and respond to user questions and requests. Additionally, voice recognition technologies were implemented to convert audio into text and enable the virtual assistant to interact with the user through voice.

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

Keywords: Virtual Assistant, RASA, Voice Recognition, Whisperíndice

[INTRODUCCIÓN 6](#_Toc129849672)

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

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

[1.2 Asistente Personal Inteligente 11](#_Toc129849675)

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

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

[1.4.1 DialogFlow 13](#_Toc129849678)

[1.4.2 Amazon Lex 14](#_Toc129849679)

[1.4.3 IBM Watson 15](#_Toc129849680)

[1.4.4 RASA Framework 16](#_Toc129849681)

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

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

[Python 18](#_Toc129849684)

[Visual Studio Code 19](#_Toc129849685)

[RASA 20](#_Toc129849686)

[Whisper 20](#_Toc129849687)

[Mongo DB 20](#_Toc129849688)

[FastAPI 21](#_Toc129849689)

[Git 22](#_Toc129849690)

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

[SCRUM 22](#_Toc129849692)

[Conclusiones del capítulo 23](#_Toc129849693)

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

[2.1 Sprint Backlog 24](#_Toc129849695)

[2.2 Arquitectura del Sistema 25](#_Toc129849696)

[2.3 Requisitos no Funcionales 25](#_Toc129849697)

[2.3.1 Requisitos de Software 25](#_Toc129849698)

[2.3.2 Requisitos de Hardware 25](#_Toc129849699)

[2.4 Diagrama de Secuencia 26](#_Toc129849700)

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

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

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

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

[CONCLUSIONES Y RECOMENDACIONES 30](#_Toc129849705)

[REFERENCIAS BIBLIOGRÁFICAS 31](#_Toc129849706)

[GLOSARIO DE TÉRMINOS 35](#_Toc129849707)

[ANEXOS 36](#_Toc129849708)

INTRODUCCIÓN

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

Los chatbot tuvieron sus inicios en los años 60, simultáneamente con el procesamiento del lenguaje natural (NLP, Natural Language Processing), en un sistema de preguntas y respuestas (QA systems, question answering systems). A su vez, se denominan como un modelo característico de sistemas de conversaciones que ejecutan tareas específicas de interacción textual [1].

Hay tres partes principales en un sistema de chatbot: una parte de comprensión del lenguaje natural (NLU) que obtiene las intenciones del usuario, una parte de administración de diálogo que monitorea el sistema actual y el estado de la conversación, y una parte de generación de lenguaje natural que responde al usuario. A medida que pasaron los años y la tecnología se hizo más eficiente, los chatbots se volvieron más versátiles y utilizados, aunque el principal catalizador podría haber sido el impulso de Facebook por “Messenger Chatbots” [2], donde se popularizó en gran medida su uso. Cabe destacar el trabajo de varias plataformas líderes en la nube para la creación de chatbots: DialogFlow de Google, wit.ai de Facebook, Microsoft LUIS, IBM Watson Conversation, Amazon Lex, SAP Conversation AI, y otras plataformas conocidas como RASA, Botsify, Chatfuel, Manychat, Flow XO, Chatterbot, Pandorabots, Botkit y Botlytics.

A partir del 2020 con la crisis desatada por el COVID19 los chatbots han emergido como una herramienta clave para la lucha global contra la pandemia, ya que permite la identificación de personas con síntomas y muchas otras enfermedades con un alto grado de precisión, además de proporcionar respuestas precisas a la mayoría de las preguntas, por medio de fuentes creíbles de información ante la gran cantidad de personas necesitadas de consejos [3].

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

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

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

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

Por otro lado, los chatbots de dominios generales son aquellos que han sido diseñados para comprender el lenguaje natural en general y responder preguntas sobre una amplia variedad de temas. Estos chatbots no están especializados en un campo particular de conocimiento y pueden responder preguntas generales sobre temas como el clima, las noticias y el entretenimiento [4].

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

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

**Problema de investigación:**

El problema de investigación de esta tesis se centra en el desarrollo de asistentes virtuales que utilicen canales de texto, audio y video, con el objetivo de abordar las dificultades de atención y comunicación que enfrentan las personas ancianas, discapacitadas, con problemas visuales y niños. Se busca proporcionar respuestas precisas y adaptadas a su estado emocional, brindando una experiencia más efectiva y personalizada para estos grupos.

**Objeto de estudio:**

Reconocimiento de voz para el asistente conversacional

**Campo de Estudio:**

Asistentes Virtuales

**Campo de Acción:**

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

**Objetivo General:**

Integrar la tesis de Ernesto Duvalón y mejorar la experiencia del usuario al agregar funciones de audio (comunicación en lenguaje natural) y reconocimiento de emociones faciales al asistente virtual, con el objetivo de brindar respuestas adaptadas y comprender las necesidades de los usuarios .

**Objetivos Específicos:**

Análisis del estado del arte.

Estudio de la investigación de José Ernesto Duvalón

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

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

Implementación de la plataforma

* Pruebas y despliegue de la plataforma

**Hipótesis:**

Se desarrollará un software que permita la integración de RASA, Whisper y FaceAPI para la inclusión del canal de voz y reconocimiento de emociones en el asistente virtual que se va a diseñar, con el objetivo de mejorar la interacción con los usuarios y ofrecer una experiencia más personalizada y satisfactoria en la resolución de dudas y consultas. Esta herramienta permitirá una mayor precisión en el reconocimiento de voz y una mejor interpretación de las emociones de los usuarios, lo que se traducirá en una mayor eficiencia en las respuestas ofrecidas por el 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.

**Estructura del informe:**

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

diferentes aspectos teóricos y su fundamentación.

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

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

posibles resultados a obtener.

Conclusiones, Recomendaciones, Referencias Bibliográficas y Anexos.

**Aportes de la investigación:**

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

# MARCO TEÓRICO

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

## Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural (NLP, por sus siglas en inglés) es una subárea de la inteligencia artificial que se enfoca en el análisis y la comprensión del lenguaje humano. Se trata de un campo multidisciplinario que combina aspectos de la lingüística, la computación y la psicología para desarrollar sistemas capaces de comprender y producir textos en una forma natural.

