Facultad 2

Plataforma de Chatbots conversacionales para la Universidad de Ciencias Informáticas

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autor: Alexander Rodríguez Londres

Tutor:

Dr. C. Héctor Raúl González Díez

Ing. Vladimir Milián Núñez

Co-tutora:

Ing. Camila Martínez Pita

La Habana, agosto de 2023

Año 65 de la Revolución

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título ***"Plataforma de Chatbots conversacionales para la Universidad de Ciencias Informáticas"*** concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firman la presente a los 20 días del mes de agosto del año 2023.

_____

Firma del autor

Alexander Rodríguez Londres

_____ _____

Firma del tutor Firma del tutor

Dr. C. Héctor Raúl González Díez Ing. Vladimir Milián Núñez

_____

Firma de la cotutora

Ing. Camila Martínez Pita

RESUMEN

En la presente investigación se desarrolla una plataforma de chatbots conversacionales que permite la integración efectiva de Inteligencia Artificial en el estudio de la asignatura de Estructura de Datos. Con el fin de alcanzar los objetivos trazados se analizan los principales conceptos asociados a los chatbots conversacionales, se realiza una investigación a los sistemas similares existentes, tanto en el ámbito nacional como internacional. Se utilizaron como lenguaje de desarrollo Python, html5, css3 y JavaScript, se seleccionó la metodología Scrum y se realizan un conjunto de pruebas para garantizar el correcto funcionamiento de las funcionalidades propuestas, teniendo en cuenta también que la plataforma cumple con los requerimientos del cliente.

Palabras claves: chatbots, conversacionales, plataforma

ÍNDICE

ÍNDICE DE TABLAS

,

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En el mundo contemporáneo, donde la información y la comunicación juegan un papel fundamental, las Tecnologías de la Información y Comunicación (TIC) se han convertido en un elemento esencial en la rutina diaria de la sociedad a nivel global. Estas TIC, que abarcan desde la expansión de Internet hasta el desarrollo de dispositivos móviles inteligentes, han transformado la forma en que vivimos, trabajamos y nos relacionamos. En este contexto, la inteligencia artificial (IA) ha emergido como un pilar fundamental en el desarrollo de soluciones tecnológicas innovadoras que impactan profundamente en la vida de las personas. La inteligencia artificial, una disciplina que busca dotar a las máquinas de la capacidad de aprender y razonar como seres humanos, ha experimentado un crecimiento exponencial en las últimas décadas. Desde asistentes virtuales en dispositivos móviles hasta sistemas de recomendación en plataformas de streaming, la IA está presente en una amplia gama de aplicaciones que enriquecen las experiencias diarias. La capacidad de las máquinas para procesar datos, comprender el lenguaje humano y tomar decisiones basadas en el aprendizaje automático ha transformado industrias enteras y ha abierto nuevas posibilidades en campos tan diversos como la medicina, la automoción y la atención al cliente.

Uno de los desarrollos más notables dentro de la IA es la creación de bots conversacionales[1], que son programas diseñados para interactuar con los usuarios a través del lenguaje natural. Estos bots, también conocidos como chatbots, han adquirido una importancia significativa en la comunicación digital y se utilizan en una variedad de aplicaciones que van desde la atención al cliente automatizada hasta la asistencia en la navegación por sitios web y la provisión de información en tiempo real. Los chatbots se han convertido en un puente fundamental entre las empresas y sus clientes, ofreciendo respuestas rápidas y soluciones eficientes a las preguntas y necesidades de los usuarios.

Los chatbots han emergido como una herramienta innovadora y valiosa en el ámbito universitario. Estos sistemas, diseñados para comunicarse con seres humanos a través del lenguaje natural, están transformando la manera en que se relaciona con las instituciones educativas. Los chatbots, o bots conversacionales, son capaces de ofrecer respuestas instantáneas, resolver dudas frecuentes, proporcionar información relevante y facilitar la navegación en sistemas y plataformas académicas. Su

[1] es un programa de software diseñado para mantener conversaciones con seres humanos a través de interfaces de chat, voz o texto

capacidad para brindar atención eficiente al usuario, así como para mejorar la comunicación y el acceso a recursos académicos, ha convertido a los chatbots en aliados clave en el entorno universitario.

En el marco de las iniciativas impulsadas por el gobierno cubano para promover la utilización de las TIC, surgió la Universidad de Ciencias Informáticas (UCI). La UCI se erigió como el epicentro para la informatización del país y el impulso de la industria del software, respaldando así el progreso económico de Cuba. Dentro de su programa académico, la asignatura Estructura de Datos ocupa un lugar fundamental pues se enfoca en la enseñanza de conceptos clave relacionados con la organización y gestión eficiente de datos en el contexto de la programación orientada a objetos a través de algoritmos, métodos y otras técnicas para el desarrollo de un software de alta calidad. El empleo de chatbots se ha convertido en una práctica muy común y eficaz para adquirir información técnica, resolver dudas y obtener orientación sobre los lenguajes de programación impartidos en la universidad, pues son un recurso rápido y disponible las 24 horas para mejorar las habilidades en la asignatura. A pesar de los avances tecnológicos a nivel global, se destaca una significativa ausencia de una plataforma de chatbots conversacionales propia tanto a nivel nacional como en el ámbito universitario; esto es originado debido a diversos factores, entre los que se incluyen limitaciones económicas y de infraestructura, así como la falta de acceso a recursos y conocimientos especializados en el desarrollo de IA.

Esta carencia tecnológica limita a la comunidad universitaria a la dependencia de chatbots externos disponibles en Internet, lo cual conlleva una pérdida de control sobre la calidad y la actualización de las respuestas brindadas; pues muchos de los chatbots disponibles en línea utilizan modelos de inteligencia artificial desactualizados, la información proporcionada puede ser inexacta y no estar alineada con las necesidades y los requisitos específicos de la institución. Este escenario pone en riesgo la calidad de la educación y los servicios que la universidad puede ofrecer; además de que también introduce una variabilidad en la calidad de las respuestas, ya que algunos de estos son de pago y pueden generar costos adicionales para los estudiantes y el personal; y aquellos que son gratuitos, a menudo tienen limitaciones en el ciclo de respuesta, lo que puede resultar en experiencias de usuario insatisfactorias y prolongadas esperas para recibir asistencia.

Partiendo de la situación problemática expuesta anteriormente se plantea el siguiente **problema a resolver:** ¿Cómo mitigar la dependencia de chatbots externos disponibles en Internet y sus efectos negativos en la calidad y actualización de las respuestas ofrecidas a la comunidad universitaria?

Para dar solución al problema de investigación planteado, se define como **objetivo general:** Desarrollar una plataforma de Chatbots conversacional para la Universidad de Ciencias Informáticas que permita la integración efectiva de Inteligencia Artificial en el estudio de la asignatura de Estructura de Datos. Teniendo como **objeto de estudio:** Plataformas de Chatbots conversacionales; definiéndose como **campo de acción:** Modelos de lenguaje natural.

Para cumplir con el objetivo marcado se plantean las siguientes **objetivos específicos:**

- Elaboración del marco teórico de la investigación y tendencias en el desarrollo de bots conversacionales.
- Caracterización de plataformas existentes en el ámbito internacional.
- Definición de la metodología, herramientas y lenguajes a utilizar en el desarrollo de la propuesta.
- Identificación de las funcionalidades a tener en cuenta para el desarrollo de la propuesta.
- Implementación de la plataforma.
- Validación de la propuesta, a través de una estrategia de pruebas, que permita evaluar su potencial y rendimiento.

Método Científico

El método científico de investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. El método científico se puede clasificar en teóricos y empíricos, los cuales están dialécticamente relacionados.

Métodos Teóricos

- **Analítico-Sintético:** Empleado para el análisis, evaluación y selección de las técnicas a emplear en el desarrollo de la plataforma, de esta forma se pudo determinar los principios fundamentales que debe cumplir la propuesta de solución.
- **Modelación:** Utilizado con el objetivo de crear abstracciones de la plataforma desarrollado mediante diferentes diagramas ingenieriles. Permitiendo una mejor comprensión del diseño de la propuesta de solución.

Métodos empíricos

- **Análisis documental:** Mediante la selección y recolección de la documentación asociada al tema abordado, se hace uso de este método con el objetivo de extraer la información necesaria para lograr los resultados esperados.
- **Entrevista:** se realiza para la obtención de toda la información referente al desarrollo de las funcionalidades a desarrollar en la plataforma.

La presente investigación consta de tres (3) capítulos, los cuales se resumen brevemente a continuación:

Capítulo 1: Fundamentación Teórica y Metodológica sobre las plataformas de chatbots conversacionales.

En este capítulo se presentan los elementos teóricos que sirven de base a la investigación del problema planteado. Se describen los conceptos fundamentales asociados al dominio del problema; además, se proporciona una descripción de las herramientas, tecnologías y metodología de desarrollo de software a utilizar para dar solución al problema planteado.

Capítulo 2: Planificación y Diseño de la propuesta de solución.

En este capítulo se realiza la planificación y diseño de la propuesta de solución según los requisitos funcionales obtenidos. Se realiza una descripción del sistema que se pretende desarrollar, se elaboran las Historias de Usuarios para identificar las funcionalidades con las que debe cumplir el sistema, además de presentar la arquitectura seleccionada para su desarrollo.

Capítulo 3: Implementación y Prueba de la propuesta de solución.

En este capítulo se presentan los elementos relacionados con la implementación del sistema. Se define la estrategia de pruebas a realizar para verificar el cumplimiento de los requisitos de software y se presenta el resultado del proceso de prueba y de la validación del sistema.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA Y METODOLÓGICA SOBRE LAS PLATAFORMAS DE CHATBOTS CONVERSACIONALES

La comunicación ha sido un pilar fundamental en la evolución de la sociedad, y a lo largo de la historia, ha experimentado una constante transformación. En la era digital actual, la tecnología ha revolucionado la forma en que las personas se conectan, interactúan y comparten información. En este contexto, los chats se rigen como una herramienta esencial que ha redefinido la comunicación en tiempo real.

1.1 Chat

Un chat es un servicio de mensajería instantánea que permite comunicarse a dos o más personas de forma inmediata y mantener una conversación por escrito en tiempo real. Los usuarios comparten un programa común a través de Internet en el cual se escribe lo que se desea en una pequeña consola de texto (Merino, 2011). Los chats se consideran como canales de conexión entre los diferentes ordenadores que posibilitan el dialogo colectivo entre sujetos que se encuentran separados en espacios físicos diferentes (Loos, 2016). El chat, como medio de comunicación, ha sido uno de los grandes avances de las tecnologías de la información y comunicación. Ha logrado que millones de personas se comuniquen al instante sin importar las distancias o diferencias de horario.

Los chats son conversaciones que se llevan a cabo al instante gracias al uso de un software conectado a una red de Internet. Se pueden diferenciar chats públicos**,** grupos de conversación en los cuales puede participar cualquier persona, o chats privados**,** que solo pueden participar usuarios autorizados. El objetivo primordial de los chats es lograr que las personas se comuniquen e intercambien información al instante**.** Sin importar donde se localicen, acortando de esta manera las barreras de la distancia y el tiempo de manera mucho más económica (Chat, 2021).

1.2 Bot

Un Bot es una aplicación de software automatizada que realiza tareas repetitivas en una red. Dicha aplicación sigue instrucciones específicas para imitar el comportamiento humano, pero es más rápida y precisa. Este también se puede ejecutar de forma independiente sin intervención humana, pues sigue reglas e instrucciones precisas para realizar sus tareas. Una vez activados, los bots pueden comunicarse entre sí o con los

humanos utilizando protocolos de comunicación de red estándar. Funcionan continuamente para realizar tareas programadas con muy poca intervención humana (Amazon, 2021).

1.3 Chatbot

Es un programa informático que utiliza IA y procesamiento del lenguaje natural (PLN) para comprender las preguntas de lo clientes y automatizar las respuestas a dichas preguntas, simulando la conversación humana (IBM, 2021). Está programado para que interactúe con el cliente y resuelva dudas, pero sin que haya una persona física contestando, tienen la ventaja de que están disponibles siempre para resolver las dudas de los usuarios a cualquier hora del día (Martínez Peña, 2017). Utilizan tecnologías de aprendizaje profundo como la conversión de texto en voz, el reconocimiento automático de voz y el procesamiento del lenguaje natural para simular conversaciones y diálogos humanos.

Los consumidores utilizan chatbots para muchos tipos de tareas, desde la interacción con aplicaciones móviles hasta el uso de dispositivos para fines específicos, como termostatos inteligentes y aparatos de cocina inteligentes. En el ámbito comercial, también hay usos diversos. Los vendedores lo utilizan para personalizar las experiencias de los clientes, los equipos de TI para habilitar el autoservicio, y los centros de contacto de cliente confían en los chatbots para optimizar las comunicaciones entrantes y remitir a los clientes a los recursos adecuados.

Las interfaces conversacionales también pueden variar. Los chatbots se suelen utilizar en aplicaciones de mensajería de redes sociales, plataformas de mensajería autónomas o aplicaciones en sitios web. Entre los casos de uso más frecuentes, se incluyen:

- Encontrar restaurantes locales y dar direcciones.
- Definir campos dentro de formularios y aplicaciones financieras.
- Obtener respuestas a preguntas sobre atención sanitaria y programación de citas.
- Recibir asistencia general del servicio al cliente de una marca favorita.
- Establecer un recordatorio para realizar una tarea basándose en la hora o la ubicación.
- Mostrar las condiciones meteorológicas en tiempo real y recomendaciones sobre cómo vestir (IBM, 2021).

1.3.1 Inteligencia Artificial (IA)

Es una disciplina y un conjunto de capacidades cognoscitivas e intelectuales expresadas por sistemas informáticos o combinaciones de algoritmos cuyo propósito es la creación de máquinas que imiten la inteligencia humana para realizar tareas, y que pueden mejorar conforme recopilen información. Abarca un amplio conjunto de métodos y disciplinas, en particular, los sistemas de visión, percepción, habla y diálogo, toma de decisiones y planificación, resolución de problemas y robótica, así como otros tipos de aplicaciones de aprendizaje autónomo.

Esta brinda ventajas en muchos sectores en cuanto a la prestación de servicios nuevos e innovadores, así como la capacidad de mejorar su alcance, eficacia y precisión. Por otro lado, amplía y conjuga muchas de esas ventajas con información estadística y macrodatos. Sobre la base de un análisis evolutivo, la IA facilita la transición de modelos empresariales y políticos y de enfoques normativos, basados actualmente en análisis **descriptivos** y de identificación de tendencias, a nuevos modelos y enfoques más adaptables, **proactivos** y predictivos basados en evidencias. En particular, se usa para detectar la evolución de vulnerabilidades sanitarias y de riesgos en la esfera de los seguros, entre otras muchas aplicaciones.

La utilización de herramientas y técnicas de este dominio de aplicación propicia nuevas oportunidades en una gran cantidad de sectores. La IA, así como otros algoritmos, se utilizan ampliamente para las búsquedas en línea, las actividades de entretenimiento, las redes sociales, los coches de conducción autónoma, el reconocimiento visual, las herramientas de traducción, los asistentes/altavoces inteligentes y la transformación de voz en texto, entre muchas otras aplicaciones (UIT, 2023).

