



Universidad de Oriente

Sede "Julio Antonio Mella"

Facultad de Ingeniería en Telecomunicaciones, Informática y
Biomédica

Trabajo de Diploma

En opción al título de Ingeniero en Informática

Título: "Herramienta digital para la construcción de conocimiento automático para un Asistente Virtual".

Autor: Jorge Ernesto Duvalón Hernández.

Tutores: MSc. Dionis López Ramos.

**Proyecto de Investigación "Soluciones informática para la
gestión de los procesos universitarios a partir de plataformas
de gobierno electrónico"**

Santiago de Cuba, 2022

"Año 64 de la Revolución"

Resumen

La población requiere respuestas inmediatas y acciones en tiempo real de diferentes servicios institucionales (Ej.: Salud, Legalidad, Seguridad, entre otros). En los momentos actuales de desarrollo tecnológico y científico, los canales tradicionales de gestión no pueden satisfacer la demanda pico de búsqueda de información por parte de la población. Para resolver esta necesidad han sido creados los asistentes virtuales (Agentes Conversacionales) o robots conversacionales. Los asistentes virtuales son programas que intenta imitar la conversación que puede proveer un ser humano, además de concebirse como herramientas digitales que permiten la interacción hombre máquina. Los mismos son ampliamente utilizados en el sector empresarial, salud y gobierno porque garantizan una atención al usuario las 24 horas.

A pesar de los grandes beneficios que proporcionan los asistentes virtuales, la creación del conocimiento que usan para dar respuestas a las preguntas y la interacción con los usuarios es laboriosa y costosa. Esto es debido a la necesidad de reunir especialistas y aglutinar la información necesaria para estos asistentes virtuales, además de que este arduo proceso puede dificultar la creación de los asistentes virtuales.

En esta investigación se propone el diseño e implementación de una herramienta para la creación, entrenamiento y despliegue de asistentes virtuales, reduciendo la necesidad de la interacción con especialistas. Para la creación de esta herramienta y el despliegue de los asistentes virtuales se emplea el lenguaje de programación Python y el marco de trabajo Rasa especializado en la creación de asistentes virtuales.

Abstract

Title: "Digital tool for automatic building of a Virtual Assistants knowledge".

The population requires immediate responses and real-time actions from different institutional services (eg, Health, Legality, Security, among others). In the current moments of technological and scientific development, the traditional management channels cannot satisfy the peak demand and the search for information. To solve this need, virtual assistants (Conversational Agents) or conversational robots have been created. Virtual assistants are programs that try to imitate the conversation that a human being can provide, in addition to being conceived as digital tools that allow human-machine interaction. They are widely used in the business, health and government sectors because they guarantee 24-hour customer service.

Despite the great benefits virtual assistants provide, creating the knowledge they use to answer questions and interact with users is time-consuming and expensive. This is due to the need to gather specialists and bring together the necessary information for these virtual assistants, in addition to the fact that this arduous process can make it difficult to create virtual assistants.

This research proposes the design and implementation of a tool for the creation, training and deployment of virtual assistants, reducing the need for interaction with specialists. For the creation of this tool and the deployment of virtual assistants, the Python programming language and the Rasa framework specialized in the creation of virtual assistants are used.

Índice

Capítulo 1. Marco Referencial	9
1.1 Las plataformas para desarrollar robots conversacionales	9
1.2 Breve historia de los agentes conversacionales	9
1.2.1 Estado del arte de los marcos para desarrollar robot conversacional	10
Azure Bot Service	10
DialogFlow	11
BotPress	13
Plato Research Dialogue System	14
Conclusión sobre las plataformas para la creación de Asistentes Virtuales	16
Python	17
PyCharm Community	17
RASA	18
MongoDB	18
YAML	18
GIT	19
1.4 Metodología de desarrollo de software	19
Conclusiones del capítulo	20
Capítulo 2. Planificación y Diseño	21
2.1 Propuesta del Sistema	21
2.2 Usuarios del Sistema	22
2.3 Planificación del desarrollo del Sistema	22
2.3.1. Requisitos Funcionales	22
2.3.2. Requisitos no Funcionales	23
2.3.3. Historias técnicas	24
2.3.4. Historias de usuario	24

2.3.6. Arquitectura del Sistema	26
Conclusiones del capítulo	28
3.1. Instalación de las herramientas utilizadas	29
3.2 Implementación de Funcionalidades	30
3.3 Análisis económico	33
3.3.1. Estimación de costo y tiempo.....	33
3.4 Pruebas al sistema.....	37
Conclusiones del capítulo	40
Conclusiones	41
Recomendaciones	41
Anexos	44
Instalación	44
Iniciando el sistema	44
Opción 1. Crear Asistente Virtual	45
Opción 2. Generar Conocimiento	46
Algoritmos implementados para idioma español.....	46
Opción 3. Entrenar Asistente Virtual	51
Opción 4. Probar Asistente Virtual.....	53

Introducción

Un Asistente virtual (AV) es un programa informático que permite a los seres humanos interactuar con la tecnología utilizando una variedad de métodos de entrada (voz, texto, gestos, tacto, etc.) y que suele estar disponible las 24 horas, los 7 días a la semana y los 365 días del año. [1]

Durante muchos años, los asistentes virtuales se utilizaron sólo en entornos de servicio al cliente, pero ahora se han añadido otros casos de uso principalmente dentro de las empresas para mejorar la experiencia del cliente y la eficiencia empresarial. Los asistentes virtuales son cada vez más populares y se conocen por una variedad de nombres diferentes: *robot conversacional de inteligencia artificial*, *asistente virtual inteligente*, *asistente digital*, entre otros. [1]

Así como los robots conversacionales se conocen por una variedad de nombres diferentes, también suelen tener diferentes grados de inteligencia. Un agente conversacional básico es apenas un poco más avanzado que una solución de interfaz gráfica interactiva (front-end) para responder a preguntas frecuentes (FAQs).

Luego están los agentes conversacionales contruidos en algunos de los marcos para desarrollar asistentes virtuales disponibles en el mercado. Estos pueden ofrecer características más avanzadas como el de recogida de datos u otras capacidades transaccionales simples, como por ejemplo tomar un pedido para una pizza. Sin embargo, sólo los asistentes virtuales con inteligencia artificial tienen la capacidad de ofrecer una experiencia conversacional sofisticada que la mayoría desea incorporar a su trabajo. [1]

Una atención al cliente pronta, particular y eficaz, servicio de venta, asistencia y acompañamiento puntual, así como ser una valiosa fuente de datos e información que nos pueden dar un panorama vital del negocio son otras de las ventajas que brinda el uso de los asistentes virtuales. Además, también pueden identificar de manera eficiente el comportamiento y las necesidades de los clientes a partir de un conocimiento previamente dado e incluso llegar a entrenarse a partir de las conversaciones, pueden recopilar datos importantes para una mejor experiencia con el usuario. Con estos datos, los sistemas cognitivos que utilizan Inteligencia Artificial (IA) son capaces de depurar, ordenar, analizar y hacer referencias cruzadas de información, poniéndola disponible para diferentes usos: optimización de sitios web corporativos y redes sociales; redacción de mensajes, personalizados o destinados a un sector específico; llegando a mejorar el propio robot conversacional. [2]

En nuestro país actualmente esta tecnología no es muy explotada, aunque han existido, como es el ejemplo de Amanda (Usada para información sobre las elecciones en Cuba) [3]; ELIZ que es un asistente virtual para la plataforma ENZONA la cual brinda información sobre los servicios de la aplicación; La revista Alma Mater tiene un chatbot donde se puede recibir el boletín de

publicaciones semanales, aclara dudas o medidas sobre la Covid-19 y el estado de las alertas ciclónicas en el país, enviar sugerencias u opiniones [4]. El centro de soporte de la Universidad de Ciencias Informáticas (UCI) posee un bot llamado “C.S. Chatbot” para la atención al cliente, donde da respuestas a diferentes cuestiones como por ejemplo las incidencias reportadas por los usuarios [5].

Las personas necesitan satisfacer dudas sobre cualquier tema constantemente y las entidades han buscado formas para ello con el uso de personal especializado que debe estar disponibles las 24 horas. Muchas veces este personal no puede atenderlos por horario o problemas con el servicio el cual genera altos costos. Las personas muchas veces tienen que moverse grandes distancias para acceder a la información que necesitan.

Los robots conversacionales pueden resolver las necesidades antes expuestas, dar las ventajas que se han mencionado mejorando así las formas de trabajo con el cliente; pero necesitan el conocimiento necesario para ello.

Un asistente virtual requiere de una base de conocimientos para poder generar las posibles respuestas a los clientes. Esta base de conocimiento puede crearse a partir del conocimiento de especialistas, que crean o reúnen el posible conocimiento que pueda servir para dar respuestas a las posibles preguntas o conversaciones de los usuarios. Otra opción es la indexación de grandes volúmenes de información que permita la inferencia de conocimiento a partir de minar la información almacenada en grandes fuentes de datos, como los que tienen empresas transnacionales como Amazon, Google y otras. Esta opción resulta muy costosa para países en desarrollo o empresas pequeñas o gobiernos locales.

Problema de la investigación:

La creación del conocimiento para poder contar con una Base de Conocimiento que sirva a los asistentes virtuales es un reto que puede resultar en costos en tiempo y recursos. El objetivo de esta investigación es, además de facilitar el desarrollo de asistentes virtuales de tal manera que resulte en minimizar costos, recursos y trabajo manual, es potenciar el uso de los mismos mejorando sus capacidades de respuesta.

Objeto de Estudio: La gestión automatizada del conocimiento y el procesamiento del lenguaje natural.

Campo de Estudio: Los Asistentes Virtuales.

Objetivo General: Desarrollar una aplicación informática que permita la gestión o construcción de conocimiento de manera automática para Asistentes Virtuales, para que estos puedan ser más eficientes al responder cualquier duda o inquietud de los usuarios.

Objetivo Específicos

- Estudio del estado del arte de las herramientas para la creación de asistentes virtuales.

- Diseñar un prototipo de herramienta para la creación de conocimiento de un asistente virtual.
- Implementar el prototipo de herramienta diseñado.
- Probar el prototipo de herramienta diseñado en varias esferas del conocimiento humano.
- Desplegar la herramienta junto a un sistema de gestión para la creación de asistentes virtuales.

Hipótesis:

Se desarrollará una aplicación informática que construya el conocimiento para el desarrollo de asistentes virtuales de forma automática, que permita a los asistentes virtuales responder o evacuar cualquier duda de los usuarios eficientemente.

Métodos de investigación:

- Método histórico-lógico: se aplicó al realizar el análisis de la existencia de otros sistemas que den solución al problema en cuestión.
- Método de análisis y síntesis: se aplicó al realizar el análisis de todo el proceso llevado a cabo en el proyecto y sintetizar las ideas que fueron surgiendo; extrayendo los elementos comunes al objeto de estudio.

Estructura del informe:

Capítulo 1: Muestra el marco de referencia, 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:

- Un sistema que a partir de datos proporcionados genere o construya automáticamente la estructura de conocimiento para asistentes virtuales mediante técnicas de procesamiento del lenguaje natural reduciendo en gran medida el error humano y el trabajo que cuesta construir la base conocimiento.
- Construcción automática de asistentes virtuales a partir de una base de conocimiento nueva previamente construida.
- Evidenciar las facilidades que brinda este tipo de solución en el desarrollo de asistentes virtuales sin necesidad de especialistas.
- Mejora de la capacidad de respuesta de los asistentes virtuales y a su vez la calidad de su servicio al interactuar con las personas.

Capítulo 1. Marco Referencial

En este capítulo se explican los principales aspectos teóricos, los conceptos básicos de las tecnologías y la caracterización de las herramientas computacionales utilizadas.

1.1 Las plataformas para desarrollar robots conversacionales

Una plataforma de robot o agente conversacional permite diseñar, desarrollar, implementar y mantener sistemas de conversación de una manera rápida, eficiente y uniforme. La creación de soluciones de robot conversacional con inteligencia artificial atractivas puede ser compleja y los marcos ayudan a simplificar el desarrollo de estas soluciones, gracias a diversas funcionalidades y herramientas ya incluidas. Las plataformas deben tener todo lo que un desarrollador necesita para construir un sistema conversacional; desde herramientas de minería de datos y diseño, hasta programas analíticos necesarios para mantener el sistema y proporcionar información valiosa a la empresa. [1]

1.2 Breve historia de los agentes conversacionales

Los asistentes virtuales se han convertido en un importante instrumento científico tecnológico, debido al crecimiento y al avance de la inteligencia artificial.

No es algo nuevo, se vienen usando desde hace décadas, pero no fue hasta que el uso de Internet se volvió más común, que estos comenzaron a ser utilizados para dar soporte a las funciones de servicio al cliente.