El NLP tiene una amplia variedad de aplicaciones, desde la traducción automática hasta la detección de sentimientos en textos. También se utiliza en aplicaciones prácticas como la respuesta automatizada a preguntas, la clasificación de documentos y la generación de resúmenes.

Uno de los retos más importantes en el NLP es la ambigüedad inherente en el lenguaje humano. Las palabras pueden tener múltiples significados y las frases pueden tener múltiples interpretaciones, lo que hace que sea difícil para las máquinas entender el verdadero significado de un texto. Sin embargo, los avances en el aprendizaje profundo y el análisis semántico han permitido a los investigadores superar estos desafíos y desarrollar sistemas NLP cada vez más sofisticados.

En resumen, el Procesamiento del Lenguaje Natural es un campo en constante evolución que tiene el potencial de transformar la forma en que las máquinas y los humanos interactúan a través del lenguaje [5].

## Asistente Personal Inteligente

Un asistente personal inteligente es un agente tipo software que puede realizar tareas u ofrecer servicios a un individuo. Estas tareas o servicios están basados en datos de entrada de usuario, reconocimiento de ubicación y la habilidad de acceder a información de una variedad de recursos en línea (como al clima o al tráfico, noticias, precios de acciones, horario del usuario, precios al por menor, etc). Algunos ejemplos de asistentes personales son Siri de Apple, Braina, Google Assistant y Google Now de Google, Amazon Echo de Amazon, Cortana de Microsoft, S Voice y Bixby de Samsung, Voice Mate, Silvia y Hidi de HTC y Celia de Huawei. Uno de los aspectos clave de un asistente personal inteligente es su habilidad para organizar y mantener información. Esto incluye el manejo de correos electrónicos, eventos en el calendario, archivos, lista a seguir, etc [6].

## Reconocimiento Automático del habla

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

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

El reconocimiento automático del habla (ASR, Automatic Speech Recognition por sus siglas en inglés) es un proceso en el que un sistema convierte la señal acústica de una expresión hablada en texto escrito. El proceso utiliza un enfoque probabilístico para asignar una puntuación a cada secuencia de palabras candidatas, evaluando su propiedad fonética, conocimiento lingüístico, contigüidad de las palabras y gramática para seleccionar la mejor opción. El esquema típico del proceso de ASR incluye preprocesamiento, extracción de rasgos característicos, decodificación y posprocesamiento del resultado. El preprocesamiento incluye la segmentación de la señal, la reducción del ruido de fondo y la identificación de las partes de silencio entre frases. La extracción de rasgos característicos implica la división de la señal acústica en ventanas de tiempo y la aplicación de transformaciones matemáticas, filtros y procesos de normalización para obtener un vector de coeficientes representativo de la señal. En la etapa de decodificación, se calcula la secuencia de palabras más probable de corresponder a la señal representada por los vectores de rasgos característicos [7].

## Estado del arte de las herramientas

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

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

### DialogFlow



*Figura 1. Logotipo de DialogFlow*

DialogFlow, propiedad de Google, es otra herramienta popular para la creación de chatbots y asistentes virtuales. Ofrece una interfaz visual amigable para la creación de flujos de conversación y una gran cantidad de integraciones con otras herramientas y plataformas. DialogFlow también cuenta con la capacidad de integrar soluciones de inteligencia artificial y aprendizaje automático de Google, como Google Cloud Speech-to-Text y Google Cloud Translate [8-9].

### Amazon Lex



*Figura 2. Logotipo de Amazon Lex*

Amazon Lex es un servicio de AWS que se utiliza para construir asistentes conversacionales que se pueden usar mediante texto y voz (hasta 15 segundos de entrada de voz). Tiene mucha importancia en el mercado, ya que de él se nutre el asistente de Amazon llamado Alexa, presente en todos sus dispositivos echo. Otra de las características es que, como nos tiene acostumbrados Amazon, se permite la integración con otros servicios de Amazon de forma muy sencilla, pero nos dificulta realizar esta integración con productos externos. Pese a ello, cuenta con bastantes entornos de programación (Node.js, Python, .NET, Java, Javascript, PHP y Ruby) [10-11]

### IBM Watson



*Figura 3. Logotipo de IBM Watson*

Watson Assistant es el framework de creación de asistentes desarrollado por IBM hace poco menos de 2 años. Actualmente se puede encontrar dentro del IBM Cloud. Una de las claves del éxito de este asistente es que brinda a los desarrolladores la posibilidad de aislar la información que obtiene su asistente en una nube privada. De esta manera se pueden proteger los conocimientos resultantes de la interacción del usuario con el asistente. Los usuarios pueden interactuar con la aplicación del desarrollador a través de una variedad predefinida de canales de texto y voz. También se les ofrece la posibilidad de personalizar la interfaz para que no parezca que el asistente trabaja con la plataforma de Inteligencia Artificial Watson [12-13].

### RASA Framework



*Figura 4. Logotipo de RASA*

Rasa es una empresa de software, con central en Berlín que ofrece soluciones para crear chatbots de forma Open-Source y en Python. Al ser Open-Source proporciona muchas ventajas respecto a otras alternativas, ya que los datos del usuario que usa el asistente no pasan por terceros, y al disponer del framework localmente instalado, aunque el software deje de mantenerse, los chatbots pueden funcionar sin problemas El framework de Rasa consta de dos partes:

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

• RASA Core: Esta parte se encarga de la gestión del diálogo, se conecta con RASA NLU. El componente decide qué acciones tomar en cada momento [14-15].

### Conclusiones de las herramientas y marcos de trabajo analizados

Luego de realizar un análisis exhaustivo a estas tecnologías, se puede concluir que para el caso de uso planteado. Rasa al ser un framework basado en el lenguaje Python y permitir su configuración y modificación a través de este lenguaje da la facilidad de poderse integrar con las otras tecnologías que se utilizaran las cuales son Whisper para el Reconocimiento Automático del Habla y FastAPI para el backend de la aplicación en cuestión. Otro de los motivos de la elección de este framework se encuentra en el hecho de que tiene una basta y rica documentación, una comunidad excelente con un foro con constantes interacciones y esta herramienta se actualiza constantemente. Por último, este marco de trabajo ofrece mucha facilidad y tiene muchos tutoriales de cómo implementar modelos de machine Learning para usarlos como modelo de conocimiento del chatbot o como otro pipeline para el procesamiento del lenguaje natural. Además, Rasa cuenta con una interfaz de usuario intuitiva que permite la creación de conversaciones y flujos de diálogo de manera sencilla y eficiente.