1.3.2 Procesamiento de lenguaje natural

Es el campo de la IA que se ocupa de investigar la manera de comunicar las máquinas con las personas mediante el uso de las lenguas naturales, como el español o inglés. Este lenguaje toma elementos prestados de muchas disciplinas, incluyendo la ciencia de la computación y la lingüística computacional, en su afán por cerrar la breca entre la comunicación humana y el entendimiento de las computadoras. El procesamiento del lenguaje natural va de la mano de la analítica de texto, la cual cuenta, agrupa y categoriza palabras para extraer estructura y significado de grandes volúmenes de contenido. Esta se utiliza para explorar contenido textual y derivar nuevas variables de texto crudo que se

puede visualizar, filtrar o utilizar como entradas para modelos predictivos u otros métodos estadísticos (Moreno, 2017).

1.3.3 Machine Learning

Machine learning es una forma de la IA que permite a un sistema aprender de los datos en lugar de aprender mediante la programación explícita. Sin embargo, no es un proceso sencillo. Conforme el algoritmo ingiere datos de entrenamiento, es posible producir modelos más precisos basados en datos. Un modelo de machine learning es la salida de información que se genera cuando entrena su algoritmo de machine learning con datos. Después del entrenamiento, al proporcionar un modelo con una entrada, se le dará una salida. Por ejemplo, un algoritmo predictivo creará un modelo predictivo. A continuación, cuando proporcione el modelo predictivo con datos, recibirá un pronóstico basado en los datos que entrenaron al modelo.

Machine learning permite modelos a entrenar con conjuntos de datos antes de ser implementados. Algunos modelos de machine learning están online y son continuos. Este proceso iterativo de modelos online conduce a una mejora en los tipos de asociaciones hechas entre los elementos de datos. Debido a su complejidad y tamaño, estos patrones y asociaciones podrían haber sido fácilmente pasados por alto por la observación humana. Después de que un modelo ha sido entrenado, se puede utilizar en tiempo real para aprender de los datos. Las mejoras en la precisión son el resultado del proceso de entrenamiento y la automatización que forman parte del machine learning.

Las técnicas de machine learning son necesarias para mejorar la precisión de los modelos predictivos. Dependiendo de la naturaleza del problema empresarial que se está atendiendo, existen diferentes enfoques basados en el tipo y volumen de los datos (IBM, 2022).

1.3.3.1 Categorías del machine learning

- **Aprendizaje supervisado:** El aprendizaje supervisado comienza típicamente con un conjunto establecido de datos y una cierta comprensión de cómo se clasifican estos datos. El aprendizaje supervisado tiene la intención de encontrar patrones en datos que se pueden aplicar a un proceso de analítica. Estos datos tienen características etiquetadas que definen el significado de los datos. Por ejemplo, se puede crear una aplicación de machine learning con base en imágenes y descripciones escritas que distinga entre millones de animales (IBM, 2022).

- **Aprendizaje no supervisado**: Se utiliza cuando el problema requiere una cantidad masiva de datos sin etiquetar. Por ejemplo, las aplicaciones de redes sociales, tales como Twitter, Instagram y Snapchat, tienen grandes cantidades de datos sin etiquetar. La comprensión del significado detrás de estos datos requiere algoritmos que clasifican los datos con base en los patrones o clústeres que encuentra. El aprendizaje no supervisado lleva a cabo un proceso iterativo, analizando los datos sin intervención humana. Se utiliza con la tecnología de detección de spam en e-mails. Existen demasiadas variables en los e-mails legítimos y de spam para que un analista etiquete una cantidad masiva de e-mail no solicitado. En su lugar, los clasificadores de machine learning, basados en clustering y asociación, se aplican para identificar email no deseado (IBM, 2022).

- **Aprendizaje de refuerzo**: Es un modelo de aprendizaje conductual. El algoritmo recibe retroalimentación del análisis de datos, conduciendo el usuario hacia el mejor resultado. Este difiere de otros tipos de aprendizaje supervisado, porque el sistema no está entrenado con el conjunto de datos de ejemplo. Más bien, el sistema aprende a través de la prueba y el error. Por lo tanto, una secuencia de decisiones exitosas conduce al fortalecimiento del proceso, porque es el que resuelve el problema de manera más efectiva (IBM, 2022).

- **Deep learning**: Es un método específico de machine learning que incorpora las redes neuronales en capas sucesivas para aprender de los datos de manera iterativa. Este es especialmente útil cuando se trata de aprender patrones de datos no estructurados. Las redes neuronales complejas de deep learning están diseñadas para emular cómo funciona el cerebro humano, así que las computadoras pueden ser entrenadas para lidiar con abstracciones y problemas mal definidos. Las redes neuronales y el deep learning se utilizan a menudo en el reconocimiento de imágenes, voz y aplicaciones de visión de computadora (IBM, 2022).

1.4 Plataformas de Chatbots Conversacionales

En el panorama digital actual, la implementación de chatbots se ha vuelto esencial para una comunicación efectiva y una atención al cliente mejorada. A continuación se examinan las características y ventajas de un grupo de chatbots que permitirá comprender cómo la tecnología de procesamiento de lenguaje natural está moldeando la interacción en línea y mejorando la experiencia del usuario en diversos contextos empresariales y de servicios.

1.4.1 Open AI ChatGPT

ChatGPT es un chatbot que se basa en inteligencia artificial y que permite responder preguntas de forma conversacional. Desarrollado por OpenAI, este es un gran modelo de lenguaje, ajustado con técnicas de aprendizaje tanto supervisadas como de refuerzo. Está compuesto por los modelos GPT-4 y GPT-3.5 (una versión mejorada de GPT 3). Desde su lanzamiento, el 30 de noviembre de 2022**,** ChatGPT ha llamado la atención de miles y miles de entusiastas de la tecnología. Todo el mundo puede hacer consultas al chatbot accediendo a la web oficial y registrándose. El entrenamiento al que ha sido sometido incluye ingentes cantidades de datos, para poder realizar una amplia variedad de tareas relacionadas con el lenguaje natural. Su capacidad para comprender el contexto y la intención detrás de las preguntas o consultas de los usuarios lo convierten en una herramienta muy útil para desarrollar chatbots y mejorar la precisión en los sistemas de búsqueda de información. No obstante, su base de conocimientos llega hasta algún momento de 2021 (Ortiz, 2023).

Características destacables:

- **Compatibilidad con múltiples idiomas**: Entiende prácticamente cualquier idioma y se puede comunicar con el usuario en su lengua materna.

- **Redacta utilizando varios estilos**: El modelo conversacional permite generar textos usando diversos estilos, por ejemplo, basados en escritores famosos.

- **Tiene una interfaz que es muy fácil de asimilar**: Se presenta con una interfaz muy simple. A pesar de la complejidad de su motor, no tiene más que un cajón de texto y una pantalla en la que se suceden las respuestas. Muy similar a cualquier otra app de chats (Otero, 2023).

1.4.2 ChatSonic

ChatSonic es una herramienta creada por Writesonic con una gran potencia y capacidades similares a las de ChatGPT. Con esta plataforma te será realmente fácil hacer búsquedas, hallar resultados, obtener resúmenes y perfeccionar tus estrategias de escritura creativa mediante instrucciones basadas en texto. Por otro lado, ChatSonic también cuenta con su propia API, facilitando que integres la IA dentro de tu sitio web para optimizar tus chatbots y ofrecer servicios automatizados de interacción con el cliente. Por todo lo anterior, esta

herramienta es ideal para quienes requieren aprovechar las ventajas de la IA conversacional sin las limitaciones de ChatGPT (Pérez, 2021).

Características principales:

- Funciona mediante el buscador y la base de datos de Google.
- Se encuentra siempre conectado a la red para ofrecer resultados con datos actuales.
- Permite la generación de imágenes y recursos visuales mediante potentes motores de IA.
- Las suscripciones están limitadas a un número de palabras por mes, por lo que estás sujeto al límite contratado.

1.4.3 Poe

Poe es una aplicación de chat de inteligencia artificial que ofrece respuestas rápidas y precisas a las preguntas de los usuarios. La aplicación utiliza tecnología de procesamiento de lenguaje natural y aprendizaje automático para comprender y responder a las preguntas de los usuarios de manera eficiente y precisa. Además de responder a las preguntas de los usuarios, Poe también ofrece una experiencia de usuario intuitiva y atractiva, con una interfaz de usuario moderna y fácil de usar. Hay muchas ventajas de usar Poe en comparación con otras aplicaciones de chat.

Algunas de las **ventajas** incluyen:

- Rapidez: con su tecnología de procesamiento de lenguaje natural y aprendizaje automático, Poe es capaz de proporcionar respuestas rápidas y precisas a las preguntas de los usuarios.
- Precisión: Poe tiene acceso a una amplia base de conocimientos en línea para proporcionar respuestas precisas y actualizadas.
- Accesibilidad: con la capacidad de responder a preguntas en varios idiomas, Poe es accesible para una audiencia global.

- Interfaz de usuario intuitiva: con una interfaz de usuario moderna y atractiva, Poe es fácil de usar y navegar (Poe, 2023).

Características principales:

- Permite a los usuarios interactuar con GPT-4 una vez al día sin costo, brindándoles acceso a una de las IA más avanzadas disponibles.

- Variedad de inteligencias artificiales: La plataforma ofrece una gran variedad de IAs basadas en diferentes modelos y tecnologías, lo que permite a los usuarios explorar y experimentar con varias soluciones de IA.

- Creación de bots personalizados: La característica más destacada de Poe.com es su capacidad para crear bots basados en ChatGPT. Los usuarios pueden proporcionar una instrucción específica para el Bot, lo que facilita su uso para otras personas. Los usuarios que interactúan con el Bot simplemente necesitan especificar lo que necesitan, y el Bot lo hará por ellos (ARIELMCORG, 2023).

Luego de una exhaustiva investigación en el campo de estudio, se constató que en Cuba no existen herramientas o sistemas previamente implementados que sean homólogos a la propuesta que se está desarrollando. Este proyecto pionero en la nación se presenta como un paso significativo en el avance tecnológico y la optimización de procesos en el contexto local. Los beneficios que esta iniciativa aportará son múltiples, ya que no solo suplirá una carencia identificada, sino que también promoverá la eficiencia, la competitividad y la innovación en las áreas en las que se aplicará, sirviendo como plataforma para futuras investigaciones y desarrollos tecnológicos.

1.4.4 Google Bard

Bard es un sistema de Inteligencia Artificial creado por Google. Se trata de un sistema conversacional, de forma que vas a poder interactuar con él mediante mensajes normales. En estos mensajes, tú le escribirás algo que quieras saber o quieras que haga, y Bard responderá o lo hará.

Esta IA del gigante del buscador está basada en LaMDA, un potente modelo de lenguaje experimental diseñado por Google específicamente para aplicaciones de diálogo. Este

modelo llevaba un tiempo en fase de pruebas muy cerrada, ya que solo unas pocas personas podían acceder a ella.

Desde hace años, Google lidera de forma incontestable el mercado de las búsquedas en Internet. Sin embargo, esta posición se ha visto recientemente amenazada con la llegada de modelos de IA capaz de darte información de forma conversacional. Si quieres buscar algo en Google, tienes que mirar entre sus webs, mientras que IAs como ChatGPT están entrenadas para darte una respuesta directa (Fernández, 2023).

Tabla 1 *Comparación entre sistemas homólogos*

Características	**Open AI ChatGPT**	**Bard**	**ChatSonic**	**Poe**
Capacidades de Generación	Alta	Alta	Alta	Alta
API Disponible	No	Si	Sí	Sí
Variedad de Modelos	GPT-4 y GPT-3.5	PaLM 2	ChatGPT	GPT-4, GPT-3.5, Llama 2, Claude-Instant
Creación de Bots	No	No	Sí	Sí
Inserción de nuevos modelos	No	No	No	No
Limitaciones Mensuales	No	Si	Sí	Sí

Facilidad de Uso	Alta	Alta	Alta	Alta

En el análisis de las herramientas disponibles, se ha identificado tres elementos cruciales:

- En primer lugar, la disponibilidad de una interacción gratuita, como la ofrecida por Poe, resulta esencial para reducir costos iniciales y permitir un mayor acceso a la herramienta. Esta característica proporciona un valor significativo al proyecto al eliminar barreras financieras para la exploración y experimentación.

- En segundo lugar, la capacidad de incorporar nuevos modelos se destaca como una ventaja crucial. Esto asegura que el sistema desarrollado pueda mantenerse al día con los avances tecnológicos y las necesidades cambiantes, lo que es especialmente relevante en el campo de la inteligencia artificial en constante evolución. Los usuarios podrán adaptar y personalizar el sistema para satisfacer sus necesidades específicas, lo que aumentará su versatilidad y su aplicabilidad en una amplia gama de contextos. Este enfoque pionero en la inserción de nuevos modelos coloca al proyecto en la vanguardia de la tecnología y brinda la oportunidad de influir en el desarrollo futuro de sistemas de inteligencia artificial. Al ofrecer a los usuarios la capacidad de integrar modelos personalizados, se empodera a las personas y organizaciones para que exploren nuevas fronteras en la generación de lenguaje natural y la automatización de tareas basadas en texto. En última instancia, este aporte puede revolucionar la forma en que se abordan los desafíos y se desarrollan soluciones en el campo de la inteligencia artificial.

- Por último, la facilidad de uso, compartida por todas las herramientas analizadas, es un elemento fundamental para maximizar la productividad y la eficiencia en el desarrollo del proyecto. Una interfaz amigable y accesible facilita la adopción rápida y efectiva de la herramienta, permitiendo que los equipos de trabajo se centren en la aplicación y mejora de la tecnología en lugar de lidiar con complejidades innecesarias.

1.5 Metodología de Desarrollo

Una metodología de desarrollo de software es un marco de trabajo utilizado para estructurar, planificar y controlar el proceso de desarrollo de software. Este marco de trabajo consiste en una filosofía de desarrollo de software, herramientas, modelos y métodos que asisten al proceso de desarrollo de software. (Molina, 2013)
Para la selección de la metodología se aplica el modelo propuesto por Barry Boehm y Richard Turner, conocido por el método de Boehm y Turner que caracteriza el proyecto de software y estima cuán **ágil** o **prescriptivo** debería ser el enfoque a utilizar a partir de cinco criterios, estos son: **tamaño del equipo**, **criticidad del producto**, **dinamismo de los cambios**, **cultura del equipo** y **personal** con que se cuenta. Cada uno de esos criterios tiene elementos que lo discriminan y por tanto se tienen en cuenta a la hora de seleccionar uno u otro enfoque.

El **enfoque prescriptivo**, denominado en algunas bibliografías como tradicional o clásicas, busca la estructura, poner orden en el caos del desarrollo de software en cuestión, el proceso es de manera secuencial, en una sola dirección y sin marcha atrás.

El **enfoque ágil** presenta como principal particularidad la flexibilidad, los proyectos en desarrollo son subdivididos en proyectos más pequeños, incluye una comunicación constante con el usuario, son altamente colaborativos y es mucho más adaptable a los cambios (Molina Montero & Vite Cervallo, 2018).

