

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:** Dr. Dionis López Ramos.

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 y la búsqueda de información. 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 7](#_Toc116232217)

[1.1 Las plataformas para desarrollar robots conversacionales 7](#_Toc116232218)

[1.2 Breve historia de los agentes conversacionales 7](#_Toc116232219)

[1.2.1 Estado del arte de los marcos para desarrollar robot conversacional 8](#_Toc116232220)

[Azure Bot Service 8](#_Toc116232221)

[DialogFlow 9](#_Toc116232222)

[Rasa 11](#_Toc116232223)

[BotPress 12](#_Toc116232224)

[Conclusión 15](#_Toc116232225)

[1.3 Herramientas, Lenguajes de programación y Tecnologías 16](#_Toc116232226)

[Python 16](#_Toc116232227)

[PyCharm Community 16](#_Toc116232228)

[RASA 16](#_Toc116232229)

[MongoDB 17](#_Toc116232230)

[YAML 17](#_Toc116232231)

[GIT 17](#_Toc116232232)

[Moon Modeler 18](#_Toc116232233)

[1.4 Metodología de desarrollo de software 19](#_Toc116232234)

[Conclusiones del capítulo 20](#_Toc116232235)

[Capítulo 2. Planificación y Diseño 21](#_Toc116232236)

[2.1 Propuesta del Sistema 21](#_Toc116232237)

[2.2 Usuarios del Sistema 22](#_Toc116232238)

[2.3 Planificación del desarrollo del Sistema 22](#_Toc116232239)

[2.3.1. Historias de usuarios 22](#_Toc116232240)

[2.3.2. Requisitos Funcionales 22](#_Toc116232241)

[2.3.3. Requisitos no Funcionales 23](#_Toc116232242)

[2.3.4. Arquitectura del Sistema 28](#_Toc116232243)

[2.3.5. Diseño de Base de Batos 29](#_Toc116232244)

[Conclusiones del capítulo 30](#_Toc116232245)

[3.1. Instalación de las herramientas utilizadas 31](#_Toc116232246)

[3.2 Implementación de Funcionalidades 32](#_Toc116232247)

[3.3 Análisis económico 38](#_Toc116232248)

[3.3.1. Estimación de costo y tiempo 38](#_Toc116232249)

[3.4 Pruebas al sistema 44](#_Toc116232250)

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 front-end para responder a preguntas frecuentes (FAQs).

Luego están los agentes conversacionales construidos 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 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. [3]

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) [4] y ELIZ que es un asistente virtual para la plataforma ENZONA. 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.

**Problema de la investigación:**

El servicio agente de atención al cliente a través de personas supone costos, gasto físico y muchas veces limitaciones en el servicio, ya sea por el horario laboral, problemas de salud o falta de personal capacitado para satisfacer la demanda que de búsqueda de información que generan las personas. **Un asistente virtual que puede estar disponible a toda hora, atender a muchos usuarios a la vez con una buena base de conocimiento que actualmente es insuficiente, necesita de especialistas y de un laborioso trabajo para construirse, puede dar solución a estas cuestiones o necesidades.**

**Objeto de Estudio:** La gestión automatizada de 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 desde las plataformas web o móvil donde se use el servicio de chat, estas plataformas se favorecerán al ganar en calidad en el servicio.

# **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](https://www.teneo.ai/); 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.

**Prueba de Turing, 1950**

La prueba de Turing plantea la pregunta de si las máquinas pueden pensar, y fue formulada por Alan Turing en su escrito “Computing Machinery and Intelligence” (Maquinaria de computación e inteligencia) publicado en 1950. En el artículo, Turing propuso una prueba en la que un interrogador debía determinar qué jugador era un ser humano y cuál era una máquina, mediante una serie de preguntas escritas. [1]

A pesar de las críticas y los defectos, la prueba se sigue realizando regularmente en la actualidad.