En cuanto a Whisper, esta tecnología de reconocimiento de voz es una opción muy interesante para utilizar en conjunto con Rasa, ya que permite la captura y transcripción de la voz en tiempo real, lo que se traduce en una experiencia de usuario más fluida y natural. Además, Whisper cuenta con un alto grado de precisión en el reconocimiento del habla, lo que garantiza la calidad de la respuesta del chatbot.

Por otro lado, FastAPI es una herramienta de desarrollo de API en Python que se caracteriza por ser rápida y fácil de usar. Su eficiencia y simplicidad la convierten en una excelente opción para el desarrollo del backend de la aplicación. Además, FastAPI ofrece una documentación clara y detallada, lo que facilita su integración con otras tecnologías, como Rasa y Whisper.

Concluyendo, la elección de Rasa como framework para el desarrollo del chatbot se debe a su flexibilidad, facilidad de integración con otras tecnologías, documentación completa y actualización constante. La combinación de Rasa con Whisper para el reconocimiento de voz y FastAPI para el backend garantiza un chatbot de alta calidad y una experiencia de usuario satisfactoria.

## Herramientas y Lenguajes de Programación

En la actualidad, el mundo de la tecnología y la informática se encuentra en constante evolución, y el desarrollo de software es una de las áreas que más ha crecido en los últimos años. Para poder llevar a cabo esta tarea, es necesario contar con herramientas y lenguajes de programación adecuados que permitan a los desarrolladores crear programas y aplicaciones de manera eficiente y efectiva. Las herramientas y lenguajes de programación son elementos fundamentales en el proceso de desarrollo de software, y en este epígrafe se hablará sobre los lenguajes y las herramientas que se utilizarán para el desarrollo del asistente.

### Python

Python es un lenguaje de programación interpretado, orientado a objetos de alto nivel y con semántica dinámica. Su sintaxis hace énfasis en la legibilidad del código, lo que facilita su depuración y, por tanto, favorece la productividad. Ofrece la potencia y la flexibilidad de los lenguajes compilados con una curva de aprendizaje suave. Aunque Python fue creado como lenguaje de programación de uso general, cuenta con una serie de librerías y entornos de desarrollo para cada una de las fases del proceso de Data Science. Esto, sumado a su potencia, su carácter open source y su facilidad de aprendizaje le ha llevado a tomar la delantera a otros lenguajes propios de la analítica de datos por medio de Machine Learning como pueden ser SAS (software comercial líder hasta el momento) y R (también open source, pero más propio de entornos académicos o de investigación). Fue creado por Guido van Rossum en 1991 y se ha convertido en uno de los lenguajes de programación más populares en la actualidad.

Algunas de las principales características de Python son su sintaxis clara y legible, su facilidad de aprendizaje y su amplia gama de bibliotecas y módulos disponibles para su uso. Además, es un lenguaje multiplataforma, lo que significa que se puede ejecutar en diferentes sistemas operativos. [16] [17]

La ventaja de utilizar Python para el desarrollo de software es que es muy flexible y se puede utilizar para una amplia gama de aplicaciones, desde la creación de aplicaciones web y móviles hasta el análisis de datos y la inteligencia artificial.

En cuanto a la inteligencia artificial, Python es una opción óptima debido a su capacidad para procesar grandes cantidades de datos y su amplia gama de bibliotecas y módulos específicos para la inteligencia artificial, como TensorFlow, Keras, PyTorch, entre otros. Además, Python es muy utilizado en el campo de la ciencia de datos, lo que lo convierte en una excelente opción para el procesamiento de datos y la creación de modelos de inteligencia artificial. Por último, el punto más importante a tener en cuenta es que es el lenguaje en el cual está basado el marco de trabajo que se va a utilizar [18].

Por lo anteriormente planteado, es la mejor opción para el desarrollo del asistente.

### Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft que permite la edición y depuración de código en varios lenguajes de programación. Es un software de código abierto que funciona en los sistemas operativos Windows, macOS y Linux.

Las ventajas de programar con Python en Visual Studio Code son varias. En primer lugar, ofrece una integración completa con el lenguaje Python, lo que permite acceder a todas sus características y librerías de una manera más fácil y rápida. Además, cuenta con una gran cantidad de extensiones y complementos que permiten personalizar y mejorar la experiencia de programación. También ofrece herramientas de depuración y pruebas integradas, lo que facilita la identificación y corrección de errores en el código. [19-21]

### RASA

RASA es un marco de trabajo de código abierto y de inteligencia artificial asincrónico que se utiliza principalmente para la creación de asistentes contextuales, es decir, aplicaciones de bot de chat. El Agente de Python es compatible con el módulo SDK de Rasa, además de los módulos NLU y Core para trabajar con acciones en las aplicaciones de bot de chat.

### Whisper

De acuerdo a la información proporcionada en la página oficial del proyecto, Whisper es un sistema de reconocimiento automático de voz (ASR) que ha sido entrenado con más de 680,000 horas de datos supervisados en varios idiomas y tareas, obtenidos de la web. Además de transcribir el audio a texto, este programa también es capaz de traducirlo al inglés en tiempo real, con resultados impresionantes hasta ahora. La herramienta es altamente precisa, capaz de reconocer la puntuación y comprender incluso voces rápidas y complejas.

Para esto, se utiliza un modelo llamado "Transformador" que se entrena en varias tareas, como reconocimiento de voz y traducción. El modelo puede hacer muchas cosas diferentes y reemplaza la necesidad de usar diferentes programas para cada tarea. Para enseñar al modelo, se usan "tokens" especiales que ayudan a identificar qué tarea se está realizando [22-23]

Mongo DB

MongoDB es una base de datos NoSQL, orientada a documentos y de código abierto. Es una base de datos muy popular en el mundo de los chatbots debido a sus características principales, como la escalabilidad horizontal, la alta disponibilidad, la flexibilidad en el esquema de datos y el soporte para consultas complejas.