ELIZA, entre 1964 y 1966

En 1964, el informático del MIT Joseph Weizenbaum inició el desarrollo de ELIZA, que se convertiría en la primera máquina capaz de hablar utilizando el procesamiento del lenguaje natural.

Eliza fue nombrada en honor al personaje de Eliza Doolittle en la obra Pygmalion de George Bernard Shaw, y engañó a muchas personas haciéndoles creer que estaban hablando con un humano. Para lograrlo, simplemente añadía palabras de los usuarios a sus propios guiones y les respondía con estos para mantener la conversación. [1]

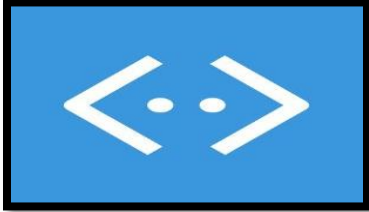
ALICE, 1995

A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) también conocido como Alicebot, o simplemente Alice, es un robot conversacional con procesamiento de lenguaje natural desarrollado por primera vez en 1995. Ganó el premio Loebner (Usa el formato de la Prueba de Turing estándar con jueces que deciden cuál de los programas participantes es el más parecido a un humano.) tres veces y fue inspirado por el programa ELIZA. [1]

Esta tecnología no se quedó estancada y siguió su avance llegando a tener varios asistentes virtuales potentes que hoy son usados mundialmente: **Siri** (creado por Apple), **Asistente de Google** (Google Assistant creado por Google), **Alexa** (creado por Amazon), **Cortana** (creado por Microsoft).

1.2.1 Estado del arte de los marcos para desarrollar robot conversacional

Azure Bot Service



Azure Bot Service es una herramienta que forma parte de los servicios en la nube de Azure que entra en la categoría Plataformas y herramientas de Robot conversacional de una pila tecnológica. [6-10] [21] [22][24]

Cuenta con la herramienta **Bot Framework Composer**:

Es un Entorno de Desarrollo Integrado (Integrated Development Environment (IDE)) de código abierto para que los desarrolladores creen, prueben, aprovisionen y administren experiencias conversacionales. Proporciona una interfaz visual que permite que los diálogos, los modelos de comprensión del lenguaje, las bases de conocimiento y las respuestas de generación de lenguaje se creen desde una plantilla, además permite que estas experiencias se extiendan con código para tareas más complejas, como la integración del sistema.

Más de 18 empresas usan la tecnología de Microsoft en sus sistemas de trabajo como por ejemplo LUMENEO, Autonom8 y WBOT.

Azure Bot Service proporciona plataformas para integrar sus agentes conversacionales: Slack, Skype, interfaz de programación de aplicaciones (siglas en inglés API) de Telegram, son algunas de las herramientas populares que se integran con Azure Bot Service.

Ventajas

- Usa técnicas de Inteligencia Artificial (IA) y procesamiento del lenguaje natural.
- Código Abierto y extensible (Bot Framework Composer).
- Soluciones de nivel empresarial.
- Propiedad y control.

Desventajas

- El SDK ("Software Development Kit" (Kit de desarrollo de software) reúne un grupo de herramientas que permiten la programación de aplicaciones informáticas) de su sistema tiene un nivel de complejidad alto, es avanzado, lo que requiere tiempo dominar su uso y más para personas que no cuentan con mucha experiencia usando esta plataforma o parecidas pertenecientes a Microsoft.
- Bot Framework Composer depende de los servicios en la nube de Azure.

Azure Bot Service

Gratis: Este es un nivel gratuito sin costo con un total de **10 000 mensajes/mes**, herramientas de creación de bots y canales estándar gratuitos.

Premium (Nivel donde se paga por tener funciones avanzadas que no están presentes en el nivel gratuito): Este nivel es una versión de pago de **0.50 USD por cada 1000 mensajes**, permiten que el robot conversacional se comuniquen con los usuarios dentro de su propia aplicación o en su aplicación web además de contener herramientas de creación de bots y canales estándar gratuitos y premium. Aparte de esto, también le cobran por los recursos consumidos en las funciones de Azure y la aplicación web de Azure.

DialogFlow



Dialogflow desarrollada por Google, es una plataforma con comprensión del lenguaje natural que facilita el diseño de una interfaz de usuario de conversación y su integración a tu aplicación para dispositivos móviles, aplicaciones web, dispositivos, bots, sistemas de respuesta de voz

interactiva, etc. Proporciona nuevas y atractivas formas para que los usuarios interactúen con tu producto. [11-15] [25] [21]

Dialogflow puede analizar múltiples tipos de entradas de tus clientes, incluidas entradas de texto o audio (como las de un teléfono o una grabación de voz). También puede responder a tus clientes de varias maneras, ya sea a través de texto o con voz sintética.

Integraciones de Dialogflow

Dialogflow proporciona a los desarrolladores una variedad de plataformas para integrar sus agentes conversacionales. Esto incluye Facebook Messenger, Skype, Slack, Twilio, Viber, Twitter, iPhone, Google Assistant y otros.

Ventajas de Dialogflow

- Agentes conversacionales integrados.
- Webhooks personalizados (Webhooks son retrollamadas HTTP de usuario. Cuando esto ocurre, la web envía una solicitud HTTP a la URL de destino configurada para el webhook.).
- Gran interfaz.
- Integraciones Out-Of-The-Box (OOTB), no requiere mucho desarrollo o personalización.
- Base de conocimientos.
- Multilingüe.
- Visualización rápida.

Desventajas de Dialogflow

- Si decide que mover una respuesta de seguimiento bajo una intención (*conjunto de expresiones dichas por el usuario que significan una acción concreta. Ej. Intención “saludar” con expresiones como “Hola” o “Que tal”*) diferente, no puede simplemente arrastrarla debajo de la intención deseada. En su lugar, deberá eliminar la intención existente, crear una nueva intención en una ubicación diferente y volver a escribir todas las frases de entrenamiento que ya haya creado, lo que desperdicia tiempo y resulta tedioso.
- Mucho trabajo manual, tiempo y capacitación: En muchos casos, Dialogflow hace que sea más complicado automatizar los procesos y ampliar el aprendizaje de su agente conversacional. Esto puede ser molesto porque tiene que ingresar muchos datos manualmente, especialmente cuando considera la necesidad de entrenar a su bot con el tiempo.

Precios de DialogFlow: La edición estándar es gratuita, en caso de ser usado para muchas peticiones lo enviará a la versión de pago que cobra \$0.002 por solicitud. Sin embargo, CX Agent Edition cobra \$20 por cada 100 sesiones de chat (Si en 5 meses hay 5000 sesiones de chat serían \$1000) y \$45 por cada 100 sesiones de voz.

Rasa



Rasa es una plataforma de código abierto para desarrollar asistentes virtuales. Las plataformas de código abierto son software con código fuente que cualquiera puede inspeccionar, modificar o mejorar. Al ser de código abierto, los desarrolladores podrán integrar características y funcionalidades adicionales según sus requisitos. La plataforma es fácil de personalizar y flexible, por lo tanto, se puede modificar según sus necesidades. [16-18] [26] [27]

Tiene un marco para la comprensión del lenguaje natural, la gestión del diálogo y las integraciones. Rasa X es un conjunto de herramientas gratuitas utilizadas para mejorar los asistentes contextuales creados con Rasa Open Source. Juntos, incluyen todas las características para crear excelentes robots conversacionales basados en texto y voz.

Hay varias compañías que usan Rasa y algunas de ellas son ERGO, Orange, Lemonade y T-Mobile.

Características

Integración en sistemas existentes. Al ser de código abierto se integra sin problemas aprovechando los beneficios de varios sistemas Back-end, APIs y Automatización Robótica de Procesos.

- Soporta varias intenciones únicas y múltiples para comprender lo que el usuario quiere, también admite entidades pre-entrenadas y

personalizadas para ayudar a modificar la intención según la solicitud del usuario.

- Aprendizaje interactivo mientras estableces una conversación con el asistente virtual, viendo así la experiencia práctica y recoger datos de los resultados que luego se revisan para mejorar la experiencia y la eficiencia del asistente.
- Integración con aplicaciones de mensajería como Facebook messenger, Google Home, Rocket, Slack, Telegram y otras.
- Inteligencia multilingüe.
- Estructura y nivel de desarrollo sólido, avanzado.
- Se basa en el lenguaje de programación Python.

Ventajas de RASA:

- Rasa tiene uno de los conjuntos de documentación y la comunidad de soporte en línea más completa. Esto es importante porque Rasa requiere una gran cantidad de conocimientos técnicos para su uso.
- Muchas opciones y posibilidades de personalización. Esto permite a los desarrolladores crear asistentes de texto y voz muy únicos impulsados por IA.
- Convierta el texto de forma libre en cualquier idioma en datos estructurados. Admite intenciones únicas y múltiples y entidades pre-entrenadas y personalizadas.

Desventajas de RASA:

- Requiere un nivel técnico considerable por su complejidad de desarrollo (comandos y trabajo con archivos) y su sofisticación.
- Compatibilidad entre su sistema y el de su Computadora Personal (PC), hay que observar o elegir bien la versión de sus programas más adecuado para tu PC, y como tiene múltiples versiones se dificulta a veces la compatibilidad.

BotPress



Botpress es una plataforma de código abierto para construir asistentes virtuales de forma fácil, accesible y rápida. [19-20] [21] [27]

La creación de robots conversacionales en esta plataforma es cómoda para los desarrolladores.

Tiene una unión de código repetitivo y la infraestructura que necesita para poner en marcha un robot conversacional. Es una plataforma completa para desarrolladores con todas las herramientas que necesita para construir, implementar y administrar robots conversacionales de nivel de producción en un tiempo récord.

Botpress cuenta con una amplia gama de empresas, incluidas agencias digitales, organizaciones Fortune 500 (lista de Compañías), gobiernos y nuevas empresas están construyendo asistentes digitales con Botpress Platform.

Características

- Tareas integradas de procesamiento de lenguaje natural, como reconocimiento de intenciones, revisión ortográfica, extracción de entidades, etiquetado y otras.
- Un estudio de conversación visual para diseñar conversaciones y flujos de trabajo de varios turnos.
- Un emulador y un depurador para simular conversaciones y depurar tu robot conversacional.
- Soporte para canales de mensajería populares como Slack, Telegram, MS Teams, Facebook Messenger y un chat web incrustable.
- Un SDK y un editor de código para ampliar las capacidades.

Ventajas de BotPress

- Se puede ejecutar en la plataforma en la nube de BotPress en internet o en la infraestructura local de su elección, lo que le brinda un control total sobre la privacidad de sus datos.
- El núcleo de la plataforma Botpress es de código abierto con miles de colaboradores de GitHub y observadores, y un próspero foro comunitario.
- La plataforma ofrece una gran experiencia de desarrollador al ser notablemente flexible, fácil de usar y rápida. Soporte para múltiples idiomas, incluyendo francés, árabe, español y más.
- Sin bloqueo de proveedores.

Desventajas de BotPress

La aplicación se actualiza con bastante frecuencia y la documentación no va actualizada con estos cambios.

- Los videos tutoriales de la plataforma no son muchos y no abarcan todo lo relacionado con la configuración al crear asistentes virtuales, además no se han actualizado a las últimas versiones que van saliendo de la plataforma por lo que el aprendizaje se hace con dificultad.
- Difícil ejecutar varias instancias desde una instalación.
- En un inicio es un tanto complicado a nivel técnico.

Plato Research Dialogue System



El Sistema de Diálogo de Investigación de Plato es un marco flexible y de código abierto que se puede utilizar para crear, entrenar y evaluar agentes de IA conversacionales en diversos entornos. Admite interacciones a través de actos de voz, texto o diálogo y cada agente conversacional puede interactuar con datos, usuarios humanos u otros agentes

conversacionales (en un entorno de múltiples agentes). Cada componente de cada agente se puede entrenar de forma independiente en línea o fuera de línea y Plato proporciona una forma fácil de envolver prácticamente cualquier modelo existente, siempre que se cumpla con la interfaz de Plato. [28]

Características

- Plato ha sido diseñado para ser lo más modular y flexible posible; es compatible con arquitecturas de IA conversacionales tradicionales y personalizadas y, lo que es más importante, permite interacciones de múltiples partes donde múltiples agentes, potencialmente con diferentes roles, pueden interactuar entre sí, entrenar simultáneamente y resolver problemas distribuidos.
- Internamente, cada componente de un agente conversacional puede ser cualquier cosa, desde un modelo estadístico (entrenado en línea o fuera de línea) hasta un conjunto de reglas (por ejemplo, usando la coincidencia de patrones para la comprensión del idioma (LU) o plantillas para la generación del idioma (LG)). Además, cada componente puede llamar a una API o servicio como Google Cloud, Amazon Transcribe o Polly para reconocimiento de voz, síntesis de voz o cualquier otra función. Además de crear aplicaciones de IA conversacionales completas, Plato se puede usar para evaluar y experimentar con varios tipos de tareas de procesamiento de lenguaje natural (PLN), como análisis de sentimientos, modelado de temas, seguimiento de estado de diálogo, generación de lenguaje social y otros. [28]

Ventajas de Plato

- Reconocimiento de voz y síntesis de voz (transcribir voz a texto y viceversa).
- Comprensión del lenguaje y generación del mismo (extraer el significado de ese texto y convertir el significado abstracto a texto).
- Seguimiento de estado (información agregada sobre lo que se ha dicho y hecho hasta ahora).
- Llamada a la API (buscar en una base de datos, consultar una API, etc.).
- Política de diálogo (generar un significado abstracto de la respuesta del agente).

