UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE COMPUTACIÓN

PROTOTIPO DE APLICACIÓN MÓVIL PARA LA RECOMENDACIÓN AUTOMÁTICA DE UNA ORIENTACIÓN PSICOTERAPÉUTICA A TRAVÉS DEL USO DE MACHINE LEARNING

Trabajo de titulación previo a la obtención del
Título de Ingenieros en Ciencias de la Computación

AUTORES: JHON POLL IDROVO TITUAÑA
BORIS ARNALDO RUANO VÁSQUEZ
TUTOR: HOLGER RAÚL ORTEGA MARTÍNEZ

Quito – Ecuador
2024

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Jhon Poll Idrovo Tituaña con documento de identificación N° 1751451988 y Boris Arnaldo Ruano Vásquez con documento de identificación N°1755494075; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 21 de febrero del 2024

Atentamente,

---------- ----------

Jhon Poll Idrovo Tituaña Boris Arnaldo Ruano Vásquez

1751451988 1755494075

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Jhon Poll Idrovo Tituaña con documento de identificación No. 1751451988 y Boris Arnaldo Ruano Vásquez con documento de identificación No. 1755494075, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "Prototipo de aplicación móvil para la recomendación automática de una orientación psicoterapéutica a través del uso de Machine Learning", el cual ha sido desarrollado para optar por el título de: Ingenieros en Ciencias de la Computación, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 21 de febrero del 2024

Atentamente,

---------- ----------

Jhon Poll Idrovo Tituaña — Boris Arnaldo Ruano Vásquez

1751451988 — 1755494075

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Holger Raúl Ortega Martínez con documento de identificación N° 1708182728, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: PROTOTIPO DE APLICACIÓN MÓVIL PARA LA RECOMENDACIÓN AUTOMÁTICA DE UNA ORIENTACIÓN PSICOTERAPÉUTICA A TRAVÉS DEL USO DE MACHINE LEARNING, realizado por Jhon Poll Idrovo Tituaña con documento de identificación N° 1751451988 y por Boris Arnaldo Ruano Vásquez con documento de identificación N° 1755494075, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 21 de febrero del 2024

Atentamente,

Fís. Holger Raúl Ortega Martínez, MSc.

1708182728

AGRADECIMIENTOS

Agradecemos a Dios por darnos la fortaleza para mantenernos firmes durante nuestro proceso de estudio y otorgarnos una buena salud para culminarlos con éxito. También agradecemos a nuestros padres por brindarnos el apoyo incondicional, siendo el pilar fundamental en cada paso que hemos dado a largo de nuestras vidas y permitirnos seguir siempre adelante y mejorar día a día en nuestro ámbito personal y social.

A nuestro tutor Fís. Holger Ortega que ha sido parte de nuestro proceso de estudio desde los inicios de la carrera, queremos agradecerle por la enorme sabiduría que ha impartido a lo largo del tiempo y hoy por hoy en nuestro proceso de tesis. También un eterno agradecimiento a la Dra. María Fernanda Cazares por habernos guiado en el campo de la psicología a pesar de ya no formar parte de la institución, le deseamos éxitos en las metas que se proponga a futuro.

Jhon Poll Idrovo Tituaña

Boris Arnaldo Ruano Vásquez

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El propósito fundamental de este proyecto es desarrollar un prototipo de aplicación móvil destinado a recomendar la orientación psicoterapéutica más adecuada para las necesidades individuales de los usuarios utilizando técnicas de procesamiento natural del lenguaje (NLP) y aprendizaje automático con transformers y BERT. El desafío se encuentra en la falta de conocimiento y comprensión por parte de los usuarios respecto a las diversas orientaciones psicoterapéuticas, lo que puede llevar a elecciones inadecuadas para su padecimiento.

Al utilizar la metodología SCRUM involucró la división de los procesos en actividades puntuales para el desarrollo de los módulos que se implementan en el prototipo de aplicación móvil, destacando el chatbot como elemento principal, durante la interacción con el usuario al utilizar el servicio de OpenAI y posteriormente guardar la conversación en una base de datos no relacional. La arquitectura implementada se respalda en una estructura de n capas que incluye una capa de datos con una plataforma de gestión de base de datos no relacional, una capa de negocio con servidores Express y Flask para enviar y recibir los datos y la capa de interfaz de usuario o presentación desarrollada en React-Native.