La escalabilidad horizontal de MongoDB permite que se puedan agregar más servidores a la base de datos para manejar grandes volúmenes de datos y tráfico de usuarios, lo que es muy importante para los chatbots que pueden tener un gran número de usuarios. Además, MongoDB ofrece replicación automática y particionamiento de datos, lo que garantiza una alta disponibilidad y rendimiento de la base de datos.

MongoDB también es muy flexible en cuanto al esquema de datos, lo que significa que no es necesario definir una estructura fija de la base de datos antes de comenzar a almacenar datos. Esto permite una mayor adaptabilidad a medida que el chatbot evoluciona y se agregan nuevas funcionalidades. Además, MongoDB soporta consultas complejas y puede manejar grandes volúmenes de datos en tiempo real, lo que es crucial para los chatbots que necesitan responder rápidamente a las solicitudes de los usuarios.

En resumen, MongoDB es una base de datos NoSQL muy popular en el mundo de los chatbots debido a su escalabilidad horizontal, alta disponibilidad, flexibilidad en el esquema de datos y soporte para consultas complejas [24].

FastAPI

Con pocos años de salida al mercado, FastAPI promete convertirse en el framework definitivo para la nueva generación de aplicaciones web que ya están naciendo.

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

Con un 14% de uso entre los desarrolladores web con Python, FastAPI es utilizado por Netflix, Uber y Microsoft.[25]

Argumentos a tomar en cuenta:

• Velocidad: FastAPI es uno de los marcos de Python más rápidos disponibles, lo que lo hace ideal para aplicaciones que requieren un rendimiento óptimo. Esto es especialmente importante para una API de chatbot, donde la velocidad de respuesta puede ser crítica.

• Facilidad de uso: FastAPI es muy fácil de usar y tiene una curva de aprendizaje baja. También incluye características útiles, como la generación automática de documentación y las pruebas integradas, que lo hacen más eficiente y conveniente para desarrollar y mantener.

• Compatibilidad con el marco Rasa: FastAPI es compatible con el marco Rasa, lo que lo hace una buena opción para su proyecto de API de chatbot. La integración de FastAPI con Rasa es simple y directa, lo que significa que puede comenzar a desarrollar su chatbot de manera rápida y eficiente.

• Comunidad: FastAPI es un marco en constante evolución con una gran comunidad de desarrolladores detrás de él. Esto significa que hay una gran cantidad de recursos y soluciones disponibles para ayudarlo a resolver cualquier problema o añadir nuevas características a su aplicación.

FastAPI es una excelente opción para la API del asistente con el marco RASA debido a su velocidad, facilidad de uso, compatibilidad y comunidad activa. Si lo que se busca es un marco de desarrollo de API robusto y eficiente, FastAPI es la mejor elección [26].

### Git

Git es un software de [control de versiones](https://es.wikipedia.org/wiki/Control_de_versiones) diseñado por [Linus Torvalds](https://es.wikipedia.org/wiki/Linus_Torvalds), pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de [código fuente](https://es.wikipedia.org/wiki/C%C3%B3digo_fuente). Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código [27-28].

Esto significa básicamente que Git es un rastreador de contenido. Así que Git puede ser utilizado para almacenar contenido — y se usa principalmente para almacenar código debido a otras características que proporciona.

Los proyectos de la vida real generalmente tienen múltiples desarrolladores trabajando en paralelo. Así que necesitan un sistema de control de versiones como Git para asegurarse de que no hay conflictos de código entre ellos.

Además, los requerimientos en este tipo de proyectos cambian constantemente. Así que un sistema de control de versiones permite a los desarrolladores revertir y regresar a una versión anterior de su código.

El sistema de ramas en Git permite a los desarrolladores trabajar individualmente en una tarea (Por ejemplo: una rama -> una tarea O una Rama -> un desarrollador). Básicamente, se puede pensar en Git como una aplicación de software pequeña que controla tu código base, si eres un desarrollador.

## Metodología de desarrollo de Software

### SCRUM

La metodología SCRUM es un enfoque ágil para el desarrollo de software que se basa en la iteración y la colaboración del equipo de trabajo. Esta metodología se enfoca en la entrega temprana y continua de software funcional, lo que permite la adaptación a los cambios en los requisitos del proyecto y la satisfacción de las necesidades del cliente.

SCRUM se compone de tres roles principales: el Product Owner, el Scrum Master y el Equipo de Desarrollo. El Product Owner es responsable de definir las funcionalidades que se deben desarrollar y priorizarlas en función de las necesidades del cliente. El Scrum Master es el facilitador del proceso de desarrollo, asegurando que el equipo de trabajo siga los principios y prácticas de SCRUM. Finalmente, el Equipo de Desarrollo es responsable de la implementación de las funcionalidades y la entrega del software.

La metodología SCRUM se basa en ciclos de trabajo cortos llamados "sprints", que duran de una a cuatro semanas. En cada sprint, el equipo de trabajo se enfoca en un conjunto de funcionalidades previamente definidas y se compromete a entregarlas al final del sprint. Al final de cada sprint, se realiza una revisión del trabajo realizado y se planifica el siguiente sprint.

Entre las ventajas de la metodología SCRUM se encuentran la flexibilidad para adaptarse a los cambios en los requisitos del proyecto, la entrega temprana y continua de software funcional, la mejora en la comunicación y colaboración del equipo de trabajo, y la reducción de los riesgos del proyecto.

En conclusión, la metodología SCRUM es un enfoque ágil para el desarrollo de software que se enfoca en la colaboración y la entrega temprana y continua de software funcional. Esta metodología se compone de tres roles principales y se basa en ciclos de trabajo cortos llamados "sprints". SCRUM ofrece ventajas en términos de adaptación a los cambios en los requisitos del proyecto, mejora en la comunicación y colaboración del equipo de trabajo, y reducción de los riesgos del proyecto [30-32].