Desventajas de Plato

- Poca información o tutoriales principalmente en video, reseñas o comentarios de uso de terceros en internet. No tiene una gran comunidad si se compara con otras plataformas como BotPress y Rasa.
- El trabajo es por comandos, pero el proceso de desarrollo en sí para alguien con poca experiencia le resultaría más complicado de llevar a nivel técnico en comparación como, por ejemplo; Rasa, que también es por comandos y trabaja igualmente con archivos llegando a ser más flexible y de mejor entendimiento a nivel técnico al desarrollar Asistentes Virtuales.

Conclusión sobre las plataformas para la creación de Asistentes Virtuales

De las plataformas estudiadas lo mejor es el uso de sistemas Open Source (Código Abierto) como lo son Plato, BotPress y RASA por su manejo y la posibilidad de modificar en aras de adaptar y dar mejores facilidades en comparación con las otras plataformas, permite una mayor colaboración entre los que usan este tipo de sistemas y por lo tanto tienen una gran comunidad de desarrolladores activos. En el caso de las mencionadas anteriormente se consideró como mejores opciones a BotPress y RASA; porque a nivel de flexibilidad de trabajo, funciones y proceso de desarrollo creo que tienen mejores condiciones que Plato; por lo que, si se quiere un AV fiable, simple en funcionamiento, personalizable y que esté disponible rápidamente, Botpress es la mejor opción. Si busca un AV fiable también, con algo más de complejidad en funcionamiento y desarrollo debido a las herramientas avanzadas que proporciona que hacen que su agente conversacional sea bastante sólido y completo, así como la gran comunidad global activa y bastos tutoriales que posee esta plataforma, Rasa es la mejor opción. Una relación entre estas dos plataformas que abarca casi cualquier proyecto dependiendo de su uso final, pero independientemente de su magnitud e impacto.

Las otras plataformas: Azure Bot Service, DialogFlow y Amazon Lex si bien son bastante completas y son las más usadas por las grandes empresas, estas son sistemas privados y hoy la industria del software libre es un gran paso de avance en la tecnología, una mejor alternativa, está en ascenso y es más flexible para los desarrolladores; aunque Azure Bot Service es la mejor alternativa entre estos sistemas privados si tiene ya experiencia y se necesita una buena estructura para un gran proyecto, pero si se tiene poca experiencia la mejor es DialogFlow.

Para el trabajo de investigación se contará con la plataforma Rasa, porque su forma de trabajo permite modificar o configurar a nivel de archivos el conocimiento necesario para los asistentes virtuales desarrollados en ella. Además, Rasa posee una comunidad activa y una rica documentación, así como las facilidades que brinda para desarrollar y entrenar sus agentes conversacionales gracias a sus herramientas de procesamiento de información, integración con otros sistemas, posibilidad de probar, evaluar, mejorar y personalizar.

1.3 Herramientas, Lenguajes de programación y Tecnologías

Las herramientas son objetos elaborados a fin de facilitar la realización de una tarea. Se diseñan y fabrican para cumplir uno o más propósitos específicos, por lo que son generalmente artefactos con una función técnica. Un lenguaje de programación es un lenguaje formal que especifica una serie de instrucciones para que una computadora produzca diversas clases de datos. Los lenguajes de programación pueden usarse para crear programas que pongan en práctica algoritmos específicos los cuales controlan el comportamiento físico y lógico de una computadora. La tecnología es la ciencia aplicada a la resolución de problemas concretos. Constituye un conjunto de conocimientos científicamente ordenados, que permiten diseñar y crear bienes o servicios que facilitan la adaptación al medio ambiente y la satisfacción de las necesidades humana.

Python

Python en su versión 3.7 es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Posee una licencia de código abierto, denominada Python Software Foundation License que es compatible con la Licencia pública general de GNU [29]. Se utilizó para la implementación de la herramienta informática.

Se escogió este lenguaje de programación ya que es el más usado en el trabajo con las técnicas de procesamiento del lenguaje natural e inteligencia artificial, es más flexible a la hora de implementar por la gran cantidad de librerías para casi todo tipo de tareas o funciones. Posee una gran comunidad activa de desarrolladores lo que lo posiciona como uno de los más usados actualmente, incluso llegando a ser en varias ocasiones el más utilizado.

PyCharm Community

PyCharm Community Edition 2021.1.2 es un entorno de desarrollo integrado (EDI) utilizado en la programación de computadoras, específicamente para el lenguaje Python. Proporciona análisis de código, un depurador gráfico, un comprobador de unidades integrado, integración con sistemas de control de versiones (VCSes) y es compatible con el desarrollo web con Django. PyCharm es multiplataforma, con versiones de Windows, MacOS y Linux. La Edición Community es la versión libre y se publica bajo la Licencia de Apache [30]. Se usó en el sistema operativo Windows para en integración con Python implementar las funcionalidades de la herramienta informática.

Se escogió este EDI ya que es el más popular para desarrollar aplicaciones en Python, es bastante completo, fácil de usar y satisface casi todas las necesidades con las funciones que posee en esta que es su versión libre.

RASA

Rasa es una plataforma de código abierto para desarrollar asistentes virtuales. Las plataformas de código abierto son software con código fuente que cualquiera puede inspeccionar, modificar o mejorar. Al ser de código abierto, los desarrolladores podrán integrar características y funcionalidades adicionales según sus requisitos. La plataforma es fácil de personalizar, flexible, por lo tanto, se puede modificar según sus necesidades.

MongoDB

MongoDB es un sistema de base de datos NoSQL o no relacional, orientado a documentos y de código abierto. [31]

En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON, una especificación similar a JSON (JavaScript Object Notation, es un formato de texto plano para expresar datos estructurados) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Esta base de datos es utilizada por compañías como, por ejemplo: FourSquare, Facebook, Ebay, Google, etc. Se utilizó como sistema de gestión de base de datos y guardar o administrar toda la información que manejará la herramienta informática.

Se escogió este sistema ya que es adecuado para su uso en producción y con múltiples funcionalidades. Permite la lectura y administración rápida de datos en grandes cantidades, como se basa en documentos JSON soporta varios tipos de datos que se necesitarán manejar en la implementación de la herramienta informática, es una forma eficiente y sencilla de guardar cualquier cantidad de información, incluso en un mismo documento pueden haber más de uno.

YAML

YAML es un formato de serialización de datos (La serialización consiste en un proceso de codificación de un objeto en un medio de almacenamiento (como puede ser un archivo, o un buffer de memoria) con el fin de transmitirlo a través de una conexión en red como una serie de bytes o en un formato humanamente más legible como XML o JSON, etc.) Legible por humanos inspirado en lenguajes como XML, C, Python, Perl, así como en el formato de los correos electrónicos. [32]

YAML fue creado bajo la creencia de que todos los datos pueden ser representados adecuadamente como combinaciones de listas, hashes (mapeos) y datos escalares (valores simples). La sintaxis es relativamente sencilla y fue diseñada teniendo en cuenta que fuera muy legible pero que a la vez fuese fácilmente mapeable a los tipos de datos más comunes en la mayoría de los lenguajes de alto nivel.

Se escogió ya que es el formato de los archivos de entrenamiento que usan los robots conversacionales creados en Rasa.

GIT

Git es un software de control de versiones diseñado por 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. 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, lo que en caso de cualquier inconveniente los datos no se perderán, se pueden revertir cambios fallidos y dar un seguimiento a todo el proyecto de forma detallada.

GitHub es una Plataforma de Desarrollo Colaborativo para alojar proyectos en la nube de forma online (en línea) utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador.

Se usó la versión para escritorio de GitHub (GitHub Desktop) ya que es muy cómoda al trabajar sobre una interfaz visual y no a través de comandos, rápido y fácil de usar, permite sincronizar rápidamente con tu cuenta en la nube.

1.4 Metodología de desarrollo de software Programación Extrema (XP)

XP es una metodología ágil de software que tienen como propósito satisfacer a los clientes mediante la entrega temprana y continua de un software funcional, cuando ello implica incluso apoyar el cambio de los requerimientos en cualquier etapa del desarrollo. Está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. [33]

Para desarrollar un proyecto de software, la metodología propone cuatro fases, como se muestra a continuación:



Figura 1.1: Etapas de la Programación Extrema.

Planificación

En esta fase, se plantean a grandes rasgos las historias de usuarios (HU) que son de interés para la primera entrega del producto. Las historias de usuario son construidas con la información proporcionada por los clientes y el equipo de desarrollo comienza a familiarizarse con las herramientas, metodología y prácticas que serán usadas para realizar el proyecto. Las HU fueron diligenciadas por alguno de los desarrolladores, con el fin de que el cliente pudiera concentrar su atención en el análisis del requerimiento o en el caso de que se estuviera evaluando el diseño o una entrega de iteraciones. Pese a que el cliente no fue quien escribió y diligenció las HU, siempre se contó con su revisión previa antes de finalizar la reunión. Las HU representan los requerimientos de software, y son descritas bajo el lenguaje del cliente.

Diseño

En esta fase se establece la prioridad de cada HU y, correspondientemente, los programadores establecen una estimación de esfuerzo necesario para cada una de ellas. El orden de las historias implementadas en las iteraciones será determinado por el cliente.

Codificación

Desde un principio se hacen pruebas en XP para favorecer entregas frecuentes al cliente que es el objetivo fundamental de la metodología, es imprescindible la participación del cliente como tal, o uno de los elementos fundamentales de la metodología el “Cliente In Situ” que es un representante del cliente.

Prueba

Se realizan pruebas y se verifica que se han implementado todas las HU definidas en la fase de planificación o sus actualizaciones, es de vital importancia la participación del cliente en estas verificaciones.

Conclusiones del capítulo

En este capítulo se plasma toda la información que se recopiló para el análisis respecto al objeto de investigación de este trabajo. Se realizó un estudio de los marcos para desarrollar asistentes virtuales y, teniendo en cuenta sus ventajas, desventajas y características se escogió el más idóneo para la investigación y porqué. Se pone de manifiesto, luego de la revisión de fuentes bibliográficas, las principales tecnologías, herramientas y la metodología de software empleadas, explicando el porqué de la elección de ellas. Además, se reflejan aquellos conceptos manejados para su mayor comprensión, posibilitando la comprensión de las tecnologías empleadas.

Capítulo 2. Planificación y Diseño

En este capítulo se documentan los aspectos más relevantes que fueron identificados durante el estudio del contexto del software y el sistema a partir de las necesidades encontradas. Se detallan los requerimientos del sistema mediante las Historias de Usuarios, así como aspectos relacionados con el diseño del *software*, tales como: modelo de clases, diagramas físicos de la base de datos y la arquitectura.

2.1 Propuesta del Sistema

Con el objetivo de dar solución al problema planteado anteriormente se propone desarrollar un Sistema de Generación de Conocimiento Automático (**SGCA**) para un asistente virtual desarrollado en Rasa, que permita a partir de la entrada de datos generar archivos de entrenamiento para el asistente virtual, entrenarlo e interactuar con el mismo.