**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. Alice ha ganado 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] [11] [10]

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. [6] [11]

Más de 18 empresas usan la tecnología de Microsoft en sus sistemas de trabajo como por ejemplo LUMENEO, Autonom8 y WBOT. [6] [11]

***Azure Bot Service*** proporciona plataformas para integrar sus agentes conversacionales: Slack, Skype, API Telegram, son algunas de las herramientas populares que se integran con **Azure Bot Service**. [6] [11]

***Ventajas***

* IA y lenguaje natural.
* Código Abierto y extensible (Bot Framework Composer).
* Soluciones de nivel empresarial.
* Propiedad y control.

***Desventajas*** [6] [10]

* 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 en el mercado*** [6] [10]

**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 comunique 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. [7] [12] [13] [10]

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. [7] [12] [13]

***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. [7] [12] [13]

**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** [7] [10]

* 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. [8] [13] [14]

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. [8] [13] [14]

Hay varias compañías que usan Rasa y algunas de ellas son ERGO, Orange, Lemonade y T-Mobile. [8] [13] [14]

**Características** [8] [13] [14] [10]

* 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. [9] [15] [10]

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. [9] [15]

**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. [9] [10]

**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** [9] [10]

* 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. [11]

**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 (NLP), como análisis de sentimientos, modelado de temas, seguimiento de estado de diálogo, generación de lenguaje social y otros. [11]

**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**

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 desarrollo creemos 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 en 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 busca una buena estructura para un gran proyecto, pero si está comenzando y tiene poca experiencia la mejor es DialogFlow.

Para el trabajo de investigación se contará con la plataforma Rasa, ya que por su forma de trabajo permite modificar o configurar a nivel de archivos cómo se lleva el conocimiento a los asistentes virtuales desarrollados en ella. Además, Rasa se ajusta más a la línea o idea que se está siguiendo en este trabajo por las herramientas que tiene, su comunidad activa y documentación, así como las facilidades que brinda para desarrollar y entrenar sus agentes conversacionales.

## **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](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [interpretado](https://es.wikipedia.org/wiki/Int%C3%A9rprete_(inform%C3%A1tica)) cuya filosofía hace hincapié en una [sintaxis](https://es.wikipedia.org/wiki/Sintaxis) que favorezca un código legible. Se trata de un lenguaje de programación [multiparadigma](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_multiparadigma), ya que soporta [orientación a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos), [programación imperativa](https://es.wikipedia.org/wiki/Programaci%C3%B3n_imperativa) y, en menor medida, programación funcional. Es un [lenguaje interpretado](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_interpretado), usa [tipado dinámico](https://es.wikipedia.org/wiki/Tipado_din%C3%A1mico) y es [multiplataforma](https://es.wikipedia.org/wiki/Multiplataforma). Posee una licencia de [código abierto](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto), denominada [Python Software Fundation License](https://es.wikipedia.org/wiki/Python_Software_Foundation_License) que es compatible con la [Licencia pública general de GNU](https://es.wikipedia.org/wiki/GNU_General_Public_License) [17]. Se utilizó para la implementación de la herramienta informática.

### **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 [18]. La Edición Community es la versión libre y se publica bajo la Licencia de Apache.

### **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.

(*Ver* [*Rasa*](#_Rasa) *en el Estado del Arte de los marcos para desarrollar AV*).

### **MongoDB**

MongoDB es un sistema de base de datos NoSQL, orientado a documentos y de código abierto. [19]

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) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

MongoDB es una base de datos adecuada para su uso en producción y con múltiples funcionalidades. Esta base de datos es utilizada por compañías como, por ejemplo: FourSquare, Facebook, Ebay, Google, etc.

### **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. [20]

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.

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.

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).

### **Moon Modeler**

Moon Modeler es una herramienta de modelado de datos para MongoDB principalmente, pero además para PostgreSQL, MariaDB, SQLite y GraphQL. Le permite dibujar modelos de datos de forma rápida y cómoda, crear diagramas Entidad-Relación para bases de datos, diseñar estructuras anidadas, hacer documentación de diseño de esquemas, ingeniería inversa, generar scripts y más. **[21]**

Es un sistema de pago, la versión libre sólo puede ser probada por 14 días con todas las funciones y algunas con limitaciones, pero las funciones básicas están bien cubiertas. Permite conexiones a base de datos existentes y a partir de ahí generar diagramas, relaciones existentes de la base de datos conectada, generar código correspondiente a su modelo, e incluso un reporte basado en su modelo de base de datos; todo esto resulta de mucha ayuda a la hora de trabajar o construir su modelo de base de datos en poco tiempo.