Conclusiones del capítulo

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

# ORGANIZACIÓN Y DISEÑO

En este capítulo, se abordará el proceso de planificación y diseño del asistente, en el cual se han utilizado diferentes técnicas y herramientas para lograr un modelo claro y preciso de la aplicación. Se presentarán los diagramas que se han utilizado para el modelado de la aplicación, 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 asistente. El objetivo de este capítulo es proporcionar una visión general y detallada del proceso de planificación y diseño del asistente y servir como guía para el desarrollo de futuros proyectos de este tipo.

## Sprint Backlog

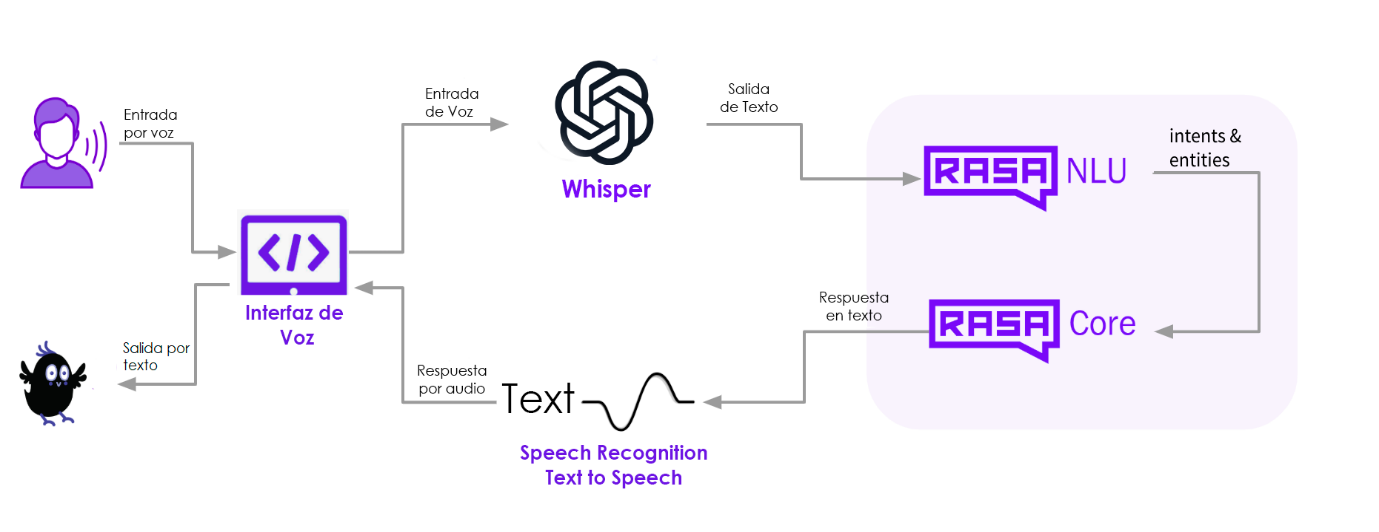
El sprint backlog es una lista de elementos de trabajo priorizados que el equipo de desarrollo debe completar durante un sprint en el marco de la metodología Scrum. Es una herramienta importante para planificar y realizar un seguimiento del progreso del equipo durante el sprint. La duración de cada uno de los siguientes sprints es de 2 semanas.



*Figura 5. Sprint Backlog*

## Arquitectura del Sistema

En el siguiente diagrama, se muestra una vista física de la arquitectura de Rasa y en particular el asistente. En él, se pueden apreciar los diferentes módulos del asistente (Interfaz de voz, Whisper, RASA\_NLU, RASA\_CORE y Text to Speech la cual hace la conversión de texto a voz de la respuesta dada por el bot al usuario).



*Figura 6. Diagrama de Arquitectura del Asistente*

## Requisitos no Funcionales

### Requisitos de Software

1. Sistema Operativo: La aplicación es compatible con los siguientes sistemas operativos:

* Microsoft Windows 10 o 11.

1. Versión de Python: La aplicación requiere Python en su versión 3.8 o superior.
2. Base de Datos: La aplicación utiliza MongoDB versión 5.0.6 como sistema de gestión de base de datos.
3. Dependencias: Todas las dependencias necesarias deben ser instaladas utilizando el gestor de paquetes PIP.

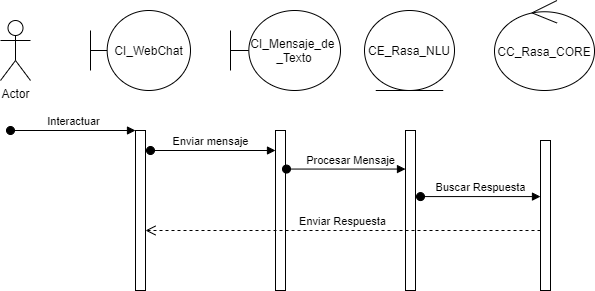
### Requisitos de Hardware

Los requerimientos mínimos de hardware para correr la aplicación:

1. Procesador: Se recomienda un procesador Intel Core i3 de 7ma generación o superior.
2. Memoria RAM: Se requieren al menos 8.0 GB de memoria RAM.
3. Almacenamiento: Debido al caché que almacena Whisper, se necesita un espacio de almacenamiento suficiente para garantizar el funcionamiento adecuado de la aplicación. El espacio exacto necesario dependerá del tamaño de la caché, que debe ser determinado durante el diseño de la aplicación.

## Diagrama de Secuencia

El diagrama de secuencia nos muestra la ruta que sigue el chatbot para poder generar una respuesta, se observa que primeramente el usuario interactúa con la interfaz web y envía un mensaje de texto el cual llega RASA\_NLU componente que se encargará de tomar el texto, procesarlo, analizarlo y descomponerlo de tal manera que el bot comprenda el contenido del mensaje, luego busque si existe alguna respuesta en RASA\_CORE y la envíe al usuario.