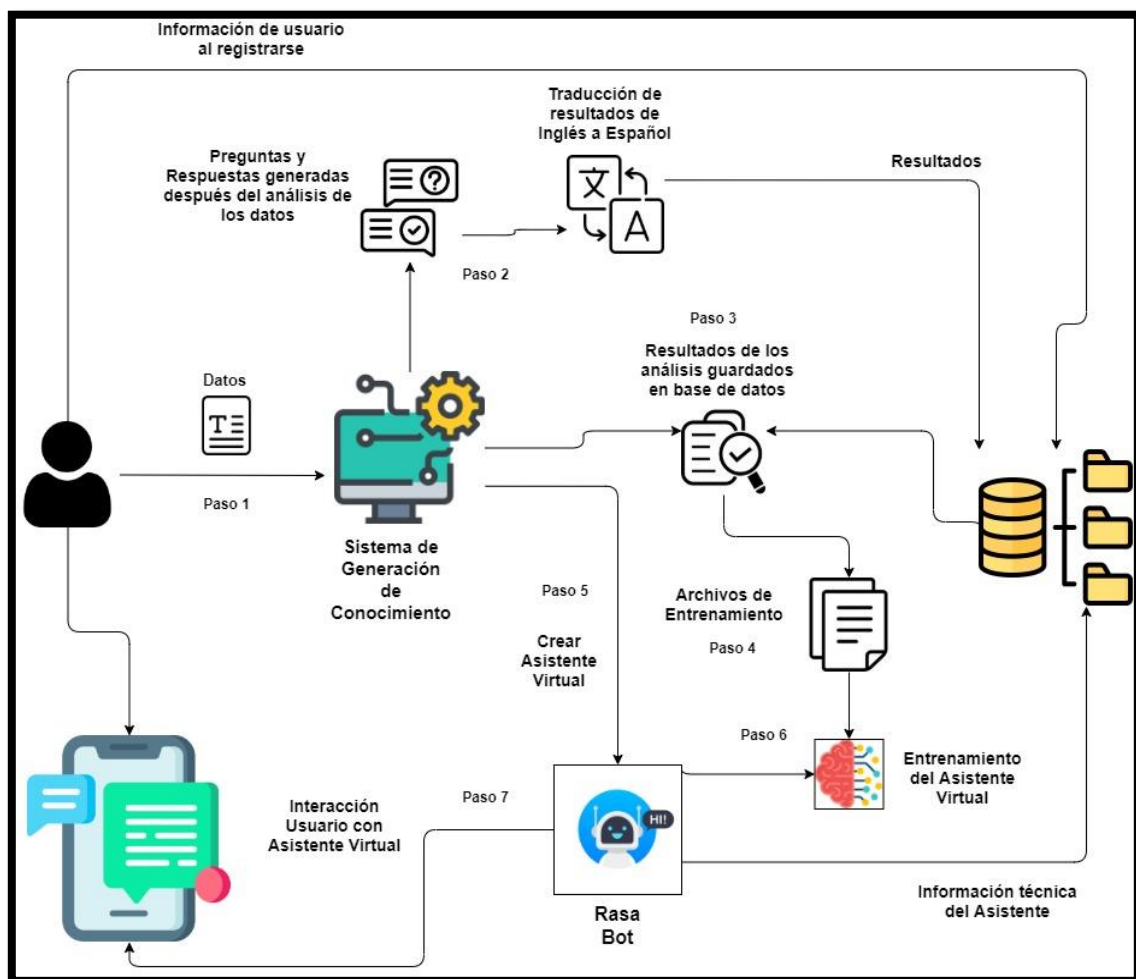


Figura 2.1. Estructura del Sistema

2.2 Usuarios del Sistema

Los usuarios son todas aquellas personas que interactúan de alguna forma con esta herramienta o que desempeñan algún rol específico. A continuación, se describen las actividades que realiza cada rol de usuario.

Tabla 2.1 Usuarios del Software

Usuario	Descripción
Administrador	Acceso a funciones internas del sistema, base de datos y procesos en general del sistema.
Cliente	Acceso a realizar las operaciones comunes para lo que fue diseñada la herramienta, dichas operaciones se realizan desde la interfaz gráfica de la misma.

2.3 Planificación del desarrollo del Sistema

La planificación es la etapa inicial de todo proyecto. Es aquí donde se comienza a interactuar con el cliente para descubrir los requerimientos del sistema y realizar los ajustes a la metodología según las características del software. Se definen los requisitos funcionales y no funcionales.

2.3.1. Requisitos Funcionales

RF1: Autenticar usuario: Permitirá el acceso al sistema y funcionalidades correspondientes.

Entrada: Usuario y contraseña.

Salida: El usuario accede al sistema.

RF2: Generar preguntas y respuestas: Es donde el usuario a partir de una entrada de datos al sistema, este último analizará dichos datos y generará preguntas y respuestas sobre el contenido.

Entrada: Datos.

Salida: Preguntas y respuestas.

RF3: Carga y guardado de datos. Mediante una base de datos, los datos se guardan para ser utilizados luego y no tener que volver a cargar los mismos datos nuevamente.

Entrada: Datos y resultados.

Salida: Guarda los datos con posibilidad de volver a ser usados.

RF4: Generar conocimiento del Asistente: Es donde creará los archivos de entrenamiento a partir de los resultados obtenidos del análisis de los datos.

Entrada: Datos.

Salida: Archivos de entrenamiento.

RF5: Crear Asistente virtual: Proporciona la posibilidad de crear el asistente virtual a partir de cierta información.

Entrada: Información pedida por el sistema.

Salida: Se crea el asistente virtual.

RF6: Entrenar Asistente: Permitirá una vez creado el asistente y los archivos de entrenamiento poder entrenarlo.

RF7: Integración a servicio web Chat: El sistema crea automáticamente la configuración para que el asistente virtual pueda responder desde el sitio web que lo requiera.

Entrada: Datos de configuración para el asistente y widget chat de Rasa para la web

Salida: Interacción desde una web mediante chat.

RF8: Probar Asistente: Permitirá probar el asistente.

Entrada: Datos del asistente a probar.

Salida: Se inicia servidor del asistente. Se inicia una web con servicio de chat donde se produce la interacción de prueba.

RF9: Conexión con cliente de mensajería: Permitirá conectar el asistente virtual creado con el cliente de mensajería Telegram a través de un Telegram Bot.

Entrada: Datos de Telegram Bot y servidor de peticiones web (webhook).

Salida: Interacción desde Telegram con su asistente virtual.

Nota: Estos requisitos funcionales se convertirán en historias de usuario.

2.3.2. Requisitos no Funcionales

Requisitos de Software

- Microsoft Windows 10 u 11
- Python 3.8
- MongoDB
- Paquetes de dependencias de la herramienta informática instalables a través de Python.

Requisitos de Hardware

Los requerimientos mínimos de hardware para correr la aplicación:

- Procesador Intel Core i3 a 2.3 GHz
- 4.0 GB de memoria RAM
- Buena conexión a Internet
- Buen espacio de almacenamiento libre

2.3.3. Historias técnicas

Las funcionalidades son complementadas por las historias técnicas que se enfocan en el diseño o la implementación. Las historias técnicas constituyen las propiedades o cualidades que el producto debe tener.

Usabilidad

El sistema debe brindar al usuario una clara navegabilidad de las funcionalidades del sistema.

Confiabilidad

El sistema debe ser capaz de funcionar correctamente cada vez que se utilice, las funciones para las que fue diseñado deben arrojar los resultados que el usuario espera, dichos resultados deben ser íntegros.

Seguridad

El espacio de trabajo de cada usuario será protegido mediante ***Autenticación personal*** para entrar y poder realizar acciones con la herramienta. La información proporcionada será almacenada en bases de datos, por tanto, el sistema debe asegurar la información que en la misma se manipulan, como por ejemplo la contraseña debe estar cifrada.

2.3.4. Historias de usuario

Las Historias de Usuarios (HU) son la técnica utilizada en XP para especificar los requisitos de software, tanto funcionales como no funcionales y se descomponen en tareas de ingeniería asignadas a los programadores para ser implementadas durante las iteraciones. Están escritas con el vocabulario del cliente, no con vocabulario técnico. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento las HU pueden romperse, emplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas.

- HU.1 Autenticación del Sistema
- HU.2 Crear Asistente Virtual
- HU.3 Gestión de Conocimiento
 - HU.4 Generar Preguntas y Respuestas
 - HU.5 Carga y guardado de datos
- HU.6 Entrenar Asistente Virtual.
 - HU.7 Integración con servicio web chat
 - HU.8 Integración con cliente de mensajería
- HU.9 Probar Asistente Virtual

Algunas de las principales HU

Tabla 2.2 Historia de Usuario: Autenticación del Sistema

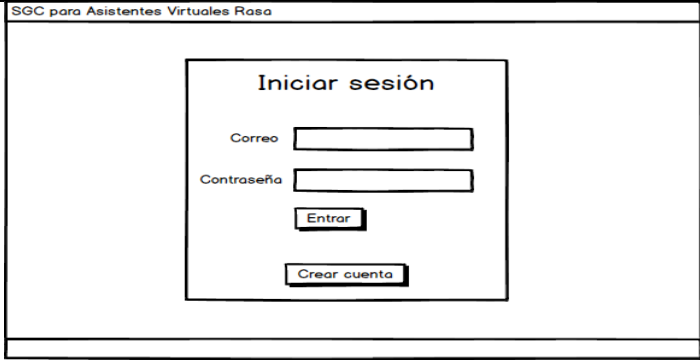
Historia de usuario	
Número:1	Nombre de HU: Autenticación del Sistema
Usuario: Cualquier usuario	
Prioridad en negocio: Alta	Riesgo de desarrollo: Bajo
Descripción: Donde el usuario inicia sesión para entrar al sistema, también en caso de no tener cuenta puede crear una.	
	

Tabla 2.3 Historia de Usuario: Gestión de Conocimiento

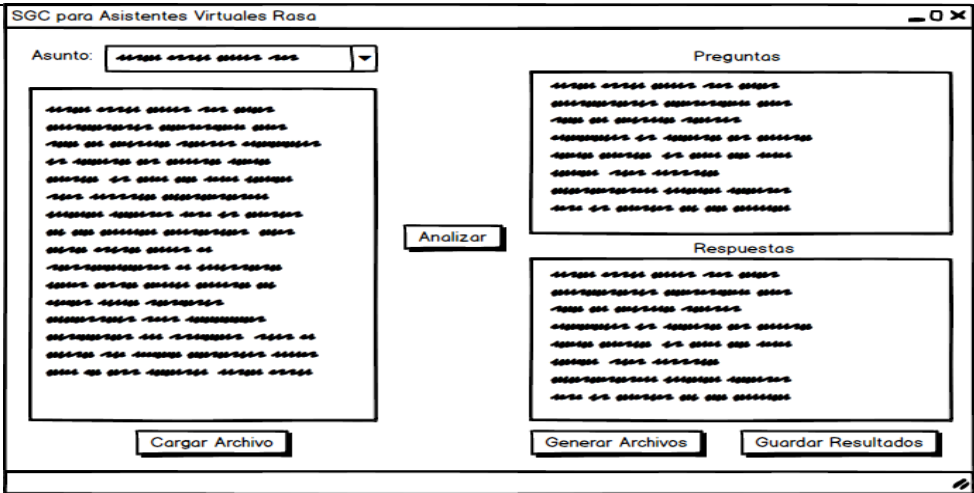
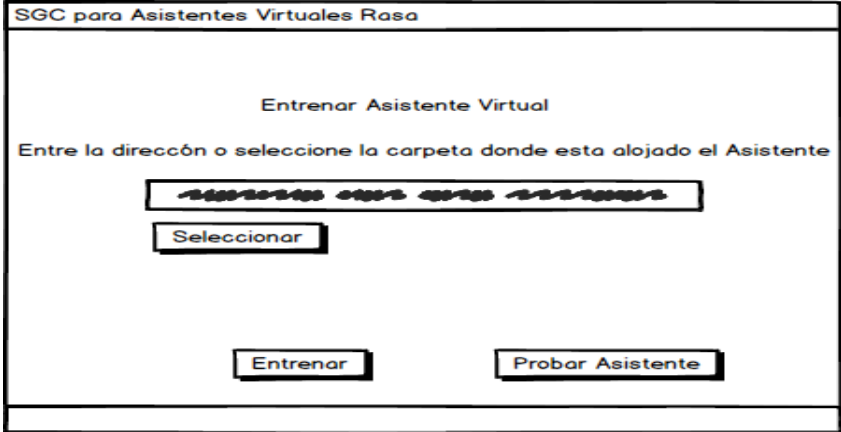
Historia de usuario	
Número:3	Nombre de HU: Gestión de Conocimiento
Usuario: Cualquier usuario	
Prioridad en negocio: Alta	Riesgo de desarrollo: Alta
Descripción: Donde se realizan todas las operaciones de generar conocimiento para el asistente virtual.	
	

Tabla 2.4 Historia de Usuario: Entrenar Asistente Virtual

Historia de usuario	
Número: 4	Nombre de HU: Entrenar Asistente Virtual
Usuario: Cualquier usuario	
Prioridad en negocio: Alta	Riesgo de desarrollo: Bajo
Descripción: Donde el usuario puede entrenar su Asistente Virtual.	
	