Se usó la versión libre de Moon Modeler para realizar el diagrama de base de datos del sistema.

## **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 realimentación 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. **[22]**

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 testeos 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**

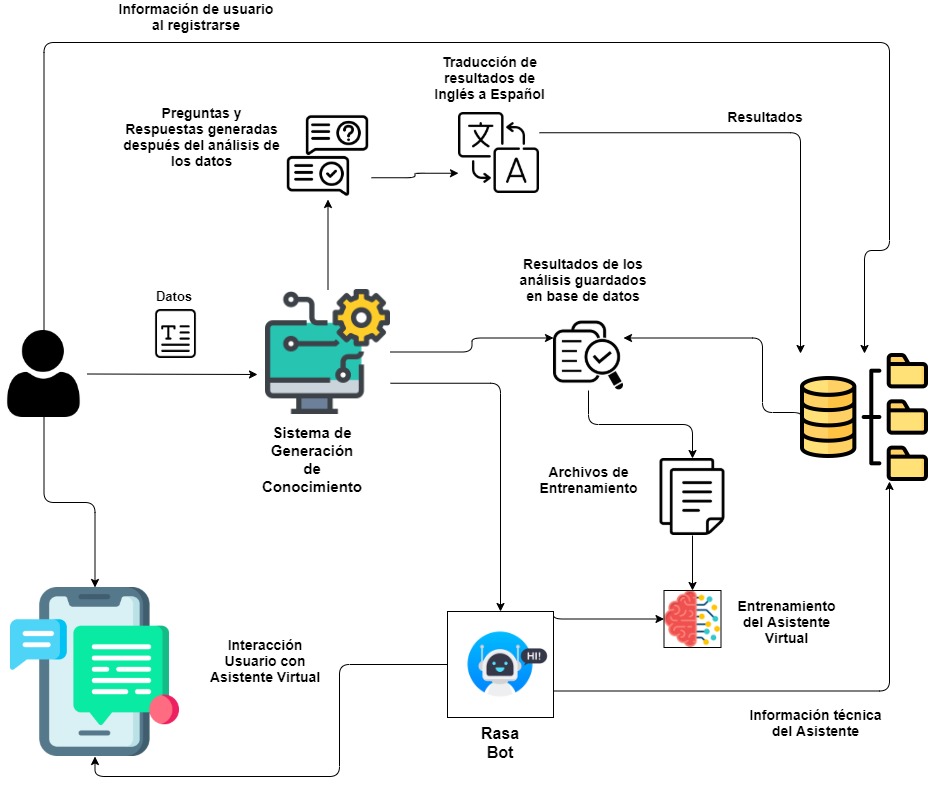
Se definieron los elementos teóricos-prácticos que ayudaron a conformar el modelo conceptual del proyecto, los distintos factores y elementos que fueron objeto de estudio para la realización del mismo, así como la metodología de desarrollo de software. Se analizaron las herramientas disponibles para ejecutar dichas tareas y se determinaron cuales se adaptan mejor para la implementación del sistema y por qué.

# **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. Historias de usuarios**

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.

### **2.3.2. 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 conocimiento del Asistente: Es donde el usuario a partir de una entrada de datos al sistema, este último analizará dichos datos y creará los archivos de entrenamiento.

Entrada: Datos.

Salida: Archivos de entrenamiento.

**RF3:** 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.

**RF4:** Entrenar Asistente y probarlo: Permitirá una vez creado el asistente y los archivos de entrenamiento poder entrenarlo, y una vez finalizado este proceso tendrá la posibilidad de probarlo e interactuar con el robot conversacional.

Entrada: Información pedida por el sistema.

Salida: Se entrena el asistente virtual y se inicia la interacción de prueba.

### **2.3.3. Requisitos no Funcionales**

**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.

**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 Corei3 a 2.3 GHz
* 4.0 GB de memoria RAM
* Buena conexión a Internet

**A continuación, se hace una lista de las historias de usuario de la aplicación:**

* Autenticación del Sistema.
* Crear Asistente Virtual
* Gestión de Conocimiento.
* Entrenar Asistente Virtual.