A continuación, se muestra una evaluación de las características del proyecto para determinar la idoneidad de un enfoque ágil o tradicional a partir de los siguientes criterios:

- **Personal:** El equipo de desarrollo cuenta con un doctor en ciencias altamente calificado para su desarrollo, dos ingenieros en ciencias informáticas y también con un estudiante de ingeniería que presenta conocimientos adquiridos durante la carrera sobre el tema propuesto. Determinando que el equipo de desarrollo contiene en su mayoría personal experimentado para la solución del proyecto.

- **Criticidad:** El equipo de desarrollo tiene una elevada responsabilidad con la calidad del producto a obtener. Los errores que presente no provocarán pérdidas de vidas humanas ni de bienes materiales.

- **Tamaño:** El equipo de desarrollo está compuesto por cuatro integrantes.

- **Dinamismo:** Se asume el constante cambio de requisitos durante el ciclo de desarrollo del software. Para reducirlo se toman en cuenta una cantidad de funcionalidades básicas que no pueden ser cambiadas a lo largo del proceso.

- **Cultura:** Los elementos antes descritos evidencian organización en el desarrollo del trabajo, pero al ser un equipo pequeño no necesitan relación contractual dada la buena comunicación y confianza entre sus miembros. Cada especialista en el desempeño de su rol será libre de incorporar ideas que no afecten el diseño del sistema, ni los compromisos planificados.

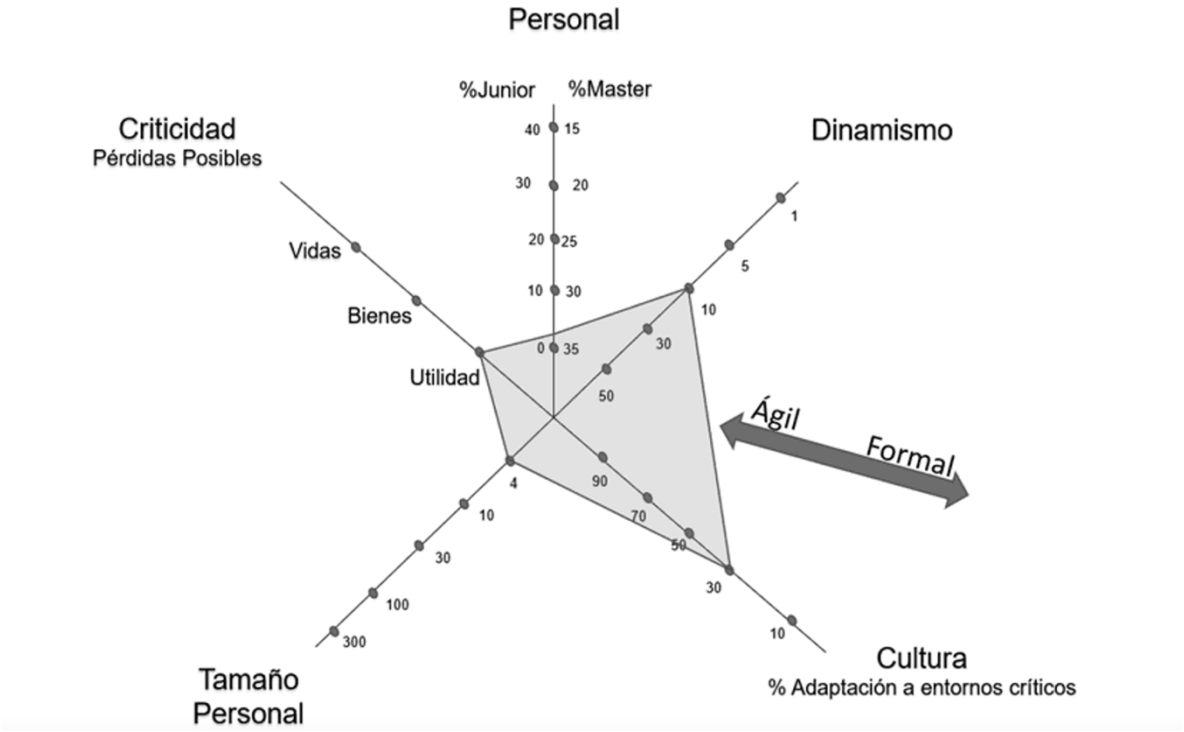


Figura 1 *Estrella de Boehm y Turner*

Teniendo en cuenta el análisis de la de la estrella de Boehm y Turner, se puede afirmar que el uso de un enfoque ágil encuadra mejor para el desarrollo del proyecto debido a que tres de sus cinco criterios son de impacto bajo, haciendo uso de una metodología ágil.

Metodología ágil

La metodología ágil, es una metodología de gestión de proyectos que están especialmente orientadas para proyectos que necesitan de una solución con una elevada simplificación sin dejar de lado el aseguramiento de la calidad del producto. Se centran en el factor humano y el producto, es decir, ellas le dan mayor valor al individuo, a la colaboración del

cliente y al desarrollo incremental del software con iteraciones muy cortas. Utiliza ciclos de desarrollo cortos para centrarse en la mejora continua del desarrollo de un producto o servicio (Molina Montero & Vite Cervallo, 2018).

El desarrollo web exige adaptación, estrategias y cambios continuos. Los usuarios se preocupan por una aplicación Web sea entregada cuando lo necesitan y no sobre el trabajo que se lleva a cabo para crear una aplicación, por lo tanto, el equipo de desarrollo de un proyecto Web debe enfatizar la agilidad. El quipo debe ser capaz de responder adecuadamente a los cambios, tales como, en el software, desarrollo, recurso humano, tecnología entre otros. El éxito del proyecto dependerá de la habilidad del equipo y la capacidad de colaboración (Pinzon & Rodríguez, 2017).

Metodología SCRUM

Se seleccionó para guiar el proceso de desarrollo de software la metodología SCRUM, como parte del cumplimiento de las exigencias del cliente, el cual planteó la necesidad de estar involucrado durante todo el proceso de desarrollo para validar la propuesta de solución a su problemática. La misma es un marco de trabajo diseñado para lograr la colaboración eficaz de equipos en proyectos, que emplea un conjunto de reglas y artefactos y define roles que generan la estructura necesaria para su correcto funcionamiento. SCRUM utiliza un enfoque incremental (Ken Schwaber, 2018).

Para entender todo el proceso de desarrollo de Scrum, se describirá de forma general las fases, los roles y los elementos que forman parte de dicha metodología.

- **Planificación del Backlog (primera fase):** se realizará un documento en el que se reflejarán los requisitos del sistema por prioridades. Se definirá también la planificación del Sprint 0, en la que se decidirá cuáles van a ser los objetivos y el trabajo que hay que realizar para esa iteración. Se obtendrá además un SprintBacklog, que es la lista de tareas y el objetivo más importante del Sprint.

- **Seguimiento del sprint (Segunda fase):** en esta fase se hacen reuniones diarias en las que las tres preguntas principales para evaluar el avance de las tareas son:

    - ¿Qué trabajo se realizó desde la reunión anterior?

    - ¿Qué trabajo se realizará hasta una nueva reunión?

- Inconvenientes que han surgido y que hay que resolver para poder continuar.

- **Revisión del sprint (Tercera fase):** Cuando se finaliza el Sprint se realiza una revisión del incremento que se ha generado y se presentan los resultados finales.

El uso de esta metodología ágil tiene como ventajas:

- **Satisfacción del cliente:** El gran diferenciador de las metodologías ágiles, es que hacen al cliente parte del equipo de trabajo y lo comprometen con el resultado final. Esto es un gran cambio con respecto a metodologías tradicionales, en los que el cliente es una persona, o grupo de personas, con el que se realiza la tarea de levantar requerimientos o necesidades de tipo funcional y que posteriormente aprueba extensos documentos que no volverá a ver hasta que el producto se encuentre terminado.

- **Simplicidad:** los eventos manejados por Scrum están claramente identificados, indicando para cada uno: quienes participan, su objetivo, el tiempo que debe tomar y cuál es el resultado esperado. Lo cual en esencia facilita a los integrantes del equipo la adopción de la metodología.

- **Inspección:** uno de los componentes que resalta Scrum, es la inspección y por ello, tres de sus eventos están orientados a estos objetivos: la reunión diaria, la revisión del sprint y la retrospectiva de este último.

- **Adaptación:** la mejor parte de la metodología es la disposición que tienen al cambio las características del producto. Este es uno de los componentes que más la diferencia con el resto, ya que el cambio puede ser efectuado en cualquier momento, incluso dentro del desarrollo de la ejecución de las diferentes iteraciones o Sprints siempre y cuando no afecte la entrega pactada (Rodríguez & Dorado, 2015).

Fundamentos de SCRUM

Sprint: Evento principal que corresponde a una ventana de tiempo donde se crea una versión utilizable del producto (incremento).

Elementos de un Sprint

- **Planeación del Sprint:** Se define su plan de trabajo: qué se va a entregar y cómo se logrará. Es decir, el diseño del sistema y la estimación de cantidad de trabajo.

- **Daily Scrum:** Es un evento del equipo de desarrollo de quince minutos, que se realiza cada día con el fin de explicar lo que se ha alcanzado desde la última reunión; lo que se hará antes de la siguiente; y los obstáculos que se han presentado.

- **Revisión del Sprint:** ocurre al final del Sprint y su duración es de cuatro horas para un proyecto de un mes (o una proporción de ese tiempo si la duración es menor). En esta etapa: el dueño del proyecto revisa lo que se hizo, identifica lo que no se hizo y discute acerca del Product Backlog; el equipo de desarrollo cuenta los problemas que encontró y la manera en que fueron resueltos, y muestra el producto y su funcionamiento.

- **Retrospectiva del Sprint (Feedback):** Es una reunión de tres horas del equipo Scrum en la que se analiza cómo fue la comunicación, el proceso y las herramientas; qué estuvo bien, qué no, y se crea un plan de mejoras para el siguiente Sprint (Ken Schwaber, 2018).

Roles

- **Product Owner:** Es la única persona autorizada para decidir las funcionalidades y características que tendrá el producto. Es quien representa al cliente y usuarios del software (Bahit, 2012) .

- **Scrum Máster:** Es quien interactúa con el equipo, el cliente y los gestores. Garantiza el funcionamiento de los procesos y la metodología. Debe ser miembro del equipo y trabajar a la par. Coordina los encuentros diarios y se encarga de eliminar obstáculos (Molina, 2013).

- **Scrum Team:** Es el equipo de desarrolladores multidisciplinario, integrado por programadores, diseñadores, arquitectos y testers, que en forma auto-organizada son los encargados de desarrollar el producto (Bahit, 2012).

Artefactos de SCRUM

- **Product Backlog (Listado de Requisitos):** Es una lista ordenada por valor, riesgo, prioridad y necesidad de los requerimientos que el dueño del producto define, actualiza y ordena.

- **Sprint Backlog:** Es un subconjunto de ítems del Product Backlog y el plan para realizar en el Incremento del producto. Debido a que el Product Backlog está organizado por prioridad, el Sprint Backlog es construido con los requerimientos más prioritarios del Product Backlog y con aquellos que quedaron por resolver en el Sprint anterior.

- **Incremento:** Es la suma de todos los ítems terminados en el Sprint Backlog. Si hay ítems incompletos deben ser devueltos al Product Backlog con una prioridad alta para que sean incluidos en el siguiente Sprint (Cadavid et al., 2013).

1.6 Herramientas y Tecnologías

Las herramientas, tecnologías y metodologías son componentes fundamentales para la realización de cualquier software, los cuales forman parte de la política del proyecto y están explícitos en la elaboración de la aplicación.

1.6.1 Herramientas de Desarrollo

Google Colab

Es un entorno colaborativo de Google que permite trabajar con Notebooks y el lenguaje de programación de Python y que a su vez permite almacenar dichos cuadernos y trabajar con datos que tengas almacenados en el Drive y compartirlos con el equipo de trabajo.

Algunas ventajas que ofrece son:

- No requiere configuración, el ambiente está listo para ser usado. Python y muchas librerías están instaladas por defecto.

- Es muy intuitivo, por lo que no se necesita mucho conocimiento previo para usar la herramienta.

- Acceso gratuito.

- Conexión directa a la nube.

- Existen celdas de texto y de código para mantener documentado el programa desarrollado, además de todas las notas que se pueden agregar aparte de los comentarios del código.

- Colab está enlazado a Google drive, por lo que los blocs de notas se guardan por defecto en este servicio en una carpeta que se crea de manera automática.

- Es basado en Jupyter Notebooks, por lo que podrás importar y exportar archivos entre estas plataformas sin problema.

Permite usar GPU de manera gratuita para entrenar los modelos y realizar ciertos procesos realizando todo el procesamiento en la nube de manera gratuita (López, 2022).

Visual Paradigm v.17.1

Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Existe una alternativa libre y gratuita de este software, la versión Visual Paradigm UML 6.4 Community Edition (Community Edition, ya que existe la Enterprise, Professional, etc.). Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (Ecured, 2015).

1.6.2 Lenguajes de Desarrollo

Gherkin

Gherkin es un Lenguaje Específico de Dominio (DSL), que son lenguajes diseñados en concreto para resolver un problema muy específico. Y, en este caso, el problema que quiere solucionar Gherkin es un problema de comunicación entre los perfiles de negocio y los perfiles técnicos a la hora de trabajar bajo un enfoque BDD. También se podría trabajar

Gherkin con otros enfoques de desarrollo de software, pero lo ideal y lo natural es hacerlo con BDD, al ser mejor práctica y para el enfoque de programación para el que se desarrolló Gherkin.

Gherkin está compuesto por varios elementos que permiten esta comunicación entre perfiles de negocio y técnicos de forma sencilla. Gherkin tiene elementos de tipo características, de tipo comportamientos, de tipo acciones, etcétera, pero todo en lenguaje natural, que un perfil de negocio entienda y un técnico pueda trasladar a código (Corrales, 2020).

Gherkin es un lenguaje de marcado, es decir, un sistema de etiquetado o codificación que se utiliza para describir la estructura y el contenido de un documento, en nuestro caso, de una historia de usuario. Este formato a la hora de escribir historias de usuarios (HU) permite añadir una pieza más a la integración continua que va desde la definición de una necesidad o funcionalidad hasta su puesta en producción y posterior seguimiento mediante métricas.

Uno de los mayores retos a los que se enfrentan los Product Owners y Product Managers es crear historias de usuario que comprendan todas las partes involucradas. Para ello se usa BDD, Behavior Driven Development o desarrollo guiado por el comportamiento. Esto permite escribir escenarios de prueba en un formato comprensible tanto para las personas más técnicas que desarrollan esa HU como para las personas no-técnicas (los muggles o techless).

Habitualmente se usa mockups o maquetas para dar forma a nuestra historia de usuario. Si bien es cierto, los mockups no siempre son tan útiles, sobre todo si trabajas con APIs o entornos con menor carga visual, así que una buena alternativa para enriquecer las HU está en el proceso BDD y, en particular, en el lenguaje Gherkin (Calatrava, 2021).

Python v.3.11.5

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo.

Los beneficios de Python incluyen los siguientes:

- Los desarrolladores pueden leer y comprender fácilmente los programas de Python debido a su sintaxis básica similar a la del inglés.