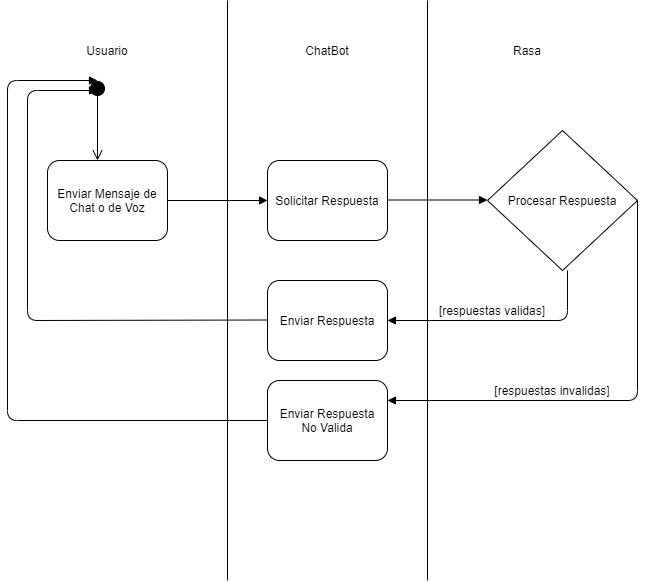
*Figura 7. Diagrama de Secuencia del Asistente*

## Diagrama de Actividades del Asistente

El diagrama de actividades representa el flujo de interacción entre el usuario y el chatbot. El proceso comienza cuando el usuario envía un mensaje de chat o voz al chatbot. Luego, el chatbot solicita una respuesta al usuario, lo que indica que está listo para recibir una entrada.

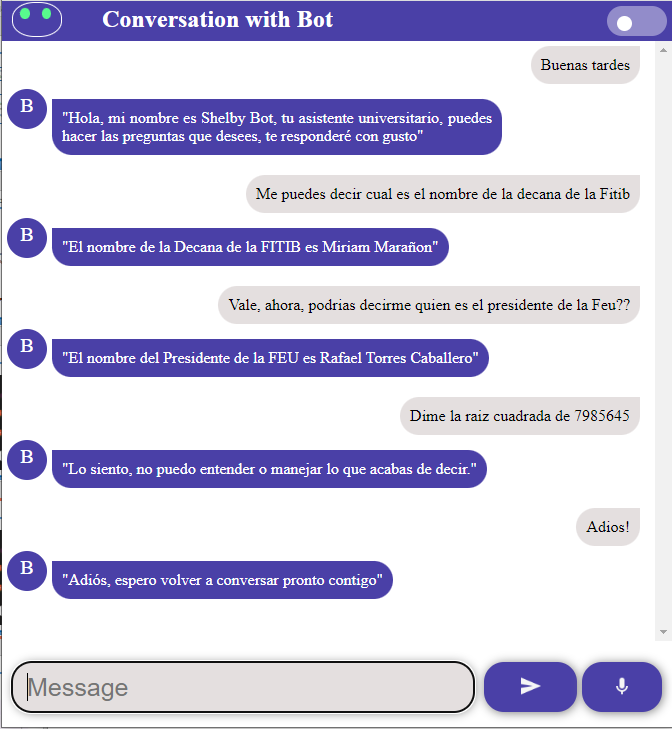
El chatbot procesa la respuesta del usuario utilizando el marco de desarrollo de software RASA NLU, que es una herramienta de procesamiento del lenguaje natural que ayuda a comprender y extraer información relevante de la entrada del usuario. Si RASA NLU encuentra una respuesta válida, el chatbot la envía al usuario como respuesta.

Sin embargo, si RASA NLU no puede encontrar una respuesta válida, el chatbot envía un mensaje al usuario indicando que su respuesta no es válida. Esto puede deberse a que la entrada del usuario no se ajusta a los patrones o intenciones que el chatbot reconoce.



*Figura 8. Diagrama de Actividades del Asistente*

## Diseño de la interfaz



*Figura 9. Interfaz del asistente*

IMPLEMENTACIÓN Y PRUEBA

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

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

## Instalación de las herramientas empleadas

### Python 3.8

1. Visita el sitio web oficial de Python en https://www.python.org/downloads/ y haz clic en "Download Python 3.8.x".

2. Selecciona el sistema operativo que estás utilizando (Windows, MacOS o Linux) y descarga el archivo de instalación correspondiente.

3. Una vez que se haya completado la descarga, abre el archivo de instalación.

4. Sigue las instrucciones del instalador para seleccionar las opciones de instalación que desees.

5. Asegúrate de agregar Python a la variable de entorno PATH, lo que permitirá que Python se ejecute desde la línea de comandos. En Windows, selecciona la opción "Add Python 3.8 to PATH" durante la instalación.

6. Haz clic en "Install Now" para comenzar la instalación.

7. Espera a que se complete la instalación.

8. Verifica que la instalación se haya completado correctamente abriendo la línea de comandos y escribiendo "python --version". Si se muestra la versión de Python 3.8.x, la instalación se ha completado correctamente.

### Instalación de Rasa

Vía Gratuita (Por Datos Móviles):

1. Tipear en la consola de comandos:

pip install rasa --index-url http://nexus.prod.uci.cu/repository/pypi-proxy/simple/ --trusted-host nexus.prod.uci.cu

### Instalación de Whisper

1. Instalación de Chocolatey

Abrir la PowerShell de Windows como administrador y escribir el comando:

Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))

1. Instalación de ffmpeg

Ejecutar el comando

choco install ffmpeg

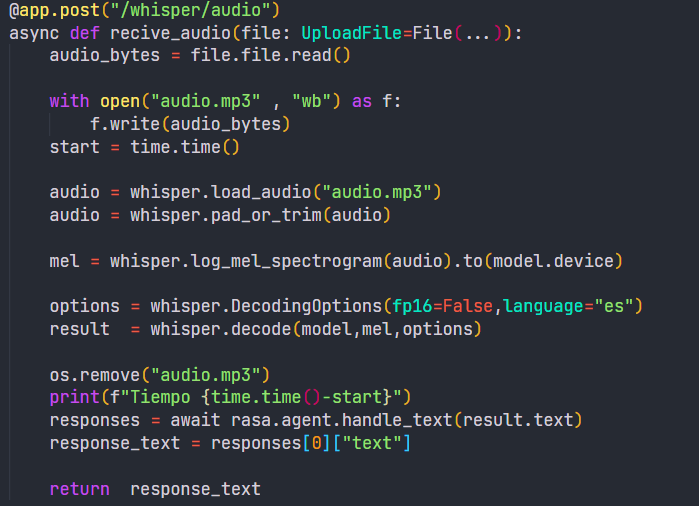
1. Por último, instalar las dependencias de Whisper

pip install torch --index-url http://nexus.prod.uci.cu/repository/pypi-proxy/simple/ --trusted-host nexus.prod.uci.cu

pip install pip install git+https://github.com/openai/whisper.git

## Algoritmos Importantes

### Algoritmo de Reconocimiento del Habla y Transcripción



*Figura 10. Algoritmo de Reconocimiento del Habla y Transcripción*

Este algoritmo es un endpoint de una API construida con FastAPI que recibe un archivo de audio en formato mp3 y devuelve el texto transcritpito de dicho audio en español.