**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.** | |
| **Propuesta de interfaz:**  D:\DATOS ERNESTO\ING INFORMATICA\TESIS NETO\Documentos extras\Propuestas de interfaz\Login.png | |

**Tabla 2.3 Historia de Usuario: Crear Asistente Virtual**

|  |  |
| --- | --- |
| **Historia de usuario** | |
| **Número:2** | **Nombre de HU:** Crear Asistente Virtual |
| **Usuario:** Cualquier usuario | |
| **Prioridad en negocio:** Alta | **Riesgo de desarrollo:** Bajo |
| **Descripción: Donde el usuario crea su Asistente Virtual con el que podrá interactuar.** | |
| **Observaciones:** Ninguna | |
| **Propuesta de interfaz:**  **D:\DATOS ERNESTO\ING INFORMATICA\TESIS NETO\Documentos extras\Propuestas de interfaz\Crear Asistente.png** | |

**Tabla 2.4 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.**  Operaciones:   1. Cargar archivos con los datos (Archivo de texto). 2. Análisis de contenido para generar preguntas/respuestas. 3. Generar archivos de entrenamiento. 4. Guardar información y resultados. 5. Cargar información y resultados ya analizados.   *Una vez que se generen los archivos se mostraría un mensaje con la opción de entrenar Asistente Virtual.* | |
| **Observaciones:** Ninguna | |
| **Propuesta de interfaz:**  **D:\DATOS ERNESTO\ING INFORMATICA\TESIS NETO\Documentos extras\Pagina principal.png** | |

**Tabla 2.5 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 y probar su Asistente Virtual al interactuar con el mismo.** | |
| **Observaciones:** Ninguna | |
| **Propuesta de interfaz:** | |

### **2.3.4. Arquitectura del Sistema**

Para el desarrollo del sistema fue utilizada la siguiente arquitectura:

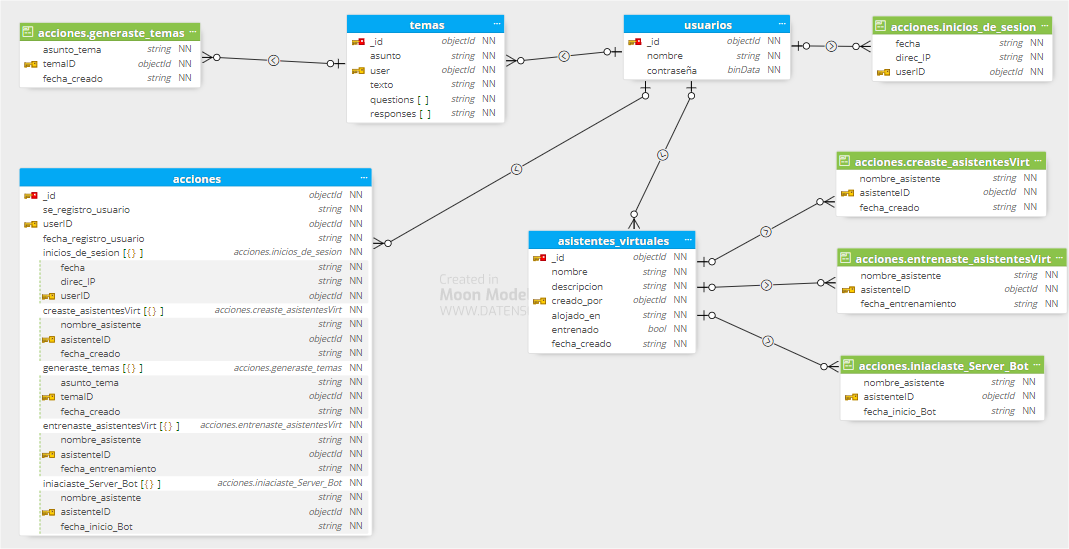


**Figura 2.2** **Arquitectura del Sistema**

* 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 guardados 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.