- Python permite que los desarrolladores sean más productivos, ya que pueden escribir un programa de Python con menos líneas de código en comparación con muchos otros lenguajes.
- Python cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los desarrolladores no tienen que escribir el código desde cero.
- Los desarrolladores pueden utilizar Python fácilmente con otros lenguajes de programación conocidos, como Java, C y C++.
- La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo. Si se presenta un problema, puede obtener soporte rápido de la comunidad.
- Hay muchos recursos útiles disponibles en Internet si desea aprender Python. Por ejemplo, puede encontrar con facilidad videos, tutoriales, documentación y guías para desarrolladores.
- Python se puede trasladar a través de diferentes sistemas operativos de computadora, como Windows, macOS, Linux y Unix (Amazon, 2019).

HTML 5

HTML es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto definiéndolo de forma sencilla, "HTML es lo que se utiliza para crear todas las páginas web de Internet". Más concretamente, HTML es el lenguaje con el que se "escriben" la mayoría de páginas web. Los diseñadores utilizan el lenguaje HTML para crear sus páginas web, los programas que utilizan los diseñadores generan páginas escritas en HTML y los navegadores que utilizamos los usuarios muestran las páginas web después de leer su contenido HTML. Aunque HTML es un lenguaje que utilizan los ordenadores y los programas de diseñó, es muy fácil de aprender y escribir por parte de las personas. En realidad, HTML son las siglas de HyperText Markup Language y más adelante se verá el significado de cada una de estas palabras (JIMÉNEZ ALVARADO, 2018).

CSS 3

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de

separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc (Eguiluz Pérez, 2008).

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java (Eguiluz, 2018).

Django v.4.2.5

Django es un marco web Python de alto nivel que fomenta un desarrollo rápido y un diseño limpio y pragmático. Creado por desarrolladores experimentados, se encarga de gran parte de las molestias del desarrollo web, por lo que puede concentrarse en escribir su aplicación sin necesidad de reinventar la rueda. Es gratis y de código abierto.

- Abstrae a los programadores de los problemas comunes del desarrollo web y acelera las tareas más frecuentes en la programación.

- A través de plantillas ayuda a separar el contenido de la presentación evitando tener que manipular la lógica de negocio cuando se necesite realizar cambios de apariencia en la página.
- Django incluye muchas aplicaciones comunes a todos los sitios web, como la autenticación de usuarios o la administración del contenido del sitio.
- Su arquitectura está inspirada en el patrón Modelo-Vista-Controlador (MVC), encargándose en gran parte de los controladores, y proveyendo herramientas para facilitar el desarrollo de las vistas y los modelos (Vincent, 2021).

Teniendo en cuenta estas características, se escoge este marco de trabajo en su versión 4.2.5 para la implementación del sistema, pues se centra en el desarrollo rápido, la reutilización y la seguridad.

Conclusiones parciales

En este capítulo, se ha establecido una sólida base teórica que sustenta la investigación en curso. Se han definido conceptos clave relacionados con el tema, se han explorado tecnologías y tendencias actuales, y se ha seleccionado una metodología de desarrollo adecuada, por lo tanto se arriba a las siguientes conclusiones:

1. La investigación de los principales conceptos relacionados a la búsqueda de la solución del trabajo de investigación y el estudio de los aplicaciones similares permitió identificar las funcionalidades que formarían parte de la aplicación a desarrollar.

2. Se ha proporcionado una visión general de las tecnologías actuales relevantes para el desarrollo de chatbots, incluyendo ChatSonic, Bard, Poe y Open AI ChatGPT. Estos ejemplos demuestran la diversidad de soluciones disponibles y cómo la inteligencia artificial está impulsando la interacción en línea.

3. Se definió SCRUM como metodología para guiar el proceso de desarrollo.

4. Se han identificado varias herramientas de desarrollo esenciales, como Google Colab, Visual Paradigm, Canva y lenguajes de programación como Python, HTML, CSS, JavaScript, y el framework como Django. Estas herramientas proporcionan los medios para construir y diseñar la solución de manera efectiva.

CAPÍTULO 2: DISEÑO E IMPLEMENTACIÓN DE LA PLATAFORMA DE CHATBOTS CONVERSACIONALES

Luego de haberse realizado un estudio a las soluciones existentes y seleccionado las herramientas, metodología y lenguaje de programación que se utilizará en la presente investigación, en el presente capítulo se presenta la propuesta de solución y los artefactos correspondientes a la metodología de desarrollo seleccionada (SCRUM) la cual servirá de guía para tener mayor organización en la distribución del tiempo, actividades y artefactos a desarrollar, confeccionándose las Historias de Usuario que proporcionan un mayor entendimiento y comprensión del sistema.

2.1 Descripción de la propuesta de solución

Utilizando la información recopilada en el capítulo anterior se propone el desarrollo de una plataforma que brinde a los usuarios la posibilidad de interactuar con chatbots, impulsados por diferentes modelos de lenguaje, altamente eficientes que responden a sus consultas y preguntas de manera instantánea.

La plataforma facilita a los usuarios la instauración de una comunicación sin interrupciones con todos los modelos previamente integrados en el sistema por defecto. Asimismo, posibilita la incorporación de un nuevo modelo GPT en el evento de su ausencia en la interfaz. También ofrece la facultad de realizar consultas a cada uno de los modelos mediante la formulación de una pregunta, generando múltiples respuestas y seleccionando la más pertinente en función de los criterios establecidos.

2.2 Product Backlog

La ingeniería de requisitos es el proceso de desarrollar una especificación de software. Las especificaciones pretenden comunicar las necesidades del sistema del cliente a los desarrolladores del sistema (Sommerville, 2016).

El producto backLog es una lista que reúne las funcionalidades deseadas por el cliente y elementos indispensables para garantizar el éxito de un proyecto. Su función es documentar todos los requisitos que van surgiendo desde el inicio y durante el desarrollo del proyecto. Es importante destacar que es una lista dinámica que está en constante evolución. Los requisitos se clasifican en dos grupos: requisitos funcionales y requisitos no funcionales.

Tabla 2 *Descripción del Product Backlog*

No.	Funcionalidad	Descripción	Complejidad	Prioridad
1	Enviar consulta	Permite al usuario enviar un mensaje a todos los modelos de lenguaje.	Alta	Alta
2	Insertar modelo de lenguaje	Permite insertar un nuevo modelo de lenguaje.	Media	Alta
3	Seleccionar modelo de lenguaje	Permite al usuario seleccionar el modelo de lenguaje con el que desea interactuar.	Media	Media
4	Iniciar chat	Permite al usuario iniciar una conversación con uno de los modelos de lenguaje.	Baja	Media
5	Regenerar	Permite al usuario regenerar las respuestas recibidas de los modelos.	Baja	Media
6	Página de inicio	Permite al usuario la navegación por la plataforma	Baja	Baja

2.2.2 Requisitos No Funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville, 2016).

Según la norma ISO/IEC 25010 el modelo de calidad representa la piedra angular en torno a la cual se establece el sistema para la evaluación de la calidad del producto. En este modelo se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado. La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor.

Tabla 3 *Requisitos No Funcionales*

No.	Requisitos	Descripción
RnF 1	Usabilidad	- El sistema podrá ser usado por personas con conocimiento básico de tecnología. - Tipo de aplicación informática: Aplicación WEB

RnF 2	Rendimiento	- Navegar sin retrasos, el portal no podrá demorarse demasiado en realizar las peticiones de los usuarios, debe tener la capacidad de responder de forma oportuna a cada una de las solicitudes del usuario sin generar contratiempos, pues esto determina el tiempo de respuesta. - La plataforma debe ser capaz de manejar una carga creciente de usuarios y conversaciones simultáneas sin degradación del rendimiento. - Los chatbots deben responder rápidamente a las preguntas de los usuarios para ofrecer una experiencia fluida y satisfactoria.
RnF 3	Software	- El sistema web será compatible con todos los navegadores web. - Sistema operativo Windows, Linux o MacOS
RnF 4	Hardware	- El sistema utilizará como mínimo un procesador Intel/AMD a 1.5 GHz - 4 GB de memoria RAM - 40 GB de almacenamiento interno disponible - Resolución de pantalla de 1.024 x 768 - Conexión a Internet
RnF 5	Soporte	- El sistema podrá ser actualizado con regularidad para su mejor funcionamiento. - En este sistema se podrá incorporar a lo largo del tiempo nuevas funcionalidades según las necesidades del usuario.
RnF 6	Restricciones del diseño e implementación	- El software será implementado usando el lenguaje de programación Python.
RnF 7	Apariencia o interfaz externa	- Usará protocolos de comunicación de paquetes básicos como TCP / IP y HTTPS.

2.3 Modelado de los Requisitos

2.3.1 Historias de Usuario

El primer paso de cualquier proyecto que siga la metodología SCRUM es definir las historias de usuario con el cliente. Las historias de usuario tienen la misma finalidad que los casos de uso pero con algunas diferencias: Constan de 3 ó 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados, etc. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia.
El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. No hay que preocuparse si en un principio no se identifican todas las historias de usuario. Al comienzo de cada iteración estarán registrados los cambios en las historias de usuario y según eso se planificará la siguiente iteración. Las historias de usuario son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración (Bustamante, 2014).

Descripción de los elementos de las Historias de Usuario establecidas por **el lenguaje Gherking**[2]**:**
Número: Indica el id del requisito funcional.
Nombre: Nombre del requisito funcional.
Usuario: Indica que usuario utilizara este requisito.
Prioridad en Negocio: Representa que requisitos deben implementarse primero. La prioridad se establece como: alta, media o baja, según la importancia de este requisito para el negocio.
Riesgo en Desarrollo: Evidencia el nivel de riesgo en caso de no realizarse la HU. El riesgo se establece como medio, alto o bajo.

[2] Ver epígrafe 1.5.2

Estimación: Este atributo no es más que una estimación hecha por el equipo de desarrollo del tiempo de duración en semanas de la HU.

Programador Responsable: Persona encargada de implementar el requisito.

Sprint Asignado: Determina en que sprint se comienza a implementar el requisito según su prioridad.

Funcionalidad: Funcionalidad

Descripción: Describe que espera el usuario de esta funcionalidad.

Escenario: una funcionalidad tiene uno o varios escenarios, que no son otra cosa que distintas características (que pueden estar relacionadas o no) que se tienen que desarrollar para poder conseguir la funcionalidad que se debe entregar a cliente. Los escenarios son elementos de tipo comportamiento.

Given: es el estado inicial del escenario, las precondiciones para que se puedan ejecutar x acciones. Es un elemento de tipo acción.

When: una acción específica que realiza el usuario, se trata de las condiciones de las acciones que se van a ejecutar y es un elemento de tipo acción.

Then: se trata del resultado esperado de las acciones ejecutadas, también es un elemento de tipo acción.

And: continúa cualquiera de las otras tres operaciones si es necesario

Tabla 4 *HU 1 Enviar Consulta*

Historia de Usuario	
Número: 1	**Nombre:** Enviar Consulta
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Estimación: 1	**Sprint Asignado:** 1
Programador Responsable: Alexander Rodríguez Londres	
FUNCIONALIDAD Enviar consulta **DESCRIPCIÓN:** **Como** usuario **Quiero** enviar un mensaje de consulta a todos los modelos de lenguaje **Para** poder escoger la respuesta que más se adecúe a lo que busco. **REGLAS Y ESPECIFICACIONES** - Esta funcionalidad se trata de enviar un mensaje de consulta a todos los modelos de lenguaje.	

- Esta funcionalidad se sitúa en un text-input (TI) dentro de un Web-Panel (WP) y se activa al cargar la página.
- Para que el botón regenerar se active se debe enviar una consulta a los modelos.
- La funcionalidad Regenerar se trata de reenviar un mensaje de consulta a todos los modelos de lenguaje para recibir mensajes diferentes.

CRITERIOS DE ACEPTACIÓN

#ESCENARIO 1 (HAPPY PATH)

Escenario: el usuario envía un mensaje de consulta y recibe múltiples respuestas exitosamente.

Given El usuario accede al WP

When hace clic sobre el TI y escribe una consulta

Then el WP se desplaza hacia arriba y en otro WP se muestran las respuestas obtenidas de cada modelo

And se activa el botón “Regenerar”

When hace clic sobre el botón “Regenerar”

Then se muestran diferentes respuestas.

#ESCENARIO 2 (NEGATIVO)

Escenario: el usuario envía un mensaje de consulta y no recibe respuestas de ningún modelo.

Given El usuario accede al WP

When hace clic sobre el Ti y escribe una consulta

Then el WP se desplaza hacia arriba y en el otro WP no se muestran las respuestas de los modelos

And se activa el botón “Regenerar” que permite cambiar las respuestas recibidas

When hace clic sobre el botón “Regenerar”

But no se muestra nada.

#ESCENARIO 3 (NEGATIVO)

Escenario: el usuario envía un mensaje de consulta y recibe múltiples respuestas exitosamente pero no se activa el botón “Regenerar”.

Given El usuario accede al WP

When hace clic sobre el Ti y escribe una consulta

Then el WP se desplaza hacia arriba y en otro WP se muestran las respuestas obtenidas de cada modelo

But no se activa el botón “Regenerar”.

Observaciones: N/A

Tabla 5 *HU 2 Insertar modelo de lenguaje*

Historia de Usuario	
Número: 2	**Nombre:** Insertar modelo de lenguaje
Usuario: Usuario del sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Estimación: 1	**Sprint Asignado:** 1
Programador Responsable: Alexander Rodríguez Londres	
FUNCIONALIDAD Insertar modelo **DESCRIPCIÓN** **Como** usuario **Quiero** insertar un nuevo modelo de lenguaje **Para** entablar una conversación con él. **REGLAS Y ESPECIFICACIONES** - Esta funcionalidad se trata de insertar un nuevo modelo de lenguaje en el sistema. - Esta funcionalidad se sitúa en un button (Bttn) “Insertar Modelo” en la parte inferior de un Web-Panel (WP) y se activa al cargar la página. **CRITERIOS DE ACEPTACIÓN** **#ESCENARIO 1 (HAPPY PATH)** **Escenario:** el usuario inserta un modelo exitosamente **Given** el usuario se sitúa dentro del WP **When** hace clic sobre el Bttn “Insertar Modelo” **Then** se muestra un formulario con campos vacíos con dos botones “Insertar” y “Cancelar” ubicados en la parte inferior del mismo **When** llena los campos dentro del formulario **Then** toca en el Bttn “Insertar” **And** se muestra un Popover con el mensaje “Insertado Correctamente” **And** se muestra en el WP el modelo insertado.	
Observaciones: N/A	

Tabla 6 *HU 3 Seleccionar modelo de lenguaje*

Historia de Usuario	
Número: 3	**Nombre:** Seleccionar modelo de lenguaje
Usuario: Usuario del sistema	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Estimación: 0.4	**Sprint Asignado:** 2

Programador Responsable: Alexander Rodríguez Londres
FUNCIONALIDAD Seleccionar modelo **DESCRIPCIÓN** **Como** usuario **Quiero** seleccionar un modelo de lenguaje **Para** iniciar un chat. **REGLAS Y ESPECIFICACIONES** - Esta funcionalidad se sitúa en el lado izquierdo del Web Panel (WP). **CRITERIOS DE ACEPTACIÓN** **#ESCENARIO 1 (HAPPY PATH)** **Escenario:** el usuario selecciona un modelo de lenguaje exitosamente **Given** el usuario se sitúa dentro del WP **When** toca el icono del lenguaje con el que desea chatear **Then** se muestra en la parte derecha del sitio el WP donde iniciar el chat con el modelo
Observaciones: N/A