Primero, el archivo de audio se lee y se guarda en un archivo temporal "audio.mp3". A continuación, se utiliza la librería "whisper" para cargar el audio, ajustar su duración y calcular el espectrograma de mel logarítmico.

Luego, se utiliza el modelo de reconocimiento de voz pre-entrenado para transcribir el audio en texto mediante el método "decode" de la librería "Whisper". Los resultados de la transcripción se almacenan en una variable "result".

Finalmente, se utiliza Rasa, una librería de procesamiento de lenguaje natural, para procesar el texto transcritpito y devolver una respuesta coherente. El tiempo que tarda el algoritmo en ejecutarse se imprime en la consola y se devuelve el texto transcritpito como respuesta a la solicitud del usuario.

CONCLUSIONES Y RECOMENDACIONES

**Conclusiones**

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

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

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

**Recomendaciones**

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

Recomendación 1

Recomendación 2

Recomendación 3

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

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

REFERENCIAS BIBLIOGRÁFICAS

[1] G. de Oliveira, R. Venson, y R. Marcelino, «REDES NEURAIS APLICADAS NO DESENVOLVIMENTO DE CHATBOTS: UMA ANÁLISE BIBLIOMÉTRICA», vol. 01, pp. 1-15, dic. 2017.

[2] Mindbowser, «5 learnings from our ‘Chatbot Survey — 2017’», Medium, feb. 09, 2017. https://chatbotsjournal.com/5-learnings-from-our-chatbot-survey-2017-72a6a4fc209c (accedido feb. 03, 2023).

[3] R. Caballero Ramírez y R. C. Ramírez, «Chatbot: una propuesta viable para la atención al cliente en el centro de soporte de la UCI.», Rev. Cuba. Cienc. Inform., vol. 15, n.o 0, oct. 2021,[Enlínea].Disponible en:

[https://rcci.uci.cu/?journal=rcci&page=article&op=view&path[]=2289](https://rcci.uci.cu/?journal=rcci&page=article&op=view&path%5b%5d=2289)

[4] B. Xu, «Domain-Specific Bots vs. Generic Bots», *DealerAI*, jun. 09, 2020. https://dealerai.com/domain-specific-bots-vs-generic-bots/ (accedido feb. 27, 2023).

[5] J. L. Beveridge and B. K. Boguraev, "Natural Language Processing," Handbook of Computational Linguistics and Natural Language Processing, John Wiley & Sons, Ltd, Chichester, UK, 2015, pp. 1-26.

[6] V. K. Chaudhri, A. Cheyer, R. Guili, B. Jarrold, K. L. Myers, y J. Niekarsz, «A case study in engineering a knowledge base for an intelligent personal assistant», en *Proceedings of the 5th International Conference on Semantic Desktop and Social Semantic Collaboration - Volume 202*, Aachen, DEU, nov. 2006, pp. 25-32.

[7] D. C. Sobrino, «¿Cómo funciona el reconocimiento automático del habla? », SoldAI, may 15, 2020. https://medium.com/soldai/c%C3%B3mo-funciona-el-reconocimiento-autom%C3%A1tico-del-habla-eb038ecfe72e (accedido feb. 27, 2023).

[8] «¿Qué es y para qué funciona la herramienta de Google Dialogflow?», Devoteam G Cloud. https://gcloud.devoteam.com/es/blog/que-es-dialogflow-asistente-virtual-google/ (accedido 1 de marzo de 2023).

[9] «Dialogflow», *GitHub*. https://github.com/dialogflow (accedido 1 de marzo de 2023).

[10] «¿Qué es Amazon Lex? - Amazon Lex».

https://docs.aws.amazon.com/es\_es/lex/latest/dg/what-is.html (accedido 1 de marzo de 2023).

[11] «Compare Amazon Lex vs Rasa». https://www.peerspot.com/product\_comparisons/33881-33901 (accedido 2 de marzo de 2023).

[12] R. de Juana, «IBM Watson: casi todo lo que tienes que saber», *MuyComputerPRO*, 24 de septiembre de 2019. https://www.muycomputerpro.com/2019/09/24/ibm-watson-casi-todo-lo-que-tienes-que-saber (accedido 1 de marzo de 2023).

[13] «Which-Chatbot-Platforms-are-Easiest-to-Build-and-Train-Summa-Linguae Technologies.pdf»

[14] «Conversational AI Platform | Superior Customer Experiences Start Here», *Rasa*, 1 de diciembre de 2020. https://rasa.com/ (accedido 2 de marzo de 2023).

[15] «Comparativa de nueve frameworks de chatbot - Planeta Chatbot», 28 de abril de 2021. https://planetachatbot.com/comparativa-frameworks-chatbot/ (accedido 2 de marzo de 2023).

[16] «¿Qué es Python?» https://web.archive.org/web/20200224120525/https://luca-d3.com/es/data-speaks/diccionario-tecnologico/python-lenguaje (accedido 3 de marzo de 2023).

[17] «Applications for Python», *Codelivly*, 15 de diciembre de 2022. https://www.codelivly.com/applications-for-python/ (accedido 3 de marzo de 2023).

[18] «Python (in Machine Learning)», *Corporate Finance Institute*. https://corporatefinanceinstitute.com/resources/data-science/python-in-machine-learning/ (accedido 3 de marzo de 2023).