### **2.3.5. Diseño de Base de Batos**

El sistema cuenta con una base de datos no relacional (noSQL) que está orientada a documentos (Están diseñadas para almacenar, recuperar y administrar datos orientados a documentos o semiestructurados, capaz de manejar conjuntos de datos de longitud no fija) ya que la herramienta de Gestión de base de Datos usada es MongoDB. Se necesita 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. Contamos con cuatro colecciones (como las tablas y sus columnas en es los sistemas SQL) las cuáles contienen documentos (como las filas de las tablas en es los sistemas SQL) y estos a su vez contienen los valores de las claves o atributos contenidos en las colecciones. **(clave -> valor**; ya que usa documentos en formato BSON (*JSON Binario, es un formato de intercambio de datos usado principalmente para su almacenamiento y transferencia en la base de datos MongoDB,* *diseñado para tener un almacenamiento y velocidad más eficiente y soportar más tipos de datos que el formato JSON del cual proviene*)**)**.



**Figura 2.3 Diagrama de Base de Datos**

**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). Un usuario en este caso puede tener muchos asistentes virtuales, temas y realizar varias acciones.
* **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 se quedan registradas las acciones del usuario al interactuar con sistema (registro de usuario, inicios de sesión, operaciones respecto a las funcionalidades de la herramienta). Esta colección tendrá además de documentos, los documentos embebidos (*MongoDB permite tener documentos embebidos (documentos dentro de otros documentos)*, en este caso estos tienen el ID del documento al que están relacionados**,** dichos documentos embebidos están de color verde en la **Figura 2.3**). En este caso la lectura sería, por ejemplo, que con asistentes virtuales se realizan varias acciones (crear, entrenar, iniciar servidor de un asistente), dichas acciones que son realizadas por el usuario se quedan registradas dentro de la colección Acciones.

**Documentos en las colecciones**

* **Los documentos se van insertando a medida que el usuario guarda información en el sistema, pues al registrarse un usuario, por ejemplo, se crea un documento con los datos de dicho usuario y así sucesivamente ocurre con los demás datos que deben guardarse en la base de datos. Mientras no haya un documento definido las colecciones no se crearán.**

**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 y hora en el que el usuario inicia sesión en el sistema.
* Además de la información anterior, se registrará la IP de la computadora desde donde se inicia la sesión.
* También se registran las acciones que realiza en usuario al transitar por las funcionalidades del sistema.

## **Conclusiones del capítulo**

En este capítulo se describió el funcionamiento del sistema, haciendo énfasis en las etapas de planificación y diseño de la metodología XP. Se analizaron los elementos que caracterizan o definen el proceso. Además, se presenta una visión general de la propuesta del sistema a desarrollar, y, por último, se exponen los elementos del diseño de la base de datos que se implementa.

**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.**)

***-*** *Para instalar todas las dependencias (Incluido el Rasa) de la herramienta de una vez. Ejecutar el siguiente archivo (Requiere Internet).*

Instalar requerimientos Python.bat

1. Instalación de MongoDB: Ejecutar Instalador y seguir los pasos de instalación hasta el final sin cambiar nada en el proceso.
2. Instalación del PyCharm Community Edition: Ejecutar el instalador, luego lo utilizamos para la programación en el lenguaje Python.
3. 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.
4. Se implementaron las funciones para autenticación de usuario.
5. Se buscó una solución de terceros para la generación de preguntas y respuestas a partir de una entrada de datos. Se modificó algunas partes de su código para ajustarlo a la forma de funcionamiento de la herramienta que estamos implementando. Por algunas limitaciones de la solución encontrada los datos a analizar deben ser en inglés.
6. Usando la biblioteca *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.
7. Luego se crearon las funciones para los procesos de creación, entrenamiento y prueba del Asistente Virtual.

## **3.2 Implementación de Funcionalidades**

Primeramente, abrir la herramienta al ejecutar el siguiente archivo como administrador (Requiere Internet):

systemSGCA.bat

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



**Figura 3.1** Función para registrarse en el sistema.



**Figura 3.2** Función para iniciar sesión en el sistema.

**A continuación, se muestran algunas de las funciones más importantes cuando inicia el proceso principal: Crear el conocimiento para el Asistente Virtual.**



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

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.)