2.3.6. Arquitectura del Sistema

Para el desarrollo del sistema fue utilizada la siguiente arquitectura:

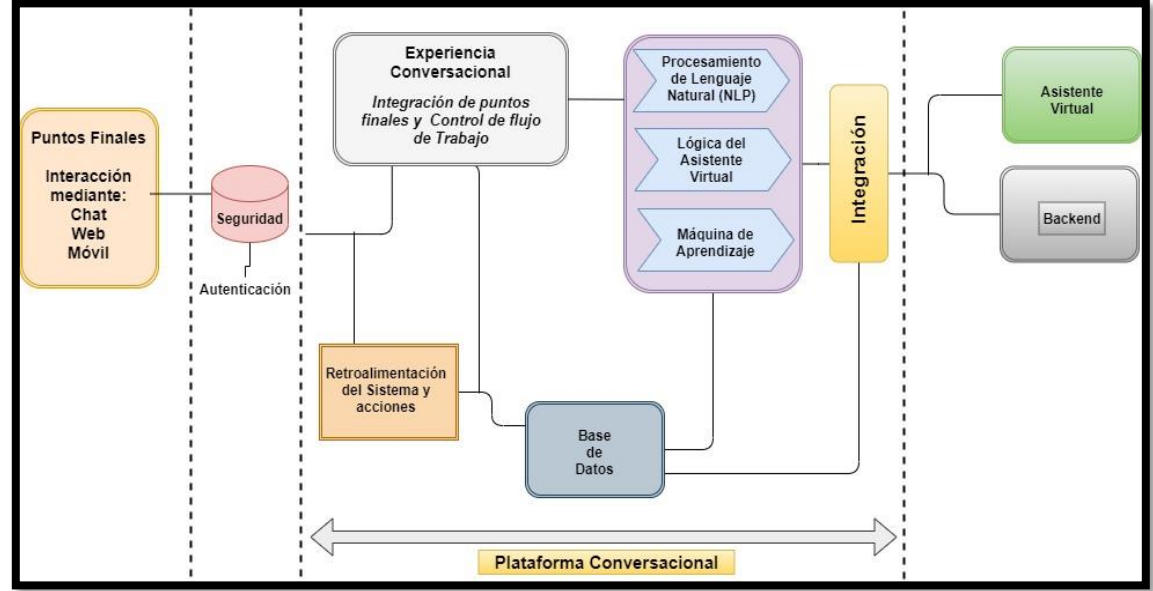


Figura 2.2 Arquitectura del Sistema

- Primeramente, el usuario se registra en el sistema (Información de usuario que se guarda en base de datos). Una vez registrado ya puede iniciar sesión y usar las funcionalidades de la herramienta.
- Luego el usuario entra los datos (en inglés) al sistema y este analiza dichos datos, dando como resultado preguntas y respuestas relacionadas con el contenido de la información proporcionada. Los resultados serán traducidos al idioma español y guardado en la base de datos.
- El sistema tomará los resultados de la base de datos y generará los archivos de entrenamiento para el asistente virtual.
- El sistema permite crear el asistente (la información técnica del asistente se guarda en base de datos) a través de la plataforma Rasa y entrenarlo con los archivos generados.
- El usuario una vez haya transitado por todo el proceso puede establecer una conversación con el asistente creado y verificar la experiencia a través de los puntos finales: chat, web, móvil.

2.3.7. Diseño de Base de Datos

El sistema cuenta con una base de datos no relacional (NoSQL) al usar MongoDB que está orientado a documentos (Están diseñadas para almacenar, recuperar, administrar datos y manejar conjuntos de datos de longitud no fija). Los sistemas NoSQL no requieren de un esquema relacional, ni de tablas y columnas ya que las colecciones (las colecciones son como las tablas en los sistemas SQL y constituyen un conjunto de documentos) y documentos (los documentos sustituyen a las filas y columnas de los sistemas SQL) en sí hacen el equivalente a ellas y los documentos no tienen esquema o estructura predefinidas por lo que se le pueden añadir campos a voluntad (Se puede hacer una representación opcional para mejor entendimiento **Figura 2.3**). Contamos con cuatro colecciones que contienen documentos que poseen las claves y los valores de dichas claves. (**clave -> valor**; ya que MongoDB usa documentos en formato JSON).

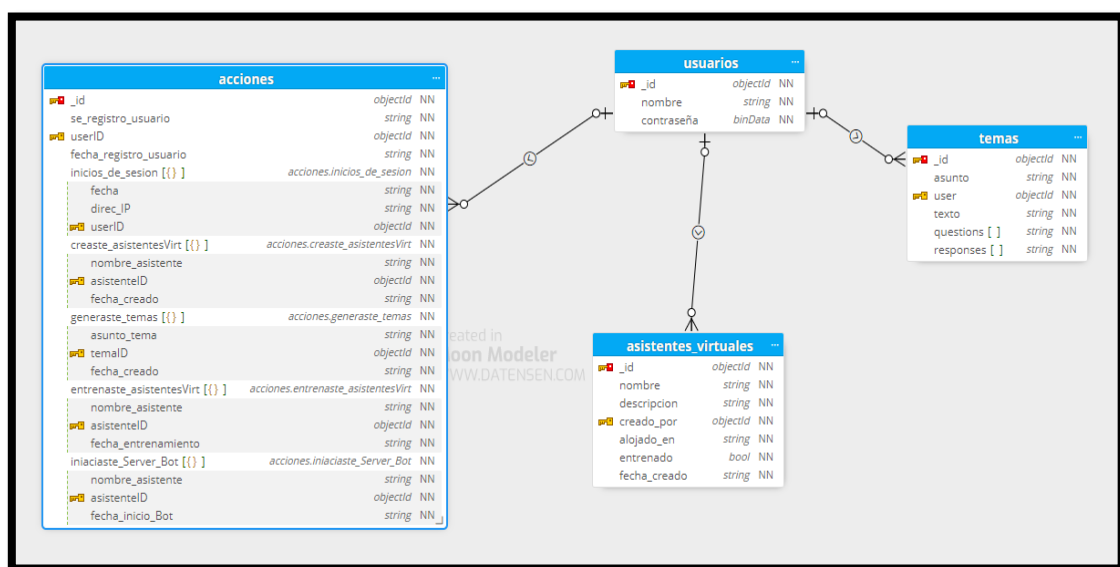


Figura 2.3 Representación de Base de Datos NoSQL

Colecciones Presentes

- **Usuarios:** Es donde se hallan los documentos con la información correspondiente a los usuarios que usan la herramienta. (nombre, contraseña de inicio de sesión, etc).
- **Asistentes Virtuales:** Es donde se encuentra los documentos con la información respecto al robot conversacional (nombre, descripción, etc.)
- **Temas:** Es donde se localizan los documentos con la información y los resultados (texto con el contenido analizado, las preguntas y respuestas derivadas del texto, etc.)
- **Acciones:** Es donde quedan registradas en sus documentos las acciones del usuario al interactuar con sistema (registro de usuario, inicios de sesión, operaciones respecto a las funcionalidades de la herramienta).

Seguridad

- La contraseña que crea el usuario será encriptada o cifrada. Y al iniciar sesión en el sistema la contraseña al escribir no será visible.
- Se registrará también las fechas, hora y la IP de la computadora en la que el usuario inicia sesión en el sistema.
- También se registran las acciones que realiza en usuario al transitar por las funcionalidades del sistema.

Conclusiones del capítulo

En el presente capítulo se mostraron aspectos relacionados con las fases de planificación y diseño del desarrollo del software. Quedaron expuestas las funcionalidades a implementar definidas por los artefactos que propone la metodología XP, se mencionan los requisitos tanto funcionales como los no funcionales, las historias de usuarios y las historias técnicas. Se representó la base de datos no relacional que usamos en la investigación y sus particularidades; Se requirió una base de datos ya que es necesario guardar información que puede ser utilizada nuevamente, ya sea para ser procesada por la herramienta informática (cómo el texto, las preguntas y respuestas) o para tener un control interno de datos acerca de procesos que tienen que ver con el funcionamiento de la aplicación. También se mencionaron aspectos esenciales a tener en cuenta para la aplicación como es la seguridad.

Capítulo 3. Implementación y Prueba

En este capítulo se presentarán algunas de las funcionalidades del sistema diseñado y detalles de su implementación, explicando las herramientas software utilizadas y aspectos relevantes que han tenido lugar en su elaboración. Se expondrán los resultados de los casos de prueba que se llevaron a cabo una vez terminado el proyecto.

3.1. Instalación de las herramientas utilizadas

1. Instalación de Python 3.8: Ejecutar el instalador y seguir los pasos que se le piden. (**Marcar opción Agregar a Path que sale en la interfaz de instalación.**)
2. Instalación de MongoDB: Ejecutar Instalador y seguir los pasos de instalación hasta el final sin cambiar nada en el proceso.
3. Instalación del PyCharm Community Edition: Ejecutar el instalador, luego lo utilizamos para la programación o implementación de las funcionalidades con el lenguaje Python.
4. Instalación opcional de GitHub Desktop: Ejecutamos Instalador, una vez instalado pudimos poner nuestra cuenta personal en github y alojar nuestro proyecto allí para tenerlo salvado y controlar todos los cambios que se hicieron durante la implementación del mismo.
5. Se implementaron las funciones para autenticación de usuario.
6. Para la generación de las preguntas y las respuestas se buscó una solución de terceros donde la entrada de datos debía ser en inglés. Sin embargo, se implementaron otras soluciones para entrada estrictamente en español que es el idioma destino del sistema y fueron tomadas en cuenta e integradas al sistema también.
7. Usando la biblioteca de Python `ramuel.yaml` para el manejo de los archivos `.yaml`, se crearon las funciones para crear cada archivo de entrenamiento. Dichas funciones reciben las preguntas y respuestas extraídas de los datos entrados.
8. Luego se crearon las funciones para los procesos de creación, entrenamiento y prueba del Asistente Virtual.

3.2 Implementación de Funcionalidades

El usuario deberá crear una cuenta para luego iniciar sesión, a continuación, se muestran las funciones que se implementaron para ello.

```
def register():
    ##### PIDIENDO DATOS PARA REGISTRO #####
    print("REGISTRARSE EN EL SISTEMA.")
    name = input('Cree su nombre de usuario : ')
    password = input('Cree su contraseña : ')

    # SI el usuario existe
    existe_user = collection.find_one({'nombre': name})
    if existe_user is None:
        # ENCRIPtar CONTRASEÑA
        hashpass = bcrypt.hashpw(
            password.encode('utf-8'), bcrypt.gensalt())

        # Insertando datos
        post = {"nombre": name,
                "contraseña": hashpass,
                "fecha_registro": datetime.today().strftime('%Y-%m-%d %I:%M'),
                "fechas_Inicio_sesion": []
                }
        post_id = collection.insert_one(post).inserted_id
        print()
        login()
    else:
        print("Este usuario ya existe. Intente de nuevo")
        register()
```

Figura 3.1 Función para registrarse en el sistema.

```
def login():
    #generateEN = quest
    generateES = questES
    ##### PIDIENDO DATOS PARA REGISTRO #####
    print("INICIAR SESIÓN PARA USAR LA APLICACIÓN")
    name = input('Entre su nombre de usuario : ')
    password = input('Entre su contraseña : ')
    login_user = collection.find_one({'nombre': name})

    if login_user and bcrypt.checkpw(
        password.encode('utf-8'), login_user['contraseña']):

        # Capturando la fecha y hora en que inicia sesion y la IP desde donde se inicia
        hostname = socket.gethostname()
        IPAddr = socket.gethostbyname(hostname)
        collection.update_one(
            {'nombre': login_user['nombre']}, {
                '$push': {'fechas_Inicio_sesion': datetime.today().strftime('%Y-%m-%d %I:%M')+ " IP: " + IPAddr}}
            )
        print()
        # ESPAÑOL bot
        generateES.main()
        # INGLÉS bot
        # generateEN.main()
    else:
        print("ERROR de Usuario o contraseña")
        login()
```

Figura 3.2 Función para iniciar sesión en el sistema.

A continuación, se muestran algunas de las funciones más importantes.

```
#####
# PLANTILLA para Archivo RASA
saludo = ['saludar',
          'despedir',
          'afirmar',
          'negar',
          'animo',
          'no_animo',
          'bot']

utterSaludo = [{'utter_saludar':
                [{'text': "¿Cómo estás?"}],
                {'utter_ayuda':
                [{'text': "¿En qué puedo ayudarte?"}],
                {'utter_feliz':
                [{'text': "¡Genial, continúa!"}],
                {'utter_adios':
                [{'text': "Adiós"}]},
                {'utter_soybot':
                [{'text': "Soy un bot, creado por Rasa."}]}]}

versionRasa = {'version': "3.0"}
intent = {'intents': saludo + ques}
config = {'session_config': {
          'session_expiration_time': 60,
          'carry_over_slots_to_new_session': True, }
}
#####

#####
# Escribiendo la plantilla en el archivo

yaml = YAML()
yaml.indent(mapping=2, sequence=4, offset=2) # Sangria y
margen
yaml.dump(versionRasa, yaml_file)
yaml.dump(intent, yaml_file)
yaml = YAML()
for i in range(len(ques)):
    auxutter.append({'utter_{}'.format(ques[i]): [{'text': res
[i]]})} # Guardando la info de repsonses

toPrintYaml = dict()
for element in utterSaludo: # Poniendo informacion en un
diccionario, clave = valor
    for key, value in element.items():
        toPrintYaml[key] = value

for element in auxutter: # Poniendo informacion en un
diccionario, clave = valor
    for key, value in element.items():
        toPrintYaml[key] = value

yaml.dump(dict({'responses': toPrintYaml}), yaml_file)
yaml.dump(config, yaml_file)

#####
# Validacion para posibles errores
except Exception as error:
```

Figura 3.3 Parte de la función para crear el archivo de entrenamiento **domain.yml**

Este archivo contiene principalmente las intenciones (frases que se espera que diga el usuario) y acciones que tomará el bot.