[19] «Get Started Tutorial for Python in Visual Studio Code». https://code.visualstudio.com/docs/python/python-tutorial (accedido 3 de marzo de 2023).

[20] «PEP 484 – Type Hints | peps.python.org». https://peps.python.org/pep-0484/#id18 (accedido 3 de marzo de 2023).

[21] «Visual Studio Code - Code Editing. Redefined». https://code.visualstudio.com/ (accedido 3 de marzo de 2023).

[22] *Whisper*. OpenAI, 2023. Accedido: 5 de marzo de 2023. [En línea]. Disponible en: https://github.com/openai/whisper

[23] E. S. Zawadzki, «Whisper, la herramienta de inteligencia artificial de OpenAI para convertir archivos de audio a texto», *Entrepreneur*, 10 de febrero de 2023. https://www.entrepreneur.com/es/tecnologia/whisper-la-herramienta-de-inteligencia-artificial-de/444821 (accedido 5 de marzo de 2023).

[24] "MongoDB - The Leading Modern, General Purpose Database", MongoDB, 2021. [Online]. Available: https://www.mongodb.com/.

[25] «¿Qué sigue después de aprender Python? Django vs. Flask vs. FastAPI», Platzi. https://platzi.com/blog/django-flask-fastapi/ (accedido feb. 01, 2023).

[26] S. Ramírez, *tiangolo/fastapi*. 2023. Accedido: 3 de marzo de 2023. [En línea]. Disponible en: <https://github.com/tiangolo/fastapi>

[27] «Git». https://git-scm.com/ (accedido 3 de marzo de 2023).

[28] «La guía para principiantes de Git y Github», *freeCodeCamp.org*, 9 de enero de 2021. https://www.freecodecamp.org/espanol/news/guia-para-principiantes-de-git-y-github/ (accedido 3 de marzo de 2023).

[29] mijacobs, «¿Qué es Git? - Azure DevOps», 9 de febrero de 2023. https://learn.microsoft.com/es-es/devops/develop/git/what-is-git (accedido 3 de marzo de 2023).

[30] Schwaber, K., & Beedle, M. (2002). Agile software development with scrum. Prentice Hall Professional.

[31] Sutherland, J., & Schwaber, K. (2011). The scrum guide. Scrum.org.

[32] Cohn, M. (2010). Succeeding with agile: software development using scrum. Addison-Wesley Professional.

Este capítulo no se enumera. Incluyen los materiales que fueron utilizados durante el desarrollo del trabajo.

Deberá usarse el estilo de citación de la IEEE (orden numérico. ubicando Al citar usando este estilo, las referencias deben ser ordenadas en el orden que fueron citadas.

Recuerde que utilizando un gestor bibliográfico como el que viene embebido en Microsoft Word o en *Endnote*, el ordenamiento de las referencias se hará automáticamente. Debe utilizar las normas Harvard.

Las referencias se señalan cuando se hace alusión directa o indirecta a un autor o a su obra, sobre todo si se le cita textualmente. Recuerde que la utilización de palabras o conceptos de otro autor sin señalar la fuente constituye un delito (PLAGIO) sancionado. Denota además cierta falta de ética.

|  |  |
| --- | --- |
| [1] | R. H. Sampieri, C. F. Collado y P. B. Lucio, Metodología de la Investigación, Istapalapa: Mc Graw Hill, 2006.  Autor(es) (Año). Título del Artículo, Título de la Revista, Número del Volumen, (Número de Edición): Páginas.  Autor(es) (Año). Título del libro en itálica. Número de la edición, Editorial, Lugar de Publicación.  Autor(es) (Año). Título del Artículo o del Capítulo en el Libro, En: Nombre del Libro, Número de la Edición, Número del Capítulo, Editores, Editorial, Lugar de la Publicación. |

Se sugiere utilizar los gestores bibliográficos[[1]](#footnote-2) para la gestión de las referencias bibliográficas, de esta forma ellos harán todo el proceso de organización y formateo por el autor.. El uso del gestor posibilita la exportación de un archivo (base de datos) con una lista detallada de las referencias utilizadas, la cual debe ser entregada al/los tutor/es. Esta base de datos es de utilidad para la continuidad de las investigaciones en la institución académica )

**BIBLIOGRAFIA (**opcional**)**

Es opcional, puede incluir aquí los libros que fueron leídos, pero que no fueron referidos en el trabajo.

GLOSARIO DE TÉRMINOS

Se utiliza cuando durante el desarrollo se han utilizado palabras, conceptos, siglas, etc., que no expresan una idea clara, precisa y concisa, o que puedan tener múltiples interpretaciones

Cuando aparece una sigla por primera vez, ella se escribe normalmente y entre paréntesis se pone el significado de sus términos en el idioma original y a su vez se pone en el Glosario de términos, cuando aparece nuevamente no es necesario poner entre paréntesis el significado de la sigla.

El glosario de términos se anexa después de la bibliografía y antes de los anexos y se escribe en orden alfabético, Se recomienda no abusar de los anglicismos y si se utilizan deben escribirse en itálicas

ANEXOS

Anexos: se colocan al final del trabajo, no se numeran como capítulos, pero sí se les coloca un número de orden luego de la palabra ANEXO; en ellos se incluye información que no es necesaria para la presentación del trabajo, pero sí útil al lector. Exigen que se hagan referencia de ellos en el contenido o cuerpo de la obra y deben aparecer en el mismo orden en que han sido citados y deben tener un pie de grabado o título que enuncie lo que ilustra el mismo.

Los anexos deben recoger aquellos aspectos del trabajo que, por su longitud, complejidad o que no era adecuado ponerlos, no se incluyen en el texto de la tesis pero que coadyuvan a una mejor comprensión de lo que se expone en ella. Sin embargo, es necesario dejar bien claro que los anexos no pueden usarse para extender la tesis más allá del límite establecido.

Es opcional su uso

**Anexo I** Inserte título del primer anexo

Inserte aquí el primer anexo.

**Anexo II** Inserte título del segundo anexo

Puede añadir tantos anexos como le sean necesarios.

1. Gestor embebido en Microsoft Word 2007 en adelante, EndNote, Zotero, etc. [↑](#footnote-ref-2)