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

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)



**Figura 3.5** Función para crear el Asistente Virtual



**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.)



**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 **[23]**. 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

**UUCP = UAW + 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](#_Tabla_3.1_Factor) 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)

#### 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 | 3\*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 |  |  | 15+10+0=**25** |

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

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

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

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

Donde:

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

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

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante o nulo y 5 un aporte muy importante. En la siguiente [tabla 3.3](#_Tabla_3.3_Factor) se muestra factor de complejidad técnica con su significado y el peso de cada uno de estos factores:

#### Tabla 3.3 Factor de complejidad técnica

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Factor | Descripción | **Peso** | **Valor** | **(Peso-i \* Valor-i)** |
| T1 | Sistema distribuido | 2 | 0 | 0 |
| 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 | 0 | 0 |
| T13 | Se requieren facilidades especiales de entrenamiento a los usuarios | 1 | 2 | 2 |
| Total |  |  |  | **39** |

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

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

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

**TCF = 0.6 + 0.01 \*39**

**TCF = 0.99**

**Factor Ambiente (EF).**

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

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

#### Tabla 3.4 Factor de ambiente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Factor** | **Descripción** | **Peso** | **Valor** | **(Peso-i \* Valor-i)** |
| E1 | Familiaridad con el modelo de proyecto utilizado | 1.5 | 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 | 1.5 | -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

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

**EF** = 1.4 - 0.03 \* 22.7

**EF** = 0.72

**Luego: UCP=UUCP \* TCF \* EF**

**UCP = 32 \* 0.99 \* 0.72**

**UCP = 23**

**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 (**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 horas-hombre**

**E = 460 horas-hombre**

En la siguiente [tabla 3.5](#_Tabla_3.5_Distribución) se muestra distribución del esfuerzo por etapas.

#### Tabla 3.5 Distribución del esfuerzo por etapas

|  |  |  |
| --- | --- | --- |
| **Actividad** | **% esfuerzo** | **Valor esfuerzo** |
| Planificación | 10 | 180 |
| Diseño | 20 | 320 |
| Codificación | 40 | 460 |
| Prueba | 15 | 250 |
| Sobrecarga | 15 | 250 |
| Total | 100 | **1460** |

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.

**Ecuación 3.6**: Costo.

**C = E (Total)\*CHH**

Dónde:

**CHH**: Costo por hombre hora

A continuación, se muestra la ecuación 2.7 de costo por hombre hora

Ecuación 2.7: Costo por hombre hora

**CHH = K \* THP**

Dónde:

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

THP: Tarifa Horaria Promedio. El salario promedio de las personas (2 personas en este caso: Diseñador (Analista) y Desarrollador) que trabajan en el proyecto que es $2100 en CUP dividido entre 160 horas.

**CHH** = 1 \* 2100 / 160

**CHH** = 13.125

Entonces:

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

El salario promedio del desarrollador de este caso de estudio es de $2400 en CUP y por tanto la THP = 2400 / 160 = 15

**C** = 460 \* 1.5 \* 15 = $ 10.350

De los resultados obtenidos se interpreta que el proyecto se desarrolla en aproximadamente 2 meses y 10 días. El costo de producción total se estima que sea **$ 10.350** en CUP.

## **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.

Referencias Bibliográficas

[1] [Robot conversacionals: La guía definitiva (2020) - IA Conversacional para Empresas | Artificial Solutions (artificial-solutions.com)](https://www.artificial-solutions.com/es/chatbots/) 11-2-2022

[2] A. Augello, G. P., A. Machi, y S. Gaglio. (2012). An approach to enhance robot conversacional semantic power and maintainability: experiences within the FRASI project.

[3] Fourault, S. (2017). The Ultimate Guide to Designing A Robot conversacional Tech Stack. Retrieved from [https://robot conversacionalsmagazine.com/the-ultimate-guide-to-designing-a-robot conversacional-tech-stack-333eceb431da](https://chatbotsmagazine.com/the-ultimate-guide-to-designing-a-chatbot-tech-stack-333eceb431da) (20 de octubre de 2020)

[4] Arce, G. P., & Arce, R. P. (2021). Transform Your Business with AI Robot conversacionals.