Los resultados obtenidos mostraron que el prototipo, especialmente el módulo de chatbot, tuvo una positiva aceptación por parte de los usuarios. En conclusión, este trabajo contribuye a la solución de la problemática identificada, brindando a los usuarios una herramienta que facilita la elección adecuada de orientación psicoterapéutica, en base al análisis de una conversión entablada entre el usuario y el chatbot.

Palabras clave: Chatbot, procesamiento natural del lenguaje, módulo, recomendar, orientación psicoterapéutica, predicción, conversaciones.

ABSTRACT

The main purpose of this project is to develop a prototype mobile application aimed at recommending the most appropriate psychotherapeutic orientation for the individual needs of users using natural language processing (NLP) and Machine Learning techniques with Transformers and BERT. The challenge lies in the users' lack of knowledge and understanding of the various psychotherapeutic orientations, which can lead to inappropriate choices for their condition.

By using the SCRUM methodology involved the division of processes into specific activities for the development of the modules that are implemented in the mobile application prototype, highlighting the chatbot as the main element, during the interaction with the user by using the OpenAI service and subsequently saving the conversation in a non-relational database. The implemented architecture is supported by an n-layer structure that includes a data layer with a non-relational database management platform, a business layer with Express and Flask servers to send and receive the data, and the user interface or presentation layer developed in React-Native.

The results obtained showed that the prototype, especially the chatbot module, had a positive acceptance by the users. In conclusion, this work contributes to the solution of the identified problem, providing users with a tool that facilitates the appropriate choice of psychotherapeutic guidance, based on the analysis of a conversion between the user and the chatbot.

Keywords: Chatbot, natural language processing, module, recommend, psychotherapeutic guidance, prediction, conversations.

INTRODUCCIÓN

La incorporación de la tecnología en el sector de la salud se revela como un componente esencial para brindar soluciones personalizadas y de fácil acceso. En este contexto, surge la necesidad de explorar y comprender los distintos tipos y clases de orientación psicoterapéutica. El objetivo central de este proyecto es el desarrollo de un prototipo de aplicación móvil con el objetivo de recomendar la orientación psicoterapéutica más adecuada a las necesidades del usuario.

Para alcanzar este objetivo, el proyecto se estructura en cuatro capítulos. En el primero, se definen las herramientas y recursos a utilizar, como tecnologías de desarrollo, servidores, editor de código, entre otros. En el segundo capítulo, se emplea la metodología SCRUM para planificar las actividades a seguir a lo largo del proyecto, garantizando el logro de los objetivos y permitiendo ajustes en caso necesario.

El tercer capítulo se enfoca en definir la arquitectura y llevar a cabo el desarrollo de la aplicación, divididos en apartados de Back-End, Front-End y Machine Learning. Se detalla la importancia de cada uno de estos aspectos en el proyecto, incluyendo el código utilizado para garantizar el correcto funcionamiento de cada sección.

El último capítulo presenta el resultado final del proyecto, describiendo la funcionalidad de cada interfaz visual. Además, se llevan a cabo pruebas de funcionalidad y aceptación, permitiendo al usuario verificar si el software cumple con los criterios establecidos y si todas sus funciones operan de manera correcta.

En conclusión, esta aplicación móvil ofrece a los usuarios una perspectiva integral de los distintos tipos y clase de orientación psicoterapéutica, permitiéndoles así descubrir que podría estar causando sus problemas emocionales.

Antecedentes

Según el trabajo realizado por (Montaño & Fernando, 2019), desarrollaron una herramienta para el campo de la psicoterapia. Su objetivo es proporcionar a los usuarios información acerca de las prestaciones proporcionadas por el centro, así como permitirles acceder a recursos y herramientas útiles para su tratamiento. La aplicación cuenta con varias secciones como: inicio, servicios, noticias y de contacto.

En base al trabajo realizado por (Lamas et al., 2018), se busca explorar en cómo los recursos digitales pueden complementar la asistencia profesional y optimizar los tratamientos en salud mental. Los resultados muestran que en los trabajadores hay poco conocimiento sobre estas posibilidades y su uso es limitado. Además, se identificaron bloqueos y trámites para la implementación de tecnologías digitales en psicoterapia. La discusión analiza los resultados del estudio y ofrece recomendaciones para la implementación de tecnologías digitales en psicoterapia. En conclusión, el estudio destaca la importancia de seguir explorando el uso de tecnologías digitales en psicoterapia para mejorar la asistencia profesional en la salud mental.

Problema

En la investigación realizada por Ezequiel Benito se identifican cinco enfoques de tratamientos psicoterapéuticos aplicados para el abordaje de la complejidad de la mente y la conducta humana (Ezequiel, 2009). El enfoque conductual está centrado en la comprensión del impacto de los estímulos ambientales en el que se registra el comportamiento del paciente en su vida cotidiana, para modificar conductas problemáticas mediante técnicas de reforzamiento y condicionamiento (Ezequiel, 2009). El Modelo Cognitivo busca identificar y transformar patrones disfuncionales del pensamiento y cambiar la percepción del mundo (Ezequiel, 2009). El modelo Existencialista-Humanista se centra en el desarrollo personal para el autoconocimiento y crecimiento personal, aspectos que serán abordados en esta investigación. Finalmente, el tratamiento sistémico, aborda las relaciones y dinámicas familiares o sociales

que determinan el comportamiento del grupo. Cada enfoque terapéutico aborda aspectos específicos de la experiencia humana y ofrece técnicas particulares para promover el bienestar emocional y ayudar a las personas a alcanzar una vida más plena y satisfactoria. La elección del enfoque dependerá de las necesidades y preferencias individuales del usuario, así como el tipo e importancia de sus dificultades emocionales (Ezequiel, 2009). La escasa compresión entre la población acerca de las diferentes orientaciones psicoterapéuticas puede causar en las personas que buscan un tratamiento psicoterapéutico seleccionen un enfoque no adecuado para su padecimiento. Por lo tanto, se establecen criterios que permitan al usuario recibir recomendaciones claras y pertinentes respecto a su padecimiento.

En la investigación realizada por Ezequiel Benito menciona la no directividad de la relación de ayuda, el Terapeuta se abstiene de guiar al cliente en una dirección específica y evita influenciar sus pensamientos y acciones según un patrón preestablecido. En lugar de eso, se enfoca en establecer un contexto favorable para desarrollo personal, sin mostrarle el camino a seguir (Ramírez Benítez, 2007).

La Orientación Psicológica es un campo de trabajo especializado que tiene como objetivo fortalecer habilidades y capacidades, en un entorno profesional que ayuda a potencializar los recursos personales (Muela Aparicio & Méndez Eneko, 2020). De esta manera, las decisiones del cliente abordan las prácticas o cualidades del terapeuta o del tratamiento que el cliente estima, desea o anticipa.

El escaso conocimiento sobre las opciones terapéuticas disponibles que ofrecen los psicólogos puede tener consecuencias negativas para aquellos que buscan mejorar su bienestar emocional y mental. Debido a la falta de comprensión de las distintas orientaciones psicológicas, las personas pueden optar por terapias que no se ajusten a sus necesidades individuales o que no aborden adecuadamente sus problemas específicos. Esto puede prolongar su padecimiento Psicológico y hacer que se sientan desanimados o desesperanzados respecto a

la posibilidad de obtener ayuda efectiva (Muela Aparicio & Méndez Eneko, 2020). Así, la comprensión y el reconocimiento de aspectos intrínsecos del paciente son fundamentales para regular la relación de manera efectiva, permitiendo una exploración más profunda de su vivencia.

Hasta el momento, los chatbots no han sido utilizados de manera efectiva para la predicción de la Orientación del tratamiento que los pacientes necesitan. Como parte integral de este proyecto de tesis, se pretende llenar este vació mediante la construcción de una aplicación móvil competente para identificar el tipo de trastorno y recomendar la orientación psicoterapéutica más adecuada. Este innovador enfoque busca aprovechar la tecnología para mejorar la accesibilidad y la precisión en el análisis y manejo de dificultades en la salud mental.

Los trastornos con mayor demanda son los siguientes: miedo/ansiedad, autoestima, ajustamiento, logro, rasgos de personalidad, comportamiento social, trastorno emocional/somático. La problemática se presenta cuando se obtiene respuestas que no se ajuste a la orientación psicológica más adecuada a las necesidades del paciente (Smith & Glass, 1977). Los chatbots puede devolver respuestas inapropiadas al interpretar declaraciones de forma literal sin tomar en consideración el contexto o las distintas formas de como las personas se expresan, ocasionado cambios en el sentido real de las frases (Fernández Trejo et al., 2019). La problemática se encuentra en obtener información general que permita analizar los diferentes trastornos psicológicos, Esta información será útil y servirá como base durante la fase de evaluación si el usuario decide comenzar una terapia (Romero et al., 2020).

En resumen, la insuficiencia de conocimiento y comprensión respecto a las diversas orientaciones psicoterapéuticas puede provocar que los pacientes no elijan una orientación adecuada para su padecimiento.

Para aportar a la solución de este problema, se pretende realizar una aplicación móvil que recomiende de forma automática la orientación psicoterapéutica que más se apegue a la necesidad del usuario.

En el caso de seguir con el padecimiento, el usuario deberá realizar una reevaluación de la recomendación obtenida inicialmente, en la que incluya la sintomatología que presenta. Si la nueva sugerencia no cumple con las expectativas, se sugiere que consulte con un profesional especializado en la salud mental.

Justificación

La integración de herramientas tecnológicas en el ámbito de la salud mental ha mostrado ser una perspectiva con promisorias implicaciones en la mejora de los servicios terapéuticos (Lamas et al., 2018). Por tanto, es de suma importancia que los profesionales de la salud se mantengan al corriente y reciban formación en el uso de tecnologías digitales.

La viabilidad del proyecto se fundamenta en la creciente utilidad y aplicabilidad de los chatbots en diversas áreas, incluyendo la evaluación e intervención psicológica. Además, la implementación de chatbots en el campo de la psicología abre la puerta a una mayor disponibilidad de servicios de apoyo psicológico (Romero et al., 2020). Al desarrollar un chatbot que pueda recopilar información relacionada a los tipos de trastornos psicológicos como: trastorno de miedo o ansiedad, trastorno de autoestima, trastorno ajustamiento, trastorno de logro, trastorno de rasgos de personalidad, trastorno de comportamiento social, trastorno emocional o somático. Es necesario proporcionar información de las orientaciones psicológicas más adecuadas para el paciente como: orientación conductual, orientación cognitiva, orientación dinámica o humanista, orientación mínima, ya que se establece una base sólida de la orientación psicológica óptima para el tratamiento del problema del paciente (Smith & Glass, 1977).

El propósito esencial de este proyecto es la creación de un prototipo para una aplicación móvil con la capacidad de reconocer el tipo de problema psicológico y el tratamiento asociado, en base a ello pueda recomendar la orientación psicoterapéutica, a través de la asistencia de un chatbot que interactúe con el usuario. Mediante un análisis de la literatura científica, se concluyó que no hay otro proyecto realizado en nuestra localidad de la misma naturaleza; además, empleando procesamiento del lenguaje natural, se posibilita el análisis de grandes volúmenes de texto con el fin de identificar patrones y tendencias relacionadas al tipo de problema psicológico que presenta el paciente como ansiedad o depresión, fobias, problemas físicos o de hábito, problemas sociales o sexuales y ansiedad durante el desempeño, en base ello se recomendará el tipo de terapia más adecuada para el paciente. Esto puede ayudar a tener un mejor entendimiento sobre el tipo de malestar que presenta un paciente durante la interacción con el chatbot.

Objetivos

Objetivo general

- Desarrollar un prototipo de aplicación móvil que recomiende la orientación psicoterapéutica más adecuada a la afección del paciente.

Objetivo específico

- Realizar una revisión de la efectividad de las orientaciones psicoterapéuticas en base a los diferentes tipos de trastornos.
- Realizar un análisis de aplicaciones y documentos relacionados al desarrollo de chatbots para detección de trastornos psicológicos.
- Construir diferentes módulos para los usuarios que permitan interactuar con la aplicación: módulo de bienvenida, registro de usuario, inicio de sesión, módulo que

muestre el resultado de la orientación psicoterapéutica recomendada para el usuario y chatbot.

Metodología

Para ejecutar el proyecto del prototipo de aplicación móvil, se adoptará la metodología SCRUM. Esta metodología, al dividir las actividades en sprints, optimizará la planificación para un desarrollo ágil e iterativo. La divulgación de resultados en fases sucesivas se gestionará mediante reuniones regulares, permitiendo revisar el progreso del proyecto, identificar obstáculos tempranamente y aplicar ajustes e iteraciones para mejorar continuamente la aplicación. La metodología scrum resulta eficaz para organizar y priorizar funciones en el desarrollo de un chatbot. Por ejemplo, permite diseñar la conversación, definir flujos de interacción y diálogos para asegurar una experiencia de usuario consistente. Además, el reconocimiento de lenguaje natural es clave para una comunicación efectiva que puede desarrollarse mediante scrum. Así, el prototipo de aplicación móvil pueda aprender a interpretar y responder adecuadamente a las consultas de los usuarios, adaptándose a su forma de expresarse.

En el desarrollo de un sprint, tienen lugar los siguientes eventos:

- Revisión de literatura: analizar artículos científicos, estudios previos, algoritmos y técnicos relacionados hacia chatbots, y procesamiento natural del lenguaje (NLP).
- Diseño y arquitectura: desarrollaremos una aplicación móvil con técnicas de procesamiento natural de lenguaje. Esto abarcará la estructura de la interfaz, la lógica del chatbot y la integración de técnicas mencionadas, posibilitando al chatbot comprender y responder consultas en lenguaje natural.
- Selección de herramientas y recursos: se identifica las herramientas y recursos necesarios que ayuden a implementar algoritmos de NLP en el chatbot con la

investigación de bibliotecas de NLP, modelos de lenguaje previamente entrenados y API's de procesamiento de texto.

- Desarrollo de prototipo: construir un prototipo de aplicación móvil compuesto por procesamiento del lenguaje natural (NLP) para una interacción inteligente y contextual. La elección de tecnologías se adaptará a la plataforma de Android, y se seguirá la metodología scrum para la planificación iterativa y entregas incrementales, permitiendo ajustes según la retroalimentación del prototipo.
- Entrenamiento del modelo de NLP: mejorar la capacidad del chatbot para comprender y responder a consulta teniendo en cuenta que se deberá ajustar y entrenar con fin de obtener un mejor resultado.
- Pruebas y evaluación: se realizarán pruebas funcionalidad, pruebas de aceptación y pruebas estructurales.

Alcances

Esta sección está organizada de acuerdo con los objetivos específicos previamente definidos donde se abordan en profundidad las características del prototipo de la aplicación móvil.

- Realizar una revisión de la efectividad de las orientaciones psicoterapéuticas en base a los diferentes tipos de trastornos.
    - En este apartado se identifican estudios científicos, investigaciones previas que muestren la efectividad de las diferentes orientaciones psicoterapéuticas en el manejo de problemas emocionales.
- Realizar un análisis de aplicaciones y documentos relacionados al desarrollo de chatbots para detección de trastornos psicológicos.

- Aquí se examinará principales características y funcionalidades en aplicaciones de chatbot utilizadas en la terapia psicológica, en ello se incluyen aspectos de interacción con los usuarios y las técnicas de conversación empleadas.
- Recopilar artículos científicos sobre el desarrollo y la implementación de chatbots en terapia psicológica, con la finalidad de comprender los fundamentos teóricos, metodologías de diseño y evidencias de efectividad.

- Construir diferentes módulos para los usuarios que permitan interactuar con la aplicación: módulo de bienvenida, registro de usuario, inicio de sesión, módulo que muestre el resultado de la orientación psicoterapéutica recomendada para el usuario y chatbot. Las funcionalidades de dichos módulos son:
  - En el módulo de bienvenida: proporcionar a los usuarios un recibimiento cálido y amigable al ingresar al sistema, donde se muestra sus nombres completos y un mensaje de bienvenida.
  - En el módulo registro de usuario: los usuarios pueden proporcionar la información necesaria para registrarse en nuestra plataforma.
  - En el módulo inicio de sesión: los usuarios tienen que proporcionar sus credenciales (nombre de usuario y contraseña