(A la izquierda se creó la plantilla del archivo y a la derecha se escribió dicha plantilla en el archivo.)

```
nlu = {
  'nlu':
    [{'intent': 'saludar',
      'examples': literal_('\n'.join([
        '- Hola',
        '- que tal',
        '- buenos días',
        '- buenas noches',
        '- buenas tardes']) + '\n')
    },
    {'intent': 'despedir',
      'examples': literal_('\n'.join(['- adiós',
        '- buenas noches',
        '- hasta luego'])
        + '\n')},
    {'intent': 'afirmar',
      'examples': literal_('\n'.join(['- Sí',
        '- por supuesto',
        '- correcto']) +
        '\n')},
    {'intent': 'negar',
      'examples': literal_('\n'.join(['- no',
        '- nunca',
        '- de ninguna
manera',
        '- no realmente'
        ]) + '\n')},
    {'intent': 'animo',
      'examples': literal_('\n'.join(['- perfecto',
        '- genial',
        ])
```

Figura 3.4 Parte de la función para crear el archivo de entrenamiento **nlu.yml**

Este archivo contiene principalmente las intenciones ya definidas y ejemplos por los que se guiará el bot de acuerdo a cada intención.

(A la izquierda se creó la plantilla del archivo y a la derecha se escribió dicha plantilla en el archivo, se puede alcanzar a ver el manejo de errores)


```

def cargarAsistentes(user):
    print()
    collection = db['bots_virtuales']
    nombre_asistente = input(
        "Escriba el nombre del asistente que desea entrenar: ")
    if (nombre_asistente):
        try:
            asistente = collection.find_one(
                {'nombre': nombre_asistente, 'creado_por': user}) #
            # Comprobar que el usuario corresponde al usuario
            # actual, si no se corresponde se genera una excepcion
            # o error
            print()
            print('Vista de los datos correspondientes al asistente
            elegido')
            print()
            print('Nombre: ' +
                str(asistente['nombre']))
            print()
            print('Descripción: ' + str(asistente['descripcion']))
            print()
            print('Alojado en: ' + str(asistente['alojado_en']))
            print()
            if (asistente['entrenado']):
                print('Ha sido entrenado: ' + "Si")
                print()
            else:
                print('Ha sido entrenado: ' + "No")
                print()
            print('Se creó en la fecha: ' + str(asistente
            ['fecha_creado']))
            print()
            confirmar = input(
                "Confirme que este es el que desea cargar: Escriba
                si o no: ")
            if (confirmar == 'si'):
                print('Ha sido entrenado: ' + "No")
                print()
                print('Se creó en la fecha: ' + str(asistente['fecha_creado']))
                print()
                confirmar = input(
                    "Confirme que este es el que desea cargar: Escriba si o no: ")
                if (confirmar == 'si'):
                    print()
                    print("El directorio o carpeta dónde está su Asistente Virtual
                    es: ", str(
                        asistente['alojado_en']))
                    confirmarDir = input(
                        "Confirme que este es el directorio o carpeta actual:
                        Escriba si o no: ")
                    if (confirmarDir == 'si'):
                        print()
                        visualConocimiento = "rasa visualize"
                        train = "rasa train"
                        if mover(str(
                            asistente['alojado_en'])):
                            print("Entrenando al Asistente")
                            print("Espere..... Esta cargando...")
                            os.chdir(str(
                                asistente['alojado_en']))
                            os.system(train) # ENTRENANDO
                            print()
                            collection.update_one(
                                {'nombre': nombre_asistente, 'creado_por': user}, {
                                    '$set': {'entrenado': True}}
                            )
                            os.system("cls")
                            print("Se le mostrará un gráfico en su navegador donde
                            podrá ver las preguntas y las respuestas inferidas a
                            cada pregunta (utter_pregunta) después del
                            entrenamiento, para verificar que el Asistente tendrá
                            alta probabilidad de responder correctamente.")
                            print()

```

Figura 3.6 Función para entrenar el Asistente Virtual (Dónde primero se buscan y muestran los datos de entrenamiento, asistentes virtuales creados por el usuario. Luego el usuario elige cual desea entrenar y que datos usar, y termina ejecutándose el entrenamiento.)

```

def generarArchivos(preguntas, respuestas):
    if (fileDomain.domYaml(preguntas, respuestas) &
        fileNLU.nluYaml(preguntas, respuestas) &
        fileStories.storiesYaml(preguntas, respuestas) &
        fileRules.rulesYaml(preguntas, respuestas)):
        os.system("cls")
        print(
            '\n' + "Archivos de entrenamiento creados con Exito :) " + '\n' +
            '.....')
        print()
    else:
        print()
        print("Algo salio mal al generar archivos :( ")

```

Figura 3.7 Función para generar los archivos de entrenamiento al recibir las preguntas y respuestas extraídas. Se llaman a las funciones que mostramos al principio que se encargan de crear los archivos.

3.3 Análisis económico

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

3.3.1. Estimación de costo y tiempo

Para determinar los costos de los sistemas desarrollados se usó el método de puntos en casos de uso [34]. 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 realizó este análisis teniendo en cuenta las Historias de Usuario proporcionadas por la metodología XP.

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

$$\text{UUCP} = \text{UAW} + \text{UUCW}$$

Donde:

- **UUCP:** Puntos de Historias de Usuarios sin ajustar.
- **UAW:** 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.

Tabla 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

Para calcular UUCW

Este valor se calcula mediante un análisis de la cantidad de HU presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los HU se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo según muestra la [tabla 3.2](#).

Tabla 3.2 Peso de las Historias de Usuarios sin Ajustar

Tipo	Descripción	Peso	Cant * peso
Simple	La HU contiene de 1 a 3 transacciones.	5	8*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			40+10+0= 50

Cálculo de los Puntos de Historias de Usuarios ajustadas.

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

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

$$UCP = UUCP * TCF * EF$$

Donde:

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

Factor de complejidad técnica (TCF).

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante o nulo y 5 un aporte muy importante. En la siguiente [tabla 3.3](#) se muestra factor de complejidad técnica con su significado y el peso de cada uno de estos factores:

Tabla 3.3 Factor de complejidad técnica

Factor	Descripción	Peso	Valor	(Peso-i * Valor-i)
T1	Sistema distribuido	2	2	2
T2	Rendimiento o tiempo de respuesta	1	5	5
T3	Eficiencia del usuario final	1	2	2
T4	Procesamiento interno complejo	1	3.5	3.5
T5	El código debe ser reutilizable	1	3	3
T6	Facilidad de instalación	0.5	2	1
T7	Facilidad de uso	0.5	5	2.5
T8	Portabilidad	2	3	6
T9	Facilidad de cambio	1	5	5

T10	Concurrencia	1	4	4
T11	Incluye objetivos especiales de seguridad	1	5	5
T12	Acceso directo a terceras partes	1	2	2
T13	Se requieren facilidades especiales de entrenamiento a los usuarios	1	2	2
Total				43

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

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

$$\text{TCF} = 0.6 + 0.01 * 43$$

$$\text{TCF} = 1.03$$

Factor Ambiente (EF).

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

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

Tabla 3.4 Factor de ambiente

Factor	Descripción	Peso	Valor	(Peso-i * Valor-i)
E1	Familiaridad con el modelo de proyecto utilizado	1.5	3.5	5.25
E2	Experiencia en la aplicación	0.5	3.5	1.75
E3	Experiencia en orientación a objetos	1	3	3
E4	Capacidad del analista líder	0.5	3.5	1.75
E5	Motivación	1	4.5	4.5
E6	Estabilidad de los requerimientos	2	4	8
E7	Personal part-time	-1	0	0
E8	Dificultad del lenguaje de programación	-1	1	-1.5
Total				22.7

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

Ecuación 3.4: Cálculo del Factor de ambiente

$$\text{EF} = 1.4 - 0.03 * \Sigma (\text{Peso}_i * \text{Valor}_i)$$

$$\text{EF} = 1.4 - 0.03 * 22.7$$

$$\text{EF} = 0.72$$

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

$$\text{UCP} = 32 * 1.03 * 0.72$$

$$\text{UCP} = 23.73$$

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

Ecuación 3.5: Esfuerzo estimado en horas hombres.

$$E = UCP * CF$$

Donde:

E: Esfuerzo estimado en horas hombres.

UCP: Punto de historias de usuarios ajustadas.

CF: Factor de conversión.

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

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

Luego

$$E = 23 * 20 \text{ horas-hombre}$$

$$E = 474 \text{ horas-hombre}$$

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

Tabla 3.5 Distribución del esfuerzo por etapas

Actividad	% esfuerzo	Valor esfuerzo
Planificación	10	50
Diseño	20	80
Codificación	40	<u>260</u>
Prueba	15	44
Sobrecarga	15	40
Total	100	474

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 \$12 000 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= 474 horas/176 ≈ equivalente a 2 meses y 69 días, este es el tiempo que tomaría desarrollar el proyecto empleando una sola persona.

2 meses y 69 días a razón de 24 días laborables por mes, representan 109 días. En nuestro caso como empleamos 3 trabajadores para el desarrollo del proyecto, el tiempo para su culminación quedará reducido a 36 días aproximadamente.

Entonces el costo total del proyecto:

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

$$C = 474 * 1.5 * 12\,000/176 = \underline{\underline{\$ 48\,477}}$$

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

Tabla 3.6 Situación actual de pago.

Organización (Sector)	Pago/mensual (Moneda Nacional)
XETID (Estatel)	10 000
DATYS (Estatel)	10 000
DESOFT (Estatel)	10 000
MYPIMES (Privado)	25 000
Freelancer o persona autónoma (Privado)	600 MLC a 120 CUP (tasa oficial actual) son 72 000

3.4 Pruebas al sistema

Un caso de prueba define una forma de examinar el sistema, a través de tanteos, donde se deben entrar datos para comprobar si el sistema devuelve los resultados esperados bajo las condiciones específicas con las que ha de probarse. Uno de los pilares de la metodología XP es el proceso de pruebas. Esta promueve a probar tanto como sea posible, reduciendo así el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección.

Las pruebas de funcionalidad y de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo del software y permiten confirmar que la historia ha sido implementada correctamente. En caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. A medida que se fue implementando la aplicación, se diseñaron un conjunto de casos de prueba para comprobar su funcionamiento de acuerdo a los requerimientos descritos en las HU, que fueron definidas en el capítulo 2.

Algunos de los casos de prueba realizados.

Tabla 3.7 **Caso de Prueba “Autenticación”**

Entrada	Resultados	Condición
Un nombre de usuario.	El sistema lanza una alerta si ya existe ese nombre de usuario.	El nombre de usuario entrado al registrarse ya existe.
Una contraseña.	No se muestra la contraseña al escribir.	Se escribe la contraseña para iniciar sesión.
Usuario y contraseña	Se guarda en base de datos la información de registro e inicio de sesión del usuario actual.	Se registra y entra al sistema.

Tabla 3.8 **Caso de Prueba “Gestión de conocimiento”**

Entrada	Resultados	Condición
Un asunto o título que identifique el contenido de los datos.	El sistema lanza una alerta si ya existe el asunto entrado.	El asunto entrado ya existe.
Datos (Uno o varios textos con determinado contenido).	Un análisis de los datos que da como resultado preguntas y respuestas referentes al contenido de dichos datos.	Entrada de datos para procesar.
Datos y resultados de análisis.	Los datos y resultados de análisis se guardan en la base de datos.	Una vez acabado el análisis y se arrojen los resultados.
Finaliza el proceso	El sistema pregunta si desea realizar un nuevo análisis con nuevos datos.	Culmina el proceso de generación de conocimiento actual.
Carga de datos y resultados guardados	Se crean los archivos de entrenamiento con la información cargada.	Se cargan los datos y se llama a la función para crear los archivos de entrenamiento.