Tabla 7 *HU 4 Iniciar chat*

Historia de Usuario	
Número: 4	**Nombre:** Iniciar Chat
Usuario: Usuario del sistema	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Estimación: 0.6	**Sprint Asignado:** 2
Programador Responsable: Alexander Rodríguez Londres	
FUNCIONALIDAD Iniciar chat **DESCRIPCIÓN** **Como** usuario **Quiero** iniciar un chat con un modelo de lenguaje **Para** entablar una conversación fluida. **REGLAS Y ESPECIFIACIONES** - Debe haberse seleccionado un modelo previamente. **CRITERIOS DE ACEPTACIÓN** **#ESCENARIO 1 (HAPPY PATH)** **Escenario:** el usuario inicia un chat correctamente **Given** el usuario seleccionó uno de los modelos de lenguaje **When** escribe un mensaje en el Text Input (Ti) dentro del Web Panel (WP) del chat	

And hace clic en el button (Bttn) de Enviar **Or** presiona la tecla Enter **Then** recibe una respuesta del modelo de lenguaje
Observaciones: N/A

Tabla 8 *HU 5 Página de Inicio*

Historia de Usuario	
Número: 5	**Nombre:** Página de Inicio
Usuario: Usuario del sistema	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Estimación: 0.2	**Sprint Asignado:** 3
Programador Responsable: Alexander Rodríguez Londres	
FUNCIONALIDAD Página de Inicio **DESCRIPCIÓN** **Como** usuario **Quiero** seleccionar la opción (duda) **Para** navegar por la herramienta **REGLAS Y ESPECIFICACIONES** - Esta funcionalidad debe contar con tarjetas (card) las cuales permitirán navegar hacia diferentes funcionalidades de la herramienta. **CRITERIOS DE ACEPTACIÓN** **#ESCENARIO 1 (HAPPY PATH)** **Escenario:** el usuario inicia un chat correctamente **Given** el usuario se sitúa sobre la card **When** el usuario hace clic sobre la card **Then** se abre la interfaz de la funcionalidad referente.	
Observaciones: N/A	

2.3.2 Estimación de esfuerzos por historias de usuario

Cada historia de usuario del product backlog debe ser estimada. Lo importante de la estimación es la realización de una reunión para ver qué tipo de dificultades, problemas y riesgos pueden surgir durante el sprint. Se suele tener en cuenta cuánto tiempo llevará su

desarrollo, cómo de difícil es y el grado de inseguridad para su implementación (Trigas Gallego, 2012).

Tabla 9 *Estimación de esfuerzos por HU*

No.	Historia de usuario	Estimación (semanas)
1	Enviar consulta	1
2	Insertar modelo de lenguaje	1
3	Seleccionar modelo de lenguaje	0.4
4	Iniciar chat	0.6
5	Inicio	0.2

La estimación total del esfuerzo es de 2.6 semanas.

2.3.3 SprintBacklog

La planificación de los sprint denominada también **Sprint Planning Meeting**, tiene como finalidad realizar una reunión con la intención de seleccionar de la lista Backlog del producto las funcionalidades sobre las que se va a trabajar, y que darán valor al producto (Trigas Gallego, 2012).

El plan de duración de los sprint se realiza luego de tener el estimado en semanas que demora implementar cada HU. Se tendrá en cuenta, además, la prioridad que el cliente le asigna a cada historia y el nivel de complejidad que estas poseen.

Tabla 10 *SprintBacklog*

Sprint	Historia de Usuario	Duración Total
1	Enviar consulta	2 semanas
	Insertar modelo de lenguaje	
2	Seleccionar modelo de lenguaje	1 semana

	Iniciar chat	
3	Página de Inicio	0.2 semanas

2.3.4 Plan de entregas

El cronograma de entregas establece las HU que serán agrupadas para conformar una entrega y el orden de las mismas. El cliente es quien las agrupa según su prioridad. El cronograma de entregas se analiza sobre la base de las estimaciones de tiempos de desarrollo.

Tabla 11 *Plan de entregas*

Historia de Usuario	Sprint	Entregable	Fecha
Enviar consulta	**1**	Versión 1.0	11/09/23
Insertar modelo de lenguaje		Versión 1.0	18/09/23
Seleccionar modelo de lenguaje	**2**	Versión 1.0	21/09/23
Iniciar chat		Versión 1.0	23/09/23
Página de Inicio	**3**	Versión 1.0	24/09/23

2.4 Diseño

2.4.1 Patrón Arquitectónico

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos, ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de

diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor (Pressman, 2014).

Modelo-Vista-Template (MVT)

El patrón Modelo-Vista-Template surge con el objetivo de reducir el esfuerzo de programación, necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de estandarizar el diseño de las aplicaciones. El patrón MVT es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los plantillas, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. A partir del uso de frameworks basados en el patrón MVT se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores.

- M (**Modelo**) es la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos; cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- P (**Plantilla**) es la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación; cómo algunas cosas son mostradas sobre una página web u otro tipo de documento.
- V (**Vista**) es la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada. Es el puente entre el modelo y las plantillas (Fernández Romero & Díaz González, 2012).

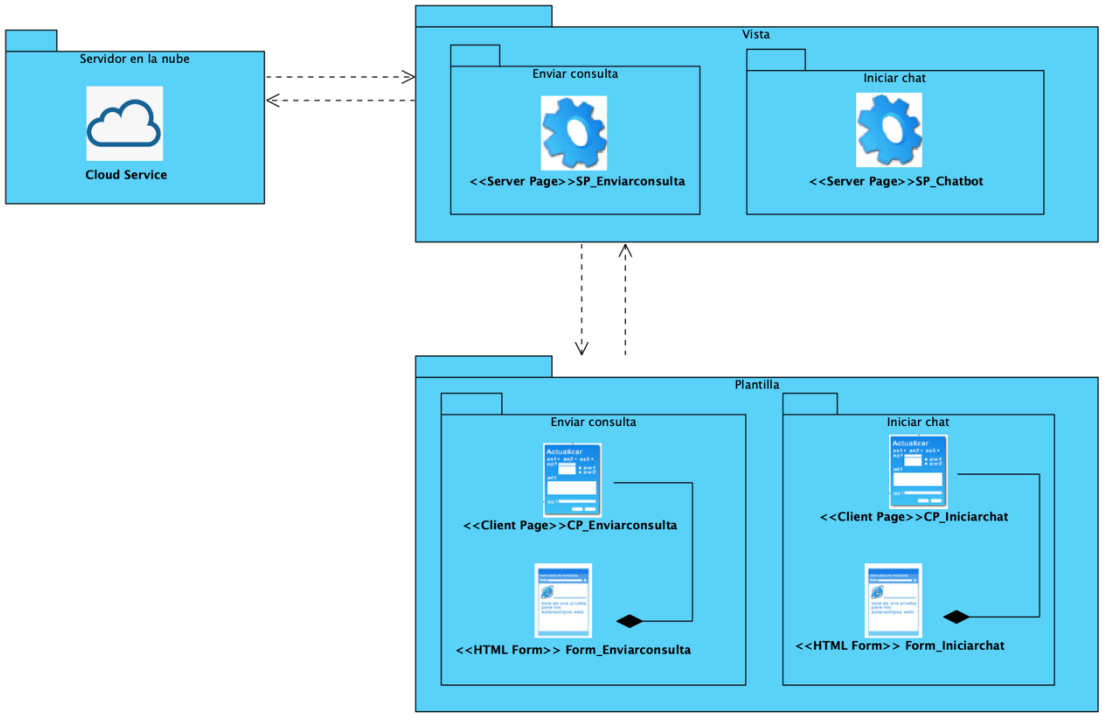


Figura 2 *Representación arquitectónica de la herramienta*

2.4.2 Mapa de navegación

Proporcionan una representación esquemática de la estructura del hipertexto, indicando los principales conceptos incluidos en el espacio de la información y las interrelaciones que existen entre ellos. La organización jerárquica de los vínculos ofrece una buena orientación a los usuarios pues la jerarquía es una forma de organización elemental (Rovira, 2001).

La plataforma de manera general estará definida según la estructura siguiente:

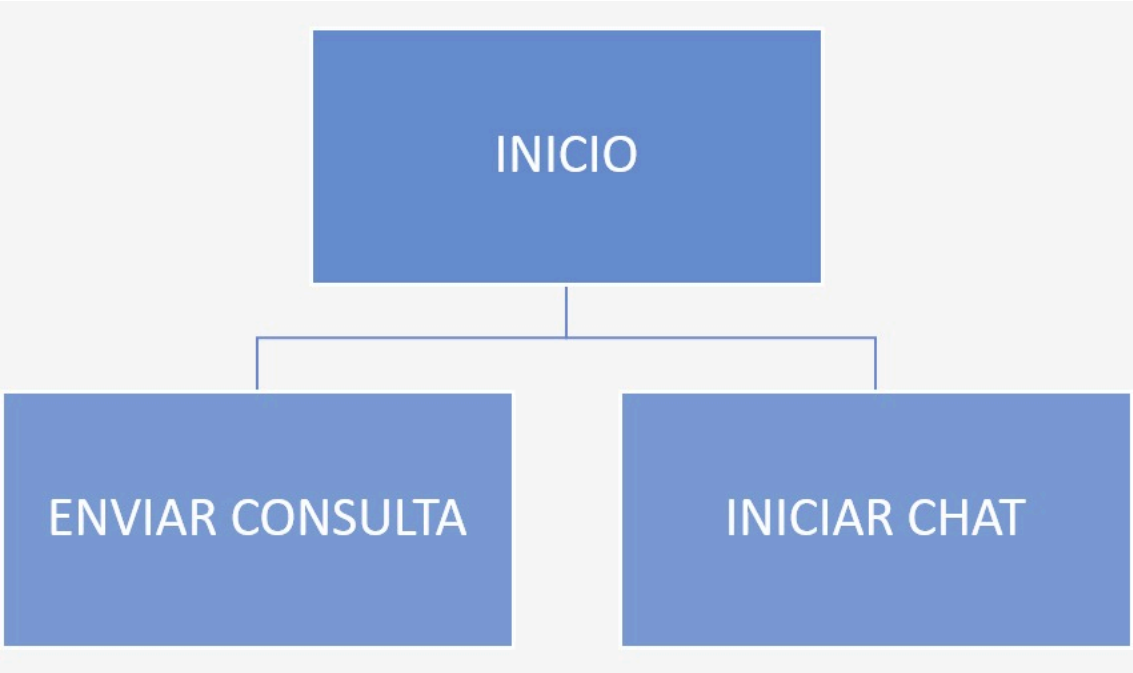


Figura 3 *Mapa de navegación*

2.4.3 Diseño de interfaz de usuario

El diseño de la interfaz de usuario crea un medio eficaz de comunicación entre los seres humanos y la computadora. Siguiendo un conjunto de principios de diseño de la interfaz, el diseño identifica los objetos y acciones de esta y luego crea una plantilla de pantalla que constituye la base del prototipo de la interfaz de usuario (Pressman, 2014).

A continuación, se muestra los prototipos de interfaz de usuario que debe tener el sistema a implementar:

2.4.3.1 Página de inicio:

La página de bienvenida del sistema contiene los siguientes elementos:

- Tarjeta para acceder a la página de consultas a los modelos.
- Tarjeta para acceder a la página de chats.

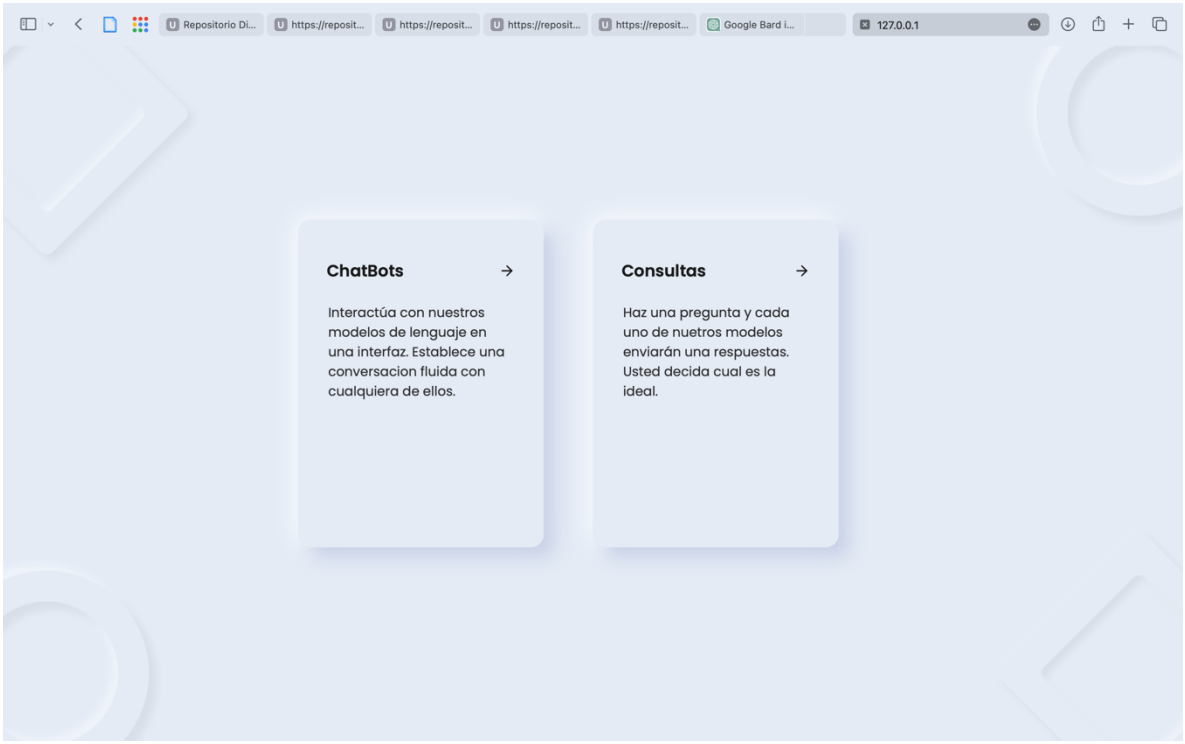


Figura 4 *Diseño de interfaz de la página de inicio*

2.4.3.2 Chatbots

La página de chatbots contiene los siguientes elementos:

- Un web panel donde se muestran los modelos de lenguaje disponibles para iniciar una conversación.
- Un web panel para chatear con los modelos.