[5] Ernesto Guerra, Amanda, el robot conversacional que sabe de elecciones en Cuba. [https://planetarobot conversacional.com/amanda-el-robot conversacional-que-sabe-de-elecciones-en-cuba-e2c3b5a7d835](https://planetachatbot.com/amanda-el-chatbot-que-sabe-de-elecciones-en-cuba-e2c3b5a7d835)

[6] <https://azure.microsoft.com/en-us/services/bot-services/> 4-11-2021

[6.1] <https://docs.microsoft.com/en-us/composer/introduction> 5-11-2021

[6.2] <https://paginapropia.com/15-ventajas-y-desventajas-de-microsoft-azure/> 6-11-2021

[6.3] <https://stackshare.io/stackups/dialogflow-vs-microsoft-bot-framework/> 5-11-2021

[6.4] <https://rijsat.com/2021/11/03/create-azure-bot-service-from-azure-portal/> 6-11-2021

[7] <https://cloud.google.com/dialogflow/docs> 4-11-2021

[7.1] <https://cloud.google.com/dialogflow?hl=es> 5-11-2021

[7.2] [https://www.robot conversacionals.org/dialogflow](https://www.chatbots.org/dialogflow) 6-11-2021

[7.3] <https://www.capterra.es/reviews/180853/dialogflow?overall_rating_ge=5> 6-11-2021

[7.4] [https://www.linkedin.com/pulse/dialogflow-vs-lex-watson-azure-bot-robot conversacional-quick-sherwin-fernandes](https://www.linkedin.com/pulse/dialogflow-vs-lex-watson-azure-bot-chatbot-quick-sherwin-fernandes) 5-11-2021

[8] <https://rasa.com/> 3-11-2021

[8.1] [https://www.spaceo.ca/ai-robot conversacional-development-using-rasa-reasons/](https://www.spaceo.ca/ai-chatbot-development-using-rasa-reasons/) 3-11-2021

[8.2] [https://www.robot conversacionals.org/best-robot conversacional-builders#builder5](https://www.chatbots.org/best-chatbot-builders#builder5) 6-11-2021

[9] <https://botpress.com/docs/introduction> 3-11-2021

[9.1] [https://www.getapp.es/reviews/2047766/botpress 4-11-2021](https://www.getapp.es/reviews/2047766/botpress%204-11-2021)

[10] <https://www.saashub.com/compare> 5-11-2021

[10.1] [https://www.spaceo.ca/top-ai-robot conversacional-marcos/](https://www.spaceo.ca/top-ai-chatbot-frameworks/) 7-11-2021

[11] Papangelis, A., Namazifar, M., Khatri, C., Wang, Y.-C., Molino, P., & Tur, G. (2020). *Plato Dialogue System: A Flexible Conversational AI Research Platform*.

[12] Waghmare, C. (2019). Introducing Azure Bot Service Building Bots for Business.

[13] Navin Sabharwal, A. A. (2020). Cognitive Virtual Assistants Using Google Dialogflow Develop Complex Cognitive Bots Using the Google Dialogflow Platform.

[14] Raj, S. (2019). Building Robot conversacionals with Python Using Natural Language Processing and Machine Learning.

[15] Álvaro Castillo Cabero, P. P. M., Joan Antoni Pastor Collado. (2020). Rasa Framework: Análisis e implementación de un Robot conversacional.

[16] Tri Chau Minh Tri, N. T. D., Koh Wee Lit (2020). Creating smart, human-like robot conversacional for businesses using BotPress platform.

[17] [https://es.wikipedia.org/wiki/Python](https://es.wikipedia.org/wiki/Python%2011-2-2022)  11-2-2022

[18] <https://es.wikipedia.org/wiki/PyCharm> 11-2-2022

[19] [NoSQL - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/NoSQL) 11-2-2022

[20] [YAML - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/YAML) 11-2-2022

[21] *Moon Modeler | Data Modeling Tool*. (s. f.). Recuperado 9 de octubre de 2022, de <https://www.datensen.com/>

[22] XP - Extreme Programing Ingenieria de Software. [XP - Extreme Programing Ingenieria de Software (mex.tl)](http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html) 15-6-2021

[23] 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