Tabla 3.9 **Caso de Prueba “Entrenar Asistente Virtual”**

Entrada	Resultados	Condición
Elige la opción entrenar asistente.	Se muestra una lista solo con los asuntos (datos guardados) disponibles del usuario actual.	Al entrar en la opción entrenar asistente.
En caso de que se equivoque al escribir el asunto que desea cargar.	El sistema lanza una alerta de que ese asunto no está entre los pertenecientes al usuario actual. Y posibilita volver a intentarlo.	Escribe el asunto con los datos que se elegirá para entrenar al asistente.
Elige la opción entrenar asistente.	Se muestra una lista solo con los asistentes disponibles pertenecientes al usuario actual.	Al entrar en la opción entrenar asistente.
En caso de que se equivoque al escribir el asistente que desea entrenar.	El sistema lanza una alerta de que ese asistente no está entre los pertenecientes al usuario actual. Y posibilita volver a intentarlo.	Escribe el nombre del asistente que se entrenará.
Carga el asistente a entrenar.	El sistema pide confirmación de ubicación o carpeta donde está alojado el asistente (es importante saber la ubicación actual del asistente para realizar el entrenamiento y probar el asistente). El usuario responde si o no a la ubicación mostrada.	Se escribe el nombre y se carga el asistente.
Carga el asistente y confirmación correcta	Comienza el proceso de entrenamiento.	Se confirma la ubicación al cargar el asistente.
El asistente fue cambiado de ubicación por el usuario.	El sistema permite actualizar la información de ubicación y seguir con el proceso de entrenamiento correctamente.	El usuario responde no a la confirmación de ubicación.

Tabla 3.10 **Caso de Prueba “Probar Asistente”**

Entrada	Resultados	Condición
Carga el asistente elegido y confirmación correcta.	Se inicia el servidor del asistente virtual.	Se confirma la ubicación al cargar el asistente.
Se inicia la web de prueba.	Se inicia un servidor web y lanza un sitio web con soporte para chat. El usuario interactúa con el asistente.	El servidor del asistente está corriendo.
Un mensaje enviado al asistente.	El asistente no responde al azar con cualquier respuesta que tenga en su base de conocimiento, sino que concretamente dice que <i>“Lo siento, no puedo entender o manejar lo que acabas de decir.”</i>	El mensaje enviado no fue entendido por el asistente virtual.

Conclusiones del capítulo

En este capítulo se detalló la instalación y uso de las herramientas, tecnologías usadas. Se describió la implementación de algunas de las principales funcionalidades para mejor entendimiento. Las pruebas realizadas al sistema permitieron ver las capacidades del mismo para responder en diferentes situaciones que se pueden dar, además de verificar como se desenvuelven algunas de las funcionalidades. El análisis del tiempo y costo del proyecto permitió evidenciar la magnitud y la viabilidad del mismo, además se llevó a cabo una investigación de los salarios que se pagan en algunas entidades y sectores de la provincia para mejor evaluación.

Conclusiones

- Se realizó un estudio del estado del arte de los marcos para crear asistentes virtuales, lo que permitió usar el más apto y óptimo para realizar el trabajo de investigación, la plataforma Rasa fue la que resultó elegida.
- Se diseñó el prototipo de la herramienta informática que permite la construcción de conocimiento para un asistente virtual de forma automática.
- Se definieron las herramientas de desarrollo y se implementó la herramienta diseñada.
- Se probó la herramienta diseñada en varias esferas de conocimiento humano y se verificaron los resultados.
- Se desplegó la herramienta junto a la plataforma para crear asistentes virtuales antes mencionada, dando como resultado una integración que permitió cumplir el objetivo y la hipótesis propuestas, así como mejorar las capacidades de respuesta de los asistentes, la interacción y atención a los usuarios.

Recomendaciones

- Continuar perfeccionando el sistema, que integrando nuevas tecnologías permita quizá la construcción de módulos para esferas específicas de conocimiento donde el asistente virtual tenga que tomar decisiones a partir de una situación planteada por el usuario; como por ejemplo un análisis médico que a partir de los síntomas que diga el usuario, el asistente pueda analizar y decidir que determinada enfermedad padece.
- Perfeccionar una interfaz visual para determinadas opciones del sistema.
- Extender el servicio de chat de los asistentes virtuales por vía correo electrónico, mensajes vía SMS por móvil.
- Extender a idioma inglés la interacción con el asistente virtual.

Referencias Bibliográficas

- [1.] [Robot conversacionales: La guía definitiva \(2020\) - IA Conversacional para Empresas | Artificial Solutions \(artificial-solutions.com\)](#) 11-2-2022
- [2.] Fourault, S. (2017). The Ultimate Guide to Designing A Robot conversacional Tech Stack. Retrieved from <https://robotconversacionismagazine.com/the-ultimate-guide-to-designing-a-robot-conversacional-tech-stack-333eceb431da> (20 de octubre de 2020)
- [3.] Ernesto Guerra, Amanda, el robot conversacional que sabe de elecciones en Cuba. <https://planetarobotconversacional.com/amanda-el-robot-conversacional-que-sabe-de-elecciones-en-cuba-e2c3b5a7d835>
- [4.] *Universitari_Bots*. (2020, septiembre 18). Cubadebate. http://www.cubadebate.cu/especiales/2020/09/18/universitari_bots/ 31-10-2022
- [5.] Rosbel Caballero Ramírez. (2021). Chatbot: Una propuesta viable para la atención al cliente en el centro de soporte de la UCI.
- [6.] <https://azure.microsoft.com/en-us/services/bot-services/> 4-11-2021
- [7.] <https://docs.microsoft.com/en-us/composer/introduction> 5-11-2021
- [8.] <https://paginapropia.com/15-ventajas-y-desventajas-de-microsoft-azure/> 6-11-2021
- [9.] <https://stackshare.io/stackups/dialogflow-vs-microsoft-bot-framework/> 5-11-2021
- [10.] <https://rijsat.com/2021/11/03/create-azure-bot-service-from-azure-portal/> 6-11-2021
- [11.] <https://cloud.google.com/dialogflow/docs> 4-11-2021
- [12.] <https://cloud.google.com/dialogflow?hl=es> 5-11-2021
- [13.] <https://www.robotconversacionales.org/dialogflow> 6-11-2021
- [14.] https://www.capterra.es/reviews/180853/dialogflow?overall_rating_ge=5 6-11-2021
- [15.] <https://www.linkedin.com/pulse/dialogflow-vs-lex-watson-azure-bot-robot-conversacional-quick-sherwin-fernandes> 5-11-2021
- [16.] <https://rasa.com/> 3-11-2021
- [17.] <https://www.spaceo.ca/ai-robot-conversacional-development-using-rasa-reasons/> 3-11-2021
- [18.] <https://www.robotconversacionales.org/best-robot-conversacional-builders#builder5> 6-11-2021
- [19.] <https://botpress.com/docs/introduction> 3-11-2021
- [20.] <https://www.getapp.es/reviews/2047766/botpress> 4-11-2021
- [21.] <https://www.saashub.com/compare> 5-11-2021
- [22.] <https://www.spaceo.ca/top-ai-robot-conversacional-marcos/> 7-11-2021
- [23.] Waghmare, C. (2019). Introducing Azure Bot Service Building Bots for Business.

- [24.] Navin Sabharwal, A. A. (2020). Cognitive Virtual Assistants Using Google Dialogflow Develop Complex Cognitive Bots Using the Google Dialogflow Platform.
- [25.] Raj, S. (2019). Building Robot conversacionales with Python Using Natural Language Processing and Machine Learning.
- [26.] Álvaro Castillo Cabero, P. P. M., Joan Antoni Pastor Collado. (2020). Rasa Framework: Análisis e implementación de un Robot conversacional.
- [27.] Tri Chau Minh Tri, N. T. D., Koh Wee Lit (2020). Creating smart, human-like robot conversacional for businesses using BotPress platform.
- [28.] Papangelis, A., Namazifar, M., Khatri, C., Wang, Y.-C., Molino, P., & Tur, G. (2020). *Plato Dialogue System: A Flexible Conversational AI Research Platform*.
- [29.] <https://es.wikipedia.org/wiki/Python> 11-2-2022
- [30.] <https://es.wikipedia.org/wiki/PyCharm> 11-2-2022
- [31.] [NoSQL - Wikipedia, la enciclopedia libre](#) 11-2-2022
- [32.] [YAML - Wikipedia, la enciclopedia libre](#) 11-2-2022
- [33.] XP - Extreme Programing Ingenieria de Software. [XP - Extreme Programing Ingenieria de Software \(mex.tl\)](#) 15-6-2021
- [34.] Puntos de caso de uso. (2021). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Puntos_de_caso_de_uso&oldid=133055102 20-9-2022

Anexos

En este apartado se tocará la instalación y uso del sistema. Mostrando ejemplos de su funcionamiento.

Instalación

- Debe tener conexión a internet.
- Debe tener Python 3.8 instalado.
- Debe tener MongoDB instalado.
- Ejecutar el archivo **"Instalar requerimientos.bat"**, una vez ejecutado debe estar conectado a internet para que comience el proceso de instalación de las dependencias, sin ellas no funcionaría.

Iniciando el sistema

- Se ejecuta en modo administrador el archivo **"systemSGCA.bat"**
- Una vez iniciado podrá registrarse e iniciar sesión, todo esto mediante un nombre de usuario y una contraseña.

Registro e inicio de sesión y funciones disponibles una vez dentro

<pre>OPCIONES 1. Registrarse 2. Iniciar Sesión Entre el número de la opción: 1 REGISTRARSE EN EL SISTEMA. Cree su nombre de usuario : user Cree su contraseña : user0000 INICIAR SESIÓN PARA USAR LA APLICACIÓN Entre su nombre de usuario : user Entre su contraseña : █</pre>	<pre>OPCIONES 1. Crear Asistente Virtual 2. Generar Conocimiento 3. Entrenar Asistente 4. Probar Asistente Entre el número de la opción: █</pre>
---	---

A continuación, se transitará por las opciones del sistema:

Opción 1. Crear Asistente Virtual

Entrada: La dirección de la carpeta donde se quiere guardar el asistente virtual al crearse.

```
Rasa le hara una primera pregunta a la que debe responder presionando la tecla ENTER

Rasa le hara una segunda pregunta a la que debe responder presionando la tecla N y luego ENTER

Welcome to Rasa! 🦊

To get started quickly, an initial project will be created.
If you need some help, check out the documentation at https://rasa.com/docs/rasa.
Now let's start! 🦊

? Please enter a path where the project will be created [default: current directory]
Created project directory at 'D:\DOCUMENTOS\prueba-rasa'.
Finished creating project structure.
? Do you want to train an initial model? 🦊 No
No problem 🦊. You can also train a model later by going to the project directory and running 'rasa train'.
```

Inicia la plataforma Rasa

actions	21/10/2022 14:53	Carpeta de archivos
data	21/10/2022 14:53	Carpeta de archivos
models	25/10/2022 16:47	Carpeta de archivos
tests	21/10/2022 14:53	Carpeta de archivos
config	25/10/2022 16:47	Archivo de origen ...
credentials	21/10/2022 16:14	Archivo de origen ...
domain	21/10/2022 16:13	Archivo de origen ...
endpoints	16/9/2022 16:26	Archivo de origen ...
graph	21/10/2022 16:18	Chromium HTML ...

Ficheros del asistente virtual creado en la ubicación dada.

Opción 2. Generar Conocimiento

Es la parte más importante del sistema que dada una entrada de datos, estos automáticamente serán analizados y se extraerán preguntas y respuestas relacionadas con el contenido.

En un principio los datos entrados sólo debían ser en inglés para luego ser traducidos al español, debido a que el algoritmo (aunque fue modificado un poco para integrarlo al sistema) pertenecía a un desarrollador externo. Este algoritmo, además, no generaba respuestas, sino que sólo generaba preguntas de formato ¿Qué es? y se optó por tomar las oraciones desde donde sale la pregunta como respuesta.

```
-----INPUT TEXT-----  
  
Criptomonedas is a coin-virtual for ensure transactions.  
Bitcoin is a decentralized cryptodivisa.  
Digital-Wallet is a application where it is possible to store, send and receive cryptomoneds.  
The Intelligent-City is a type of urban development based on the sustainability for the basic needs of institutions, companies, and of the inhabitants themselves.  
  
-----INPUT END-----  
  
Question: What is Criptomonedas?  
  
Question: What is Bitcoin?  
  
Question: What is Digital-Wallet?  
  
Question: What is Intelligent-City?
```

Entrada de un texto en inglés y las preguntas generadas del contenido en el mismo idioma.

Algoritmos implementados para idioma español

Importante saber para comprender el funcionamiento.

Similitud Coseno:

Similitud de coseno, consiste en evaluar la similitud de dos vectores calculando el coseno del ángulo entre ellos. La similitud de coseno dibuja vectores en el espacio vectorial de acuerdo con valores de coordenadas, como el espacio bidimensional más común.