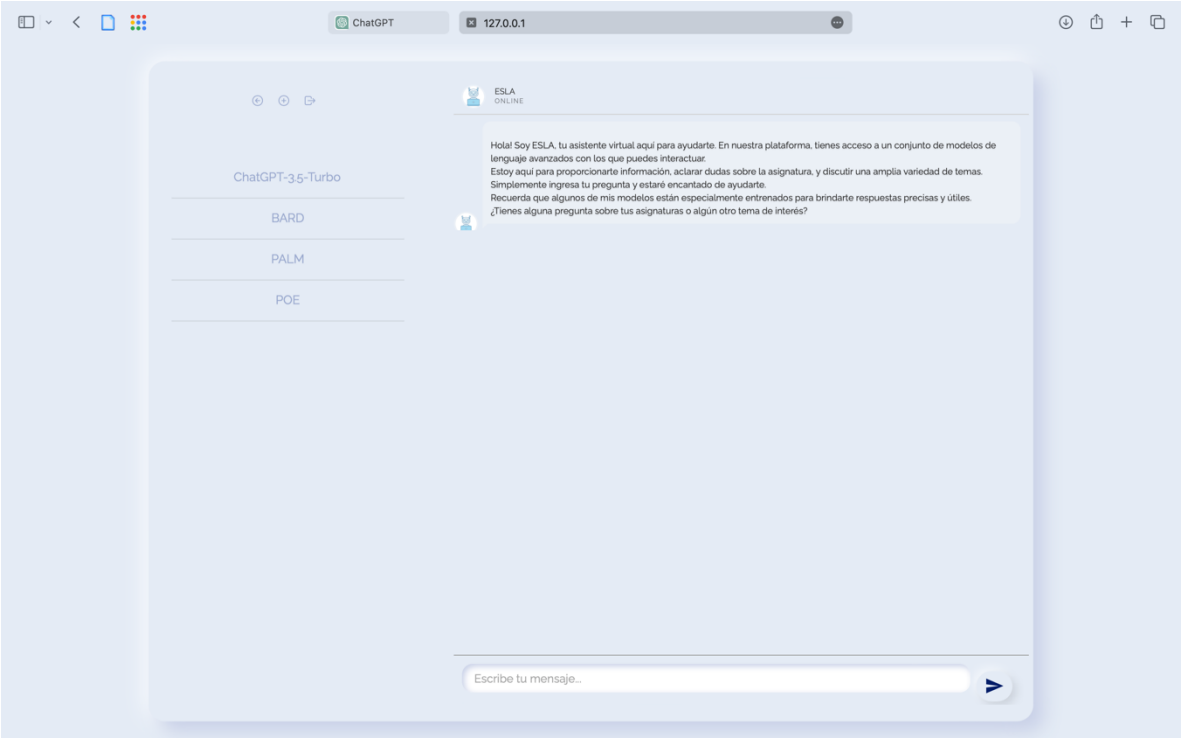


Figura 5 *Diseño de interfaz de la página de chatbots*

2.4.3.3 Consultas

La página de consultas contiene los siguientes elementos:

- Un submit donde se introduce el mensaje de consulta.
- Varias tarjetas donde se recibirán las respuestas de los modelos.

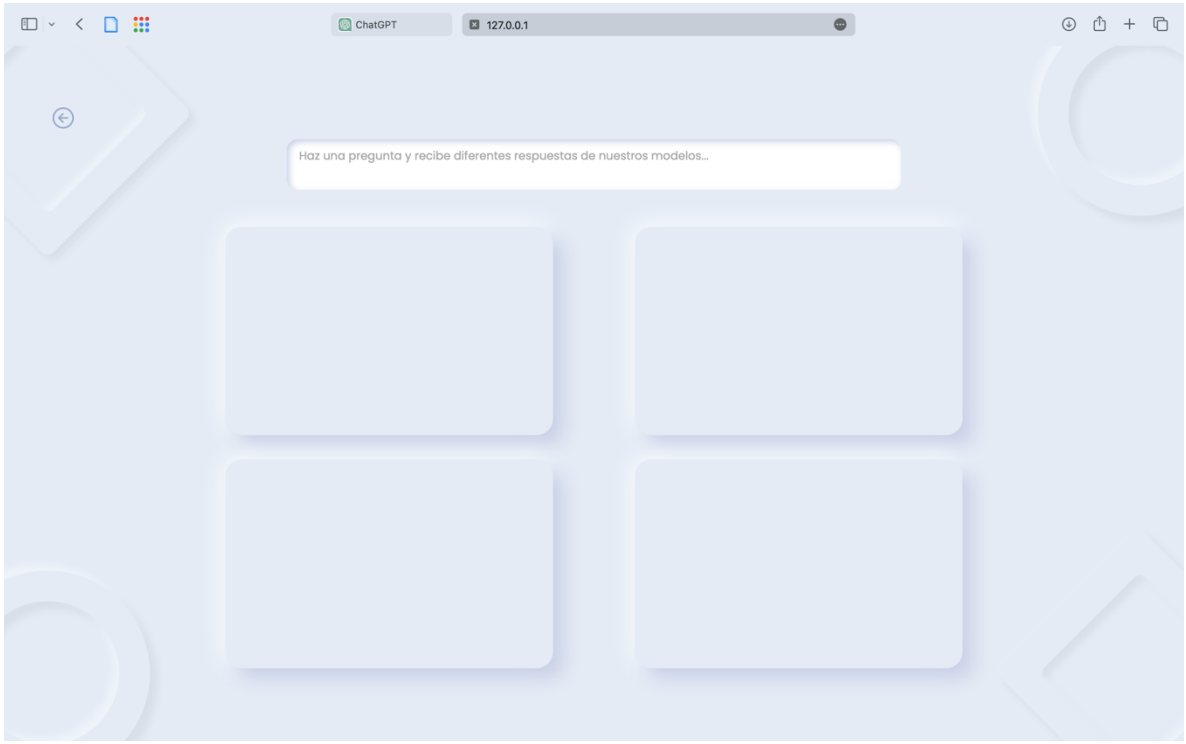


Figura 6 *Diseño de interfaz de la página de consultas*

Conclusiones Parciales

Al finalizar el desarrollo del presente capítulo se llegan a las siguientes conclusiones:

- Se han identificado las características clave, así como los requisitos tanto funcionales como no funcionales que son esenciales para la creación exitosa de la plataforma. Estos elementos se han organizado de manera efectiva en forma de historias de usuario, lo que proporciona una visión clara de las necesidades del proyecto.

- La adopción de una arquitectura bien definida y la aplicación de patrones de diseño específicos desempeñan un papel crucial en la simplificación y eficiencia del proceso de desarrollo de la solución propuesta. Estos enfoques arquitectónicos y de diseño establecidos permiten una implementación más coherente y robusta, asegurando que el portal cumpla con los objetivos planteados. Además,

- Los artefactos generados en este capítulo, como las historias de usuario y las directrices arquitectónicas, servirán como una guía esencial durante la fase de

implementación. Proporcionarán un marco sólido y orientación práctica para el equipo de desarrollo, lo que aumentará la eficiencia y la calidad del producto final.

CAPÍTULO 3: VALIDACIÓN DE LA PLATAFROMA DE CHATBOTS CONVERSACIONALES

En este capítulo se aborda sobre los estándares de codificación, las diferentes pruebas realizadas al software y que son exigidas para el buen funcionamiento y calidad de la propuesta desarrollada. Se detalla con exactitud los diferentes resultados alojados por cada prueba.

3.1 Implementación

3.1.1 Diagrama de Despliegue

El diagrama de despliegue permite modelar la disposición física o la topología de un sistema, muestra las conexiones físicas entre el hardware y las relaciones entre componentes, muestra también el hardware usado y los componentes instalados en el hardware (Larman, 2003).

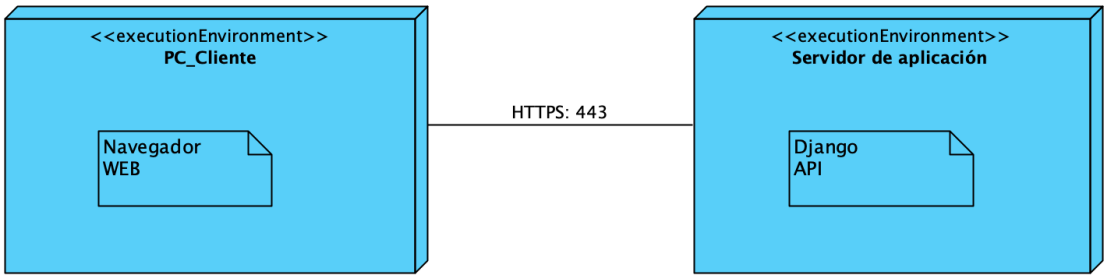


Figura 7 *Diagrama de despliegue*

A continuación, se muestra el diagrama de despliegue correspondiente a la propuesta solución desarrollada:

PC-Cliente: Es el nodo que representa la estación de trabajo que permite al usuario mediante el protocolo HTTPS y el puerto 443, acceder a la plataforma.

Servidor de Aplicaciones: Es el nodo encargado de tener instalado el sistema al que tendrán acceso los usuarios desde las estaciones de trabajo, con el framework Django y usando los servicios de la API.

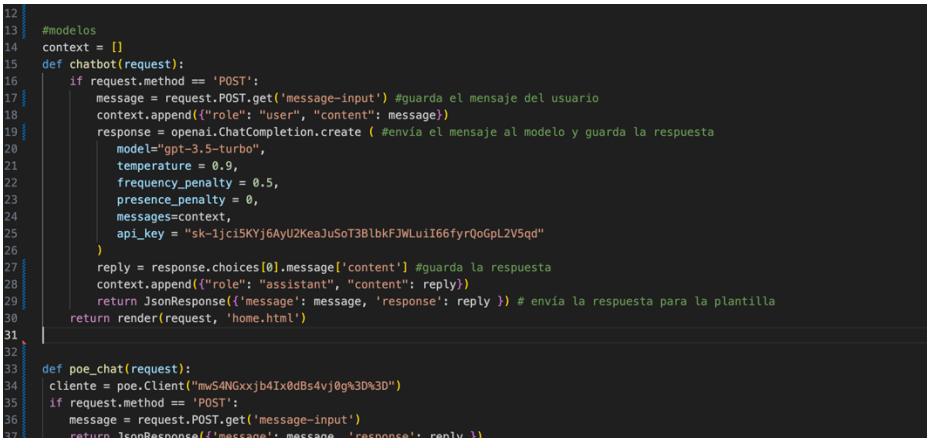
3.1.2 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez (Martínez Mengual, 2016 pág. 54).

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código (ERP, 2008). Para facilitar el entendimiento del código y establecer un modelo a seguir, se establecieron los estándares de codificación mostrados a continuación para el lenguaje de programación Python en el framework de desarrollo Django.

Estructura

- Las líneas podrán tener ochenta caracteres o menos.
- Las funciones de alto nivel y definiciones de clases se separan con dos líneas como se muestra
- Se añaden comentarios descriptivos junto a cada declaración de variables, si es necesario como se muestra en la Figura 8.



```
#modelos
context = []
def chatbot(request):
    if request.method == 'POST':
        message = request.POST.get('message-input') #guarda el mensaje del usuario
        context.append({"role": "user", "content": message})
        response = openai.ChatCompletion.create ( #envía el mensaje al modelo y guarda la respuesta
          model="gpt-3.5-turbo",
          temperature = 0.9,
          frequency_penalty = 0.5,
          presence_penalty = 0,
          messages=context,
          api_key = "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"
        )
        reply = response.choices[0].message['content'] #guarda la respuesta
        context.append({"role": "assistant", "content": reply})
        return JsonResponse({'message': message, 'response': reply }) # envía la respuesta para la plantilla
    return render(request, 'home.html')

def poe_chat(request):
 cliente = poe.Client("mwS4NGxxjb4Ix0dBs4vj0g%3D%3D")
 if request.method == 'POST':
    message = request.POST.get('message-input')
```

Figura 8 *Ejemplo de estándar de codificación: separación entre declaraciones de funciones, comentarios descriptivos y definiciones de clases*

Importaciones

- Las importaciones siempre estarán colocadas al comienzo del archivo y deberán estar agrupadas por (ver Figura 9):
  - Importaciones de biblioteca estándar.
  - Importaciones terceras relacionadas.
  - Importaciones locales de la aplicación.

proyecto > tasks > views.py

```
from django.shortcuts import redirect, render
from django.http import HttpResponse, JsonResponse
from django.contrib.auth.decorators import login_required
from django.contrib import auth
from django.contrib.auth.models import User
from django.contrib.auth import login, authenticate
import openai
from openai import ChatCompletion
import poe
import torch
from transformers import MBart50TokenizerFast, BartForConditionalGeneration
```

Figura 9 *Ejemplo de estándar de codificación: orden de las importaciones*

Nombres de clases y funciones

- Se utiliza para el nombramiento de funciones letras minúsculas separando las palabras con guiones bajos, como se muestra en la Figura 10.

```
def poe_chat(request):
 cliente = poe.Client("mwS4NGxxjb4Ix0dBs4vj0g%3D%3D")
 if request.method == 'POST':
    message = request.POST.get('message-input')
    return JsonResponse({'message': message, 'response': reply })
 return render(request, 'home.html')
```

Figura 10 *Ejemplo de estándar de codificación: nombramiento de funciones*

3.2 Estrategia de prueba

Una estrategia de pruebas de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos requieren. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados (Pressman, 2014).

Con el propósito de valorar el funcionamiento de la plataforma se decide emplear una estrategia de pruebas que permita comenzar la etapa de valoración desde el nivel más bajo de la aplicación, es decir, el código en sí. Se propone realizar la siguiente estrategia de pruebas:

- **Pruebas de aceptación:** Con estas pruebas se busca evaluar la plataforma mediante modelos de pruebas. El objetivo es validar que la plataforma cumple con el funcionamiento esperado. Estos son comprobados por el usuario de forma tal que sea este el que determine el grado de aceptación que tiene con respecto a las funcionalidades y el rendimiento del producto.

- **Pruebas unitarias:** Con el propósito de valorar el funcionamiento de la propuesta de solución se decide emplear una estrategia de pruebas que permita comenzar la etapa de valoración desde el nivel más bajo de la aplicación, es decir, el código en sí. Para esto se decide implementar y ejecutar las pruebas unitarias, debido a que estas aseguran la comprobación del correcto funcionamiento de los métodos y funciones de forma aislada.

- **Pruebas de rendimiento:** se usan para descubrir problemas de rendimiento que pueden ser resultado de la falta de recursos en el lado servidor, red con ancho de banda inadecuada, capacidades de base de datos inadecuadas, capacidades de sistema operativo deficientes o débiles.

- **Pruebas de precisión:** permiten verificar la coherencia y exactitud de las salidas generadas por el sistema, asegurando su adecuación para cumplir con los estándares de calidad y las expectativas del usuario. La precisión en las pruebas es esencial, especialmente en áreas como la inteligencia artificial, la programación y el análisis de datos, donde la exactitud de las respuestas es fundamental para la toma de decisiones y la funcionalidad efectiva.

3.3 Aplicación de las pruebas

3.3.1 Pruebas unitarias

Las pruebas unitarias son fragmentos de código que prueban otras unidades de código de una aplicación, normalmente funciones aisladas, clases. Cuando una aplicación supera todas sus pruebas unitarias, al menos puede confiar en que su función de bajo nivel es correcta.

Python utiliza pruebas unitarias ampliamente para validar escenarios durante el diseño de un programa. La compatibilidad de Python en Visual Studio Code incluye características para descubrir, ejecutar y depurar pruebas unitarias dentro del contexto de su proceso de desarrollo, sin tener que ejecutar pruebas independientemente.

El módulo *unittest* de Python proporciona un conjunto de herramientas para construir y ejecutar pruebas. La infraestructura de tests unitarios *unittest* se inspiró en primera instancia en JUnit y ofrece aspectos similares a las principales estructuras de tests unitarios más importantes de otros lenguajes. Da soporte a automatización de tests, inicialización compartida, código de cierre de los tests, agregación de los tests en colecciones e independencia de los tests de la infraestructura que los reporta.

A continuación, se muestra la realización de estas pruebas:

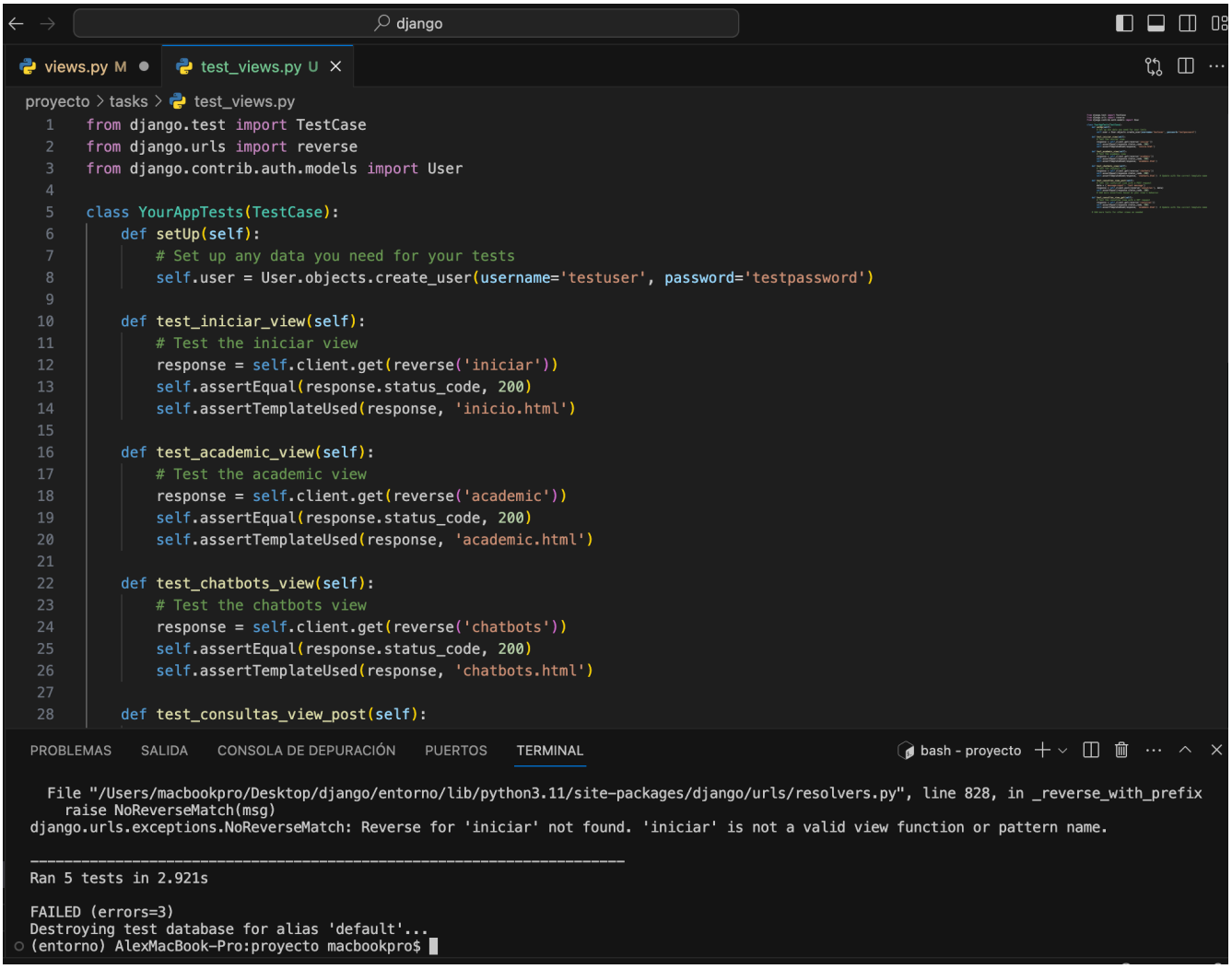


Figura 11 *Resultados de las pruebas unitarias automatizadas*

Luego de la ejecución de las pruebas, se registró la realización de un total de 5 pruebas, identificándose que 3 de ellas exhibieron fallos relacionados con la gestión de las URL y la autenticación mediante tokens para acceder a las interfaces de programación de aplicaciones (API). Subsiguientemente, se llevaron a cabo las acciones y modificaciones pertinentes para remediar estos inconvenientes.

3.2.2 Pruebas de rendimiento

Las pruebas de rendimiento se realizan para determinar lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo. También puede servir para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos. Las pruebas de rendimiento son un subconjunto de la ingeniería de pruebas, una práctica informática que se esfuerza

por mejorar el rendimiento, englobándose en el diseño y la arquitectura de un sistema, antes incluso del esfuerzo inicial de la codificación (Molyneaux, 2009). Para probar el rendimiento del sistema se realizaron pruebas de carga y estrés.

Pruebas de carga y estrés

Las pruebas de carga son diseñadas para determinar y validar la respuesta de la aplicación cuando es sometida a una carga de usuarios y/o transacciones que se espera en el ambiente de producción. Entre las dimensiones de calidad de esta prueba se encuentra la de rendimiento. Dichas pruebas se esbozan para asegurar que el sistema pueda procesar una carga esperada. Esto normalmente implica planificar un grupo de pruebas en la que la carga se va incrementado regularmente hasta que el rendimiento del sistema se hace inaceptable. Se ocupan tanto de demostrar que el sistema satisface los requerimientos como de descubrir defectos y problemas en el mismo.

Por su parte, las pruebas de estrés son creadas para encontrar el volumen de datos o de tiempo en que la aplicación comienza a fallar o es incapaz de responder a las diferentes peticiones. La fiabilidad, es una de las dimensiones de calidad de este tipo de pruebas (Sommerville, 2016).

Para la realización de este tipo de pruebas se utiliza una PC con un procesador Intel Core i7 quad core una frecuencia de 2.9 GHz, memoria RAM de 16GB y sistema operativo MacOS.

A continuación, se muestran las variables analizadas:

- **Muestra:** Cantidad de peticiones realizadas para cada URL.
- **Media:** Tiempo promedio en milisegundos en el que se obtienen los resultados.
- **Mediana:** Tiempo en milisegundos en el que se obtuvo el resultado que ocupa la posición central.
- **Min:** Tiempo mínimo que demora un hilo en acceder a una página.
- **Max:** Tiempo máximo que demora un hilo en acceder a una página.
- **Línea 90 %:** Máximo tiempo utilizado por el 90 % de la muestra, al resto de la misma le llevo más tiempo.
- **% Error:** Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria. **Rendimiento (Rend):** El rendimiento se mide en cantidad de solicitudes por segundo. **Kb/s:** Velocidad de carga de las páginas.

Tabla 12 Resultado de las pruebas de Carga y estrés

Muestra	Media (ms)	Mediana (ms)	Min (ms)	Max (ms)	Línea 90% (ms)	% Error	Rendimiento (solicitudes por segundo)	Kb/s (velocidad de carga)
100	150	140	120	210	190	5%	30	500
150	190	247690	20745.30	73	48935	2193	1.25%	15

Resultados de las pruebas de carga y estrés realizadas mediante la herramienta JMeter

Las pruebas de carga realizadas muestran que el sistema es capaz de responder a 10 usuarios virtuales que generan un total de 100 solicitudes de chatbots durante un período de 20 minutos proporciona una visión representativa de diversas métricas de rendimiento críticas. Los datos recopilados permiten evaluar el funcionamiento de la plataforma, desde el tiempo promedio de respuesta hasta la velocidad de carga de las páginas y el porcentaje de errores. Estos resultados son fundamentales para la toma de decisiones, identificando áreas de mejora y optimización necesarias para garantizar un rendimiento óptimo en condiciones de carga.

Por último, y con el objetivo de analizar el comportamiento de la plataforma en condiciones extremas, se realizó una prueba de estrés para un conjunto de 150 usuarios conectados simultáneamente. En este caso, la plataforma responde a las 190 peticiones en un tiempo promedio de 247690 milisegundos (24.7 segundos aproximadamente), pero con un porciento de error de 1.25%. Este resultado está estrechamente relacionado al entorno donde se realizó la prueba, el cual no es un servidor dedicado sino un cliente habilitado para realizar la prueba.

3.2.3 Pruebas de precisión

Con el objetivo de tener una medida de la calidad de la información que se puede obtener con el uso de la herramienta propuesta se diseñó un caso de estudio para medir la precisión de la herramienta al extraer la información de los datos. Estas pruebas se realizan mediante la comparación de las respuestas generadas por los chatbots con respuestas esperadas previamente definidas.

Tabla 13 *Pruebas de precisión*

Pregunta	Respuesta Esperada	Respuesta del Chatbot	Coincide con la Esperada
¿Cuál es el método de ordenamiento Bubble Sort?	static void bubbleSort(int[] arr) { int n = arr.length; for (int i = 0; i < n - 1; i++) { for (int j = 0; j < n - i - 1; j++) { if (arr[j] > arr[j + 1]) { // Realizar el intercambio int temp = arr[j]; arr[j] = arr[j + 1]; arr[j + 1] = temp; } } } }	Claro, el algoritmo de ordenamiento de burbuja (Bubble Sort) es un método simple de ordenación que compara repetidamente cada par de elementos adyacentes y los intercambia si están en el orden incorrecto. Aquí tienes una implementación básica en Java: ```java public class BubbleSort { public static void main(String[] args) { int[] arreglo = {64, 34, 25, 12, 22, 11, 90}; System.out.println("Arreglo antes de ordenar:"); imprimirArreglo(arreglo); bubbleSort(arreglo); System.out.println("\nArreglo ordenado:"); imprimirArreglo(arreglo); } // Método para realizar el ordenamiento de burbuja static void bubbleSort(int[] arr) { int n = arr.length; for (int i = 0; i < n - 1; i++) { for (int j = 0; j < n - i - 1; j++) {	Sí

Pregunta	Respuesta Esperada	Respuesta del Chatbot	Coincide con la Esperada
		// Compara elementos adyacentes y los intercambia si están en el orden incorrecto if (arr[j] > arr[j + 1]) { // Realizar el intercambio int temp = arr[j]; arr[j] = arr[j + 1]; arr[j + 1] = temp; } } } } // Método para imprimir un arreglo static void imprimirArreglo(int[] arr) { for (int i = 0; i < arr.length; i++) { System.out.print(arr[i] + " "); } } } ``` Este código Java implementa el algoritmo de ordenamiento de burbuja en el método `bubbleSort()`. La función `imprimirArreglo()` se utiliza para imprimir el arreglo antes y después del ordenamiento. Ten en cuenta que el Bubble Sort no es la opción más eficiente para grandes conjuntos de datos, pero es útil para	

Pregunta	Respuesta Esperada	Respuesta del Chatbot	Coincide con la Esperada
		propósitos educativos y para conjuntos de datos pequeños. Otros algoritmos de ordenamiento, como QuickSort o MergeSort, son más eficientes para conjuntos de datos más grandes.	
¿En qué se diferencia Quick Sort de Merge Sort?	Quick Sort y Merge Sort son dos algoritmos de ordenamiento. La principal diferencia es su enfoque: Quick Sort utiliza el enfoque de divide y conquista y selecciona un elemento como pivote para dividir la lista en dos sublistas, mientras que Merge Sort divide la lista en sublistas más pequeñas, las ordena y las fusiona.	Quick Sort y Merge Sort son algoritmos de ordenamiento. La principal diferencia es que Quick Sort selecciona un pivote y divide la lista en sublistas, mientras que Merge Sort divide la lista en dos mitades, las ordena y las fusiona.	No
¿Cómo funciona el algoritmo de búsqueda binaria?	La búsqueda binaria es un algoritmo de búsqueda eficiente que opera en listas ordenadas. Comienza comparando el elemento medio de la lista con el valor buscado y ajusta el rango de búsqueda en función de si el valor es mayor o menor que el elemento medio. El proceso se repite hasta que se encuentra el elemento o se determina que no está en la lista.	La búsqueda binaria es un algoritmo que divide una lista ordenada en dos mitades y compara el elemento medio con el valor buscado. Si el valor es mayor, busca en la mitad derecha; si es menor, busca en la mitad izquierda.	Sí
¿Cuándo se usa el algoritmo de ordenamiento Radix Sort?	Radix Sort se utiliza para ordenar números enteros o cadenas de texto al comparar dígitos o caracteres de posición	Radix Sort es un algoritmo de ordenamiento que se utiliza para ordenar números enteros.	No

Pregunta	Respuesta Esperada	Respuesta del Chatbot	Coincide con la Esperada
	en posición. Es especialmente útil cuando se tienen números con la misma cantidad de dígitos, como números enteros de igual longitud.		

El chatbot demostró ser capaz de proporcionar respuestas precisas y concisas en algunos casos, como la explicación de Bubble Sort. Sin embargo, se identificaron áreas de mejora en la plataforma, especialmente en lo que respecta a la diferenciación entre Quick Sort y Merge Sort, así como en la explicación del algoritmo Radix Sort, que requiere una corrección para brindar información más precisa. Estas pruebas de precisión son esenciales para garantizar que la plataforma sea una fuente confiable de conocimientos en el ámbito de la programación y los algoritmos, y se recomienda una revisión y ajuste cuidadoso de las respuestas para lograr una mayor precisión y calidad en las interacciones con los usuarios.

3.2.4 Pruebas Funcionales

En este tipo de pruebas se ejecutan los distintos servicios prestados con datos correctos e incorrectos. En caso de que los datos sean incorrectos se verifica que los mensajes de error sean los deseados y en el caso opuesto que los resultados sean los esperados (Pressman, 2014).

Pruebas de aceptación

Las pruebas de aceptación son importantes dado que significan la satisfacción del cliente con el producto desarrollado, el final de una iteración y el comienzo de la siguiente. Se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema sufra (Pressman, 2014). Estas son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales (Carrillo, 2011).

Las pruebas funcionales y las pruebas de aceptación comparten un diseño común de casos de prueba como parte de su enfoque integral para validar el sistema en desarrollo. Ambos tipos de pruebas se caracterizan por emplear casos de prueba que son meticulosamente elaborados para evaluar la funcionalidad y el rendimiento del sistema frente a los requisitos y criterios de aceptación establecidos.

Este enfoque unificado en el diseño de casos de prueba se fundamenta en la premisa de garantizar la coherencia y la exhaustividad en la evaluación del sistema. Los casos de prueba son concebidos de manera estructurada para abordar tanto los aspectos funcionales específicos de la plataforma como los criterios de aceptación definidos por las partes interesadas.

La convergencia en el diseño de casos de prueba entre las pruebas funcionales y las pruebas de aceptación promueve una eficiente cobertura de los requisitos de la plataforma y facilita la trazabilidad entre los escenarios de prueba y los criterios de aceptación del usuario final. Esta integración estratégica contribuye a la identificación temprana de posibles incongruencias entre el comportamiento del sistema y las expectativas del usuario, optimizando así el proceso de validación y asegurando la entrega de un producto de software robusto y conforme a las expectativas establecidas.

A continuación, se presenta el diseño de caso de prueba que se utilizará para comprobar el funcionamiento del sistema:

CU: Número de la HU a la cual pertenece.

Nombre: Junto al código conforma el identificador del caso de prueba.

Descripción: Acción que debe realizar el sistema.

Condiciones de ejecución: Describe las características y elementos que debe contener el sistema para realizar el caso de prueba.

Entrada/Pasos de Ejecución: Incluye las entradas necesarias para realizar el sistema, ade- más de los pasos para realizar el caso de prueba.

Resultados Esperados: Descripción de la respuesta del sistema ante el caso de prueba.

Evaluación de la prueba: Clasificación de la prueba en satisfactoria o insatisfactoria.

Tabla 14 *Caso de prueba de aceptación #1*

Caso de Prueba de Aceptación	
Código: HU1-P1	**Historia de Usuario:** HU-1
Nombre: Enviar consulta	
Descripción: El usuario puede enviar una consulta y recibir múltiples respuestas de diferentes modelos exitosamente.	
Condiciones de ejecución: El usuario debe tener acceso a internet.	
Entradas/Pasos de ejecución: El usuario escribe un mensaje de consulta, toca el botón Enviar o presiona la tecla “Enter” y luego se muestran las respuestas recibidas de los modelos. Se activa el botón Regenerar.	
Resultados esperados: El usuario recibe múltiples respuestas de los modelos.	

Evaluación de la prueba: Satisfactoria

Tabla 15 *Caso de prueba de aceptación #2*

Caso de Prueba de Aceptación	
Código: HU2-P2	**Historia de Usuario:** HU-2
Nombre: Insertar modelo de lenguaje	
Descripción: El usuario inserta un nuevo modelo de lenguaje.	
Condiciones de ejecución: El usuario debe tener acceso a internet.	
Entradas/Pasos de ejecución: El usuario toca el botón “Insertar Modelo” y se muestra un formulario con los botones “Insertar” y “Cancelar” activos. Una vez llene los campos, toca en “Insertar” y se muestra una notificación indicando el registro exitoso.	
Resultados esperados: El usuario inserta un nuevo modelo de lenguaje	
Evaluación de la prueba: Satisfactoria	

Tabla 16 *Caso de prueba de aceptación #3*

Caso de Prueba de Aceptación	
Código: HU3-P3	**Historia de Usuario:** HU-3
Nombre: Seleccionar modelo de lenguaje	
Descripción: El usuario puede seleccionar un modelo de lenguaje.	
Condiciones de ejecución: El usuario debe tener acceso a internet.	
Entradas/Pasos de ejecución: El usuario se desplaza sobre el panel y selecciona el modelo de lenguaje que desee.	
Resultados esperados: El usuario selecciona un modelo de lenguaje.	
Evaluación de la prueba: Satisfactoria	

Tabla 17 *Caso de prueba de aceptación #4*

Caso de Prueba de Aceptación	
Código: HU4-P4	**Historia de Usuario:** HU-4
Nombre: Iniciar chat	
Descripción: El usuario puede establecer una conversación fluida con un modelo de lenguaje.	

Condiciones de ejecución: El usuario debe haber seleccionado un modelo de lenguaje previamente.
Entradas/Pasos de ejecución: El usuario escribe un mensaje y toca el botón “Enviar” o presiona la tecla “Enter” para enviarlo. Una vez enviado recibe la respuesta del modelo.
Resultados esperados: El usuario inicia una conversación fluida con un modelos de lenguaje
Evaluación de la prueba: Satisfactoria

Tabla 18 *Caso de prueba de aceptación #5*

Caso de Prueba de Aceptación	
Código: HU5-P5	**Historia de Usuario:** HU-5
Nombre: Página de inicio	
Descripción: El usuario visualiza la página de inicio de la plataforma.	
Condiciones de ejecución: El usuario debe tener acceso a internet.	
Entradas/Pasos de ejecución: El usuario accede a la plataforma y selecciona la opción que desea realizar, la cual le permite la navegación hacia una funcionalidad de la herramienta.	
Resultados esperados: El usuario visualiza la página de inicio de la plataforma.	
Evaluación de la prueba: Satisfactoria	

Resultado de las pruebas de aceptación

Para la validación de los requisitos funcionales se realizaron 3 iteraciones. A continuación, se muestra la clasificación de las no conformidades detectadas, teniendo en cuenta el impacto que tienen las mismas en la solución.

En la Figura 12 se muestran los resultados obtenidos en cada una de las iteraciones de pruebas realizadas a la aplicación. Como se puede apreciar en la primera iteración fueron detectadas 3 no conformidades (NC) de las cuales fueron 2 de prioridad alta y 1 de media. Luego en la segunda iteración se encontraron 3 nuevas NC, de las cuales fueron 2 de prioridad baja y 1 de alta, siendo resueltas las mismas. En cada iteración se realizaron pruebas de regresión para comprobar que las no conformidades detectadas estaban corregidas.

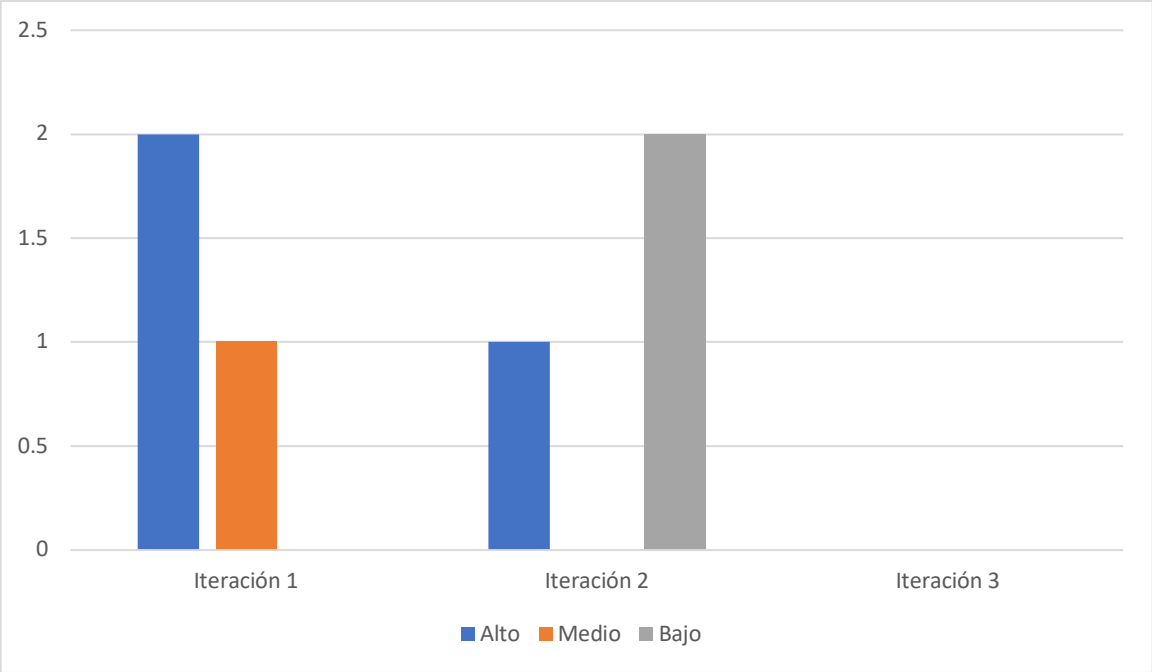


Figura 12 *Resultados de las pruebas de aceptación*

Conclusiones parciales

Como resultado del desarrollo de este capítulo, se generó el diagrama de despliegue, así como la descripción de los casos de prueba con el objetivo de que los mismos sean usados para verificar el correcto funcionamiento de la plataforma. De modo general, se puede afirmar que una vez aplicadas las pruebas los resultados fueron satisfactorios ya que se identificaron a tiempo los problemas presentados por la misma y se les suministró solución de forma inmediata.

CONCLUSIONES GENERALES

Luego de desarrollar el presente trabajo y analizar los resultados obtenidos, las conclusiones a las que se arriban son las siguientes:

- ⇒ La investigación y análisis de conceptos estableció las bases, identificando las funcionalidades esenciales de la aplicación.

- ⇒ Se exploraron las diversas tecnologías actuales relevantes para el desarrollo de chatbots, enfatizando el impacto de la inteligencia artificial en las interacciones en línea.

- ⇒ La selección de SCRUM como guía metodológica proporciona una estructura colaborativa y eficiente para el proceso de desarrollo.

- ⇒ Se destacó las herramientas esenciales, desde Google Colab hasta lenguajes de programación fundamentales para la construcción de la solución. Se delinearon los requisitos clave y la arquitectura de la plataforma, resaltando la importancia de una implementación coherente y robusta.
- ⇒ Las pruebas de software aplicadas permitieron validar el correcto funcionamiento de la propuesta de solución.

Estas conclusiones fusionadas representan una comprensión profunda y un plan sólido para el desarrollo exitoso de la plataforma, asegurando una base fuerte y una hoja de ruta clara para futuras investigaciones y aplicaciones.

RECOMENDACIONES

Se recomienda incorporar un sistema de autenticación robusto y seguro a la plataforma con el propósito de habilitar la funcionalidad de almacenamiento del historial de chats a nivel de usuario. Asimismo, se sugiere la implementación de un mecanismo que permita a los usuarios calificar las respuestas proporcionadas por los chatbots, lo que contribuirá a la mejora continua de la calidad y precisión de las respuestas. Además, se aconseja la adición de una funcionalidad que permita la regeneración individual de la respuesta de un chatbot en respuesta a una consulta específica, ofreciendo a los usuarios un mayor control y flexibilidad en su experiencia interactiva.

Estas recomendaciones enriquecerán significativamente la funcionalidad de la plataforma, optimizando la interacción y la satisfacción del usuario.

REFERENCIAS BIBLIOGRÁFICAS

Amazon. (2019). *¿Qué es Python? - Explicación del lenguaje Python - AWS*. Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/python/

Amazon. (2021). *¿Qué es un bot? - Explicación sobre los tipos de bots - AWS*. Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/bot/

ARIELMCORG, por. (2023, abril 10). *Poe.com – Tu plataforma para ChatGPT y más*. https://infosertecla.com/2023/04/10/poe-com-tu-plataforma-para-chatgpt-y-mas/

Bahit, E. (2012). *Scrum y eXtreme Programming para programadores*.

Bustamante, D. C. (2014). *Metodología XP*. UNIVERSIDAD NACIONAL EXPERIMENTAL DE LOS LLANOS OCCIDENTALES EZEQUIEL ZAMORA.

Cadavid, E., Martínez, J. D. F., & Vélez, J. M. J. P. (2013). *Revisión de metodologías ágiles para el desarrollo de software*. *11*(2), 30-39.

Calatrava, S. G. (2021, octubre 13). Qué es Gherkin: Cómo usarlo y cuáles son sus elementos. *Profile Software Services*. https://profile.es/blog/que-es-gherkin/

Carrillo, F. (2011). *Pruebas del Software: Niveles de Prueba del Software*.

Chat. (2021, noviembre). *Significado de Chat*. Significados.com. https://www.significados.com/chat/

Corrales, V. (2020). *Gherkin para escribir historias de usuario | Thiga España*. https://www.media.thiga.co/es/gherkin

Ecured. (2015). *Visual Paradigm—EcuRed*. https://www.ecured.cu/Visual_Paradigm

Eguiluz, J. (2018). *Capítulo 1. Introducción (Introducción a JavaScript)*. https://uniwebsidad.com/libros/javascript/capitulo-1

Eguiluz Pérez, J. (2008). *Introduccion a CSS*.

Fernández Romero, Y., & Díaz González, Y. (2012). *Patrón Modelo-Vista-Controlador*. *11*(1). http://revistatelematica.cujae.edu.cu/index.php/tele

Fernández, Y. (2023, julio 14). *Google Bard: Qué es, cómo funciona y qué puedes hacer con la inteligencia artificial que competirá con ChatGPT*.

IBM. (2021). *¿Qué es un chatbot? | IBM*. https://www.ibm.com/es-es/topics/chatbots

IBM. (2022). *¿Qué es Machine Learning? - México | IBM*. https://www.ibm.com/mx-es/analytics/machine-learning

JIMÉNEZ ALVARADO, Y. P. (2018). *HTML - Ensayos universitarios—7335 Palabras*. Buenas Tareas. https://www.buenastareas.com/ensayos/Html/81314431.html

Ken Schwaber, J. S. (2018). *The Scrum Guide*.

Larman, C. (2003). *UML y Patrones*.

Loos, E. M. (2016). Sanmartín Sáez, Julia. El chat: La conversación tecnológica. Madrid: Arco Libros, 2007. 95 pp. (isbn: 978-84-7635-710-1). *Rilce. Revista de Filología Hispánica*, *26*(2), 492. https://doi.org/10.15581/008.26.4739

López, B. R. (2022). ¿Qué es Google Colab? *Cursos GIS | TYC GIS Formación*. https://www.cursosgis.com/que-es-google-colab/

Martínez Peña, S. (2017, octubre 12). *Chatbot: ¿Qué es, para qué sirve y cómo funcionan? - Bloo Media*. https://bloo.media/blog/por-que-implementar-chatbot-en-tu-estrategia-de-marketing/

Merino, J. P. P. y M. (2011, septiembre 5). *Chat—Definicion.de*. Chat - Qué es, definición, características y clasificación. Definicion.de. https://definicion.de/chat/

Molina, sILVIA gabriela R. (2013). Métodologías ágiles enfocadas al modelado de requerimientos. *Informes Científicos Técnicos-UNPA*, *5*(1), 1-29.

Molina Montero, B., & Vite Cervallo, H. (2018). *Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software*.

Moreno, A. (2017, octubre 17). Procesamiento del lenguaje natural ¿qué es? - IIC. *Instituto de Ingeniería del Conocimiento*. https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/

Ortiz, P. (2023, marzo 20). *Chat GPT: Qué es, para qué sirve y su aplicación en la economía [explicado por Chat GPT]*. EDEM Escuela de Empresarios. https://edem.eu/chat-gpt-que-es-para-que-sirve-y-su-aplicacion-en-la-economia-explicado-por-chat-gpt/

Otero, E. (2023, febrero 21). *¿Qué es ChatGPT y para qué sirve? - Definición*. GEEKNETIC. https://www.geeknetic.es/ChatGPT/que-es-y-para-que-sirve

Pérez, L. (2021). *ChatGPT y ChatSonic—Descubre las diferencias*. https://neuroflash.com/es/blog/comparacion-entre-chatgpt-y-chatsonic-descubre-las-diferencias/#

Poe. (2023). *"Poe—Fast, Helpful AI chat"*. https://poe.com/s/5oEKm1dXhesJC2UderTh

Pressman. (2014). *Ingeniería del software: Un enfoque práctico, 7ma Edición – Roger S. Pressman*. https://www.freelibros.net/ingenieria/ingenieria-del-software-un-enfoque-practico-7ma-edicion-roger-s-pressman

Rodríguez, C., & Dorado, R. (2015). *¿Por qué implementar Scrum? 3*(1), 125-144.

Sommerville, I. (2016). *Ingeniería del Software Séptima Edición* (Vol. 7). PEARSON ADDISON WEASLEY, SA. Madrid, 2005.

Trigas Gallego, M. (2012). *Metodología Scrum*.

UIT. (2023, enero). *Inteligencia artificial para el bien*. https://www.itu.int/es/mediacentre/backgrounders/Pages/artificial-intelligence-for-good.aspx

Vincent, W. S. (2021). *Django for Beginners: Build websites with Python and Django: WelcomeToCode.*