La similitud (por ejemplo, entre dos o más textos y oraciones) se determina comparando vectores de palabras o “incrustaciones de palabras”, representaciones de significado multidimensional de una palabra.

Los vectores de palabras se pueden generar usando un algoritmo como word2vec que usualmente se verían así:

```
array([ 2.02280000e-01, -7.66180009e-02,  3.70319992e-01,  
       3.28450017e-02, -4.19569999e-01,  7.20689967e-02,  
      -3.74760002e-01,  5.74599989e-02, -1.24009997e-02,  
       5.29489994e-01, -5.23800015e-01, -1.97710007e-01, ])
```

Word2vec representa cada palabra distinta con una lista particular de números llamada vector. Los vectores están escogidos cuidadosamente de forma que una función matemática sencilla (la similitud coseno entre los vectores) indica el nivel de la similitud semántica entre las palabras representada por dichos vectores.

Tomado de:

<https://spacy.io/usage/linguistic-features#vectors-similarity>

<https://es.wikipedia.org/wiki/Word2vec>

Decir que la similitud coseno en algunas circunstancias no es exacta del todo (puede deberse a la forma o el funcionamiento que se implementó a la hora de hallarla, como se construyen los vectores y la información a procesar que contengan los mismos), algunos métodos que permiten calcularla pueden dar mejores resultados que otros, esto en algunas ocasiones puede variar.

Entidad nombrada:

Entidad nombrada es un objeto del mundo real, como una persona, ubicación, organización, producto, evento, etc., que se puede denotar con un nombre propio.

Token en textos:

Los tokens son unidades pequeñas significativas formadas al dividir un texto en partes más pequeñas (Tokenización). Ejemplos de tokens pueden ser palabras, caracteres, números, símbolos o n-gramas. Y de esos tokens sacar sus propiedades gramaticales (verbos, formas verbales, sustantivos, sujeto, etc.)

Texto de ejemplo usado:

Código de las Familias es una norma sustantiva del Derecho de Familia en Cuba; Cuerpo legal que regula todas las instituciones relativas a la familia: el matrimonio, el divorcio, las relaciones paterno filiales, la obligación de dar alimentos, la adopción y la tutela. Se promulga en 1975, modificado y propuesto a referendo popular el 25 de septiembre de 2022, siendo ratificado por el pueblo cubano con el 66.87 % de los votos.

Las normas contenidas en este Código se aplican a todas las familias cualquiera que sea la forma de organización que adopten y a las relaciones jurídico-familiares que de ellas se deriven entre sus miembros, y de estos con la sociedad y el Estado y se rigen por los principios, valores y reglas contenidos en la

Constitución de la República de Cuba, los tratados internacionales en vigor para el país que tienen incidencia en materia familiar y los previstos en este Código.

Es un Código inclusivo, revolucionario y novedoso en su texto como en su proceso de elaboración. Protege a niños, niñas y adolescentes, les reconoce derechos a las personas adultas mayores y en situación de discapacidad, visibiliza y reconoce derechos a sectores vulnerables.

Primera versión implementada del algoritmo

Elementos usados:

- Spacy (Librería para el procesamiento del lenguaje natural)
- Modelo pre-entrenado para generar respuestas. Enlace: <https://hugging-face.co/mrm8488/electricidad-small-finetuned-squadv1-es>
- Modelo pre-entrenado para generar preguntas. Enlace: <https://hugging-face.co/mrm8488/bert2bert-spanish-question-generation>
- Código base original de tercero. Enlace: <https://medium.com/featurepre-neur/question-generator-d21265c0648f>

En esta versión básicamente una vez que se entra el texto:

- Se divide el texto en oraciones.
- Luego se carga el modelo de generar preguntas (Son modelos implementados que fueron entrenados con grandes cantidades de datos para formular preguntas a partir de dichos datos) y se le envían las oraciones una por una dando como resultado una pregunta para cada oración.
- Cada pregunta y la oración de donde sale dicha pregunta que se analiza en el momento son enviadas al modelo que genera las respuestas, dando como resultado respuestas a dichas preguntas.

SALIDA

¿Cuál es el nombre oficial del Código de las Familias?

{'score': 0.15141291916370392, 'start': 0, 'end': 22, '**respuesta': 'Código de las Familias'**'}

¿Qué votó el pueblo cubano en contra de la enmienda?

{'score': 0.3791300356388092, 'start': 139, 'end': 146, '**respuesta': '66.87 %'**'}

El Código de la República de Cuba está sujeto a ¿qué tipo de tratado?

{'score': 0.399568110704422, 'start': 354, 'end': 378, '**respuesta ': 'tratados internacionales'**'}

¿Cuál es la definición de un código inclusivo en su conjunto?

{'score': 0.2582134008407593, 'start': 24, 'end': 49, '**respuesta': 'revolucionario y novedoso'**'}

¿Qué hace la visibilización de la gente en Cuba?

{'score': 0.3266850709915161, 'start': 137, 'end': 177, '**'respuesta': 'reconoce derechos a sectores vulnerables'**'}

En este ejemplo las preguntas generadas algunas no tienen todo el sentido si se mira el texto, por ejemplo, la segunda pregunta: ¿Qué votó el pueblo cubano en contra de la enmienda? (*Aunque respondió con el porcentaje de votos, en realidad es voto a favor y no es una enmienda*). **Con otros textos tiene mejores resultados, aunque no del todo, puede variar, como puede pasar con la próxima versión que veremos.**

Segunda versión

Elementos usados:

- Spacy
- Modelo pre-entrenado para generar respuestas.
- Modelo pre-entrenado para analizar similitud entre oraciones. Enlace: <https://huggingface.co/eduardofv/stsb-m-mt-es-distilbert-base-uncased>
- Código base original de terceros. Enlace: <https://towardsdatascience.com/semantic-similarity-using-transformers-8f3cb5bf66d6>

En esta versión básicamente una vez que se entra el texto:

- Se divide el texto en oraciones.
- Se hace un análisis de las oraciones en busca de tokens (palabras con categoría gramatical como verbos, formas verbales, sustantivos, ect.)
- Luego se usa Spacy para hacer un procesamiento de las oraciones, extraer entidades nombradas y generar preguntas de acuerdo a cada entidad.

Extraemos las entidades y generamos las preguntas:

(Esta forma es bastante parecida a la que usa el algoritmo para generar preguntas mencionado al principio que funciona con idioma inglés)

Por ejemplo, entidad **“PER”** para las personas; **“ORG”** para organizaciones; **“LOC”** para lugares, localidades, ciudades; **“MISC”** (Misceláneas) es para varios tipos como eventos, nacionalidades, productos, etc.

Para generar la pregunta por ejemplo si es una persona:

¿Quién es ___? el espacio que queda después de “es” lo sustituimos por la persona extraída. (generalmente el nombre de la persona)

Si es para una localidad:

¿Dónde queda ___? añadimos la localidad extraída en el espacio después de “queda”.

En el segundo paso tokenizamos para extraer verbos, formas verbales e incluso complementos y otras categorías gramaticales de las oraciones, éstas se usan para generar preguntas que lleven la palabra o frase con esa categoría gramatical.

Por ejemplo, si se habla de una persona y hay una forma verbal esta casi siempre significa la acción que realizó dicha persona, por lo que una pregunta sería en caso de ejemplo:

Pablo participó en un evento para desarrolladores web.

Pregunta generada: ¿Quién participó en el evento para desarrolladores web?

- Una vez que tenemos la lista de preguntas, dicha lista es recorrida a través de un ciclo enviando cada pregunta al modelo que analiza la similitud coseno entre oraciones (Spacy también puede analizar la similitud entre oraciones). Aquí se analiza la similitud de cada pregunta con las oraciones para saber con exactitud que oración es la que le corresponde por lo que se crea un ranking decreciente donde tenemos como primera candidata a la oración con mayor puntaje que sería desde donde se generó la pregunta.

Además, agregar que este análisis de la similitud se hace ya que hay preguntas que se generan que no tienen respuesta en el texto, por lo que si una pregunta no tiene similitud suficiente con la oración quiere decir que dicha oración tiene altas probabilidades de no contener la respuesta a dicha pregunta.

- Una vez que tenemos la las preguntas y las oraciones más similares a estas, son enviadas (preguntas y respuestas) a través de un ciclo al modelo que genera las respuestas, dando como resultado respuestas a dichas preguntas.

Salida

La primera pregunta coincide con la primera respuesta y así sucesivamente.

Preguntas: [
'Dónde queda o se localiza Estado', '
Qué es Derecho de Familia',
'Qué es Código',
'Qué es Constitución de la República de Cuba',
'Qué es Código de las Familias',
'Dónde queda o se localiza Cuba',
'Qué es Cuba',
'Qué es Estado',
'Qué paso en o con Cuerpo']

Respuestas: [
'la sociedad y el Estado',
'Código de las Familias',
'proceso de elaboración',
'Código de las Familias',
'una norma sustantiva del Derecho de Familia',
'con la sociedad y el Estado',
'ratificado por el pueblo cubano',
'la sociedad y el Estado',
'la sociedad y el Estado']

Los resultados arrojados por este método, al menos respecto al texto de ejemplo no tuvieron mucha correspondencia las preguntas y las respuestas, ya que aquí las preguntas generadas son muy específicas o cerradas a entidades dando como resultado que en ocasiones se alejen del contenido del texto y por ende no tener respuestas en dicho texto, independientemente de esto el modelo como mínimo siempre da una respuesta. Con otros textos donde las oraciones contenidas en el sean más concretas en la información que dan tiene mejores resultados, por ejemplo:

Texto

Criptomoneda es una moneda-virtual para asegurar transacciones. Bitcoin es una criptodivisa descentralizada. Digital-Wallet es una aplicación donde es posible almacenar, enviar y recibir criptomonedas. La Ciudad-Inteligente es un tipo de desarrollo urbano basado en la sostenibilidad para las necesidades básicas de instituciones, empresas y de los propios habitantes.

Salida

Preguntas: ['Qué es Digital Wallet', 'Qué es Una Criptomoneda', 'Dónde queda o se localiza Ciudad Inteligente', 'Qué es Ciudad Inteligente', 'Qué es Bitcoin']

Respuestas: ['posible almacenar, enviar y recibir criptomonedas', 'una moneda virtual para asegurar transacciones', 'desarrollo urbano', 'desarrollo urbano', 'criptodivisa descentralizada']

En este ejemplo hay mucha más correspondencia entre las preguntas y las respuestas, excepto la pregunta de dónde queda Ciudad Inteligente.

Opción 3. Entrenar Asistente Virtual

Una vez que tenemos las preguntas y las respuestas generadas guardadas en la base de datos y además el asistente virtual creado, procedemos a cargar los datos que queremos de la base de datos y también el asistente virtual que queremos entrenar.

Vista de algunos archivos de entrenamiento creados automáticamente para el asistente virtual con las preguntas y respuestas extraídas.

```

version: '3.1'
intents:
  - saludar
  - despedir
  - afirmar
  - negar
  - animo
  - no_animo
  - bot
  - fuera_contexto
  - ¿Qué es Criptomoneda?
  - ¿Qué es Bitcoin?
  - ¿Qué es Monedero Digital?
  - ¿Qué es Ciudad-Inteligente?
responses:
  utter_saludar:
    - text: ¿Cómo estás?
  utter_ayuda:
    - text: ¿En qué puedo ayudarte?
  utter_feliz:
    - text: ¡Genial, continúa!
  utter_adios:
    - text: Adiós, puedes consultarme cuando quieras
  utter_soybot:
    - text: Soy un bot, creado por Rasa.
  utter_fuera_contexto:
    - text: Lo siento, no puedo entender o manejar lo que acabas de decir.
  utter_¿Qué es Criptomoneda?:
    - text: Criptomoneda es una moneda virtual para asegurar transacciones.
  utter_¿Qué es Bitcoin?:
    - text: Bitcoin es una criptodivisa descentralizada.
  utter_¿Qué es Monedero Digital?:
    - text: Digital-Wallet es una aplicación donde es posible almacenar, enviar y recibir criptomonedas.

```

Archivo domain.yml

```

version: '3.1'
nlu:
  - intent: saludar
    examples: |
      - Hola
      - que tal
      - buenos días
      - buenas noches
      - buenas tardes
  - intent: despedir
    examples: |
      - adiós
      - buenas noches
      - hasta luego
  - intent: afirmar
    examples: |
      - Sí
      - por supuesto
      - correcto
  - intent: negar
    examples: |
      - no
      - nunca
      - de ninguna manera
      - no realmente
  - intent: animo
    examples: |
      - perfecto
      - genial
      - increíble
      - Me siento muy bien
      - Estoy genial
      - Estoy increíble
      - Estoy bien

```

Archivo nlu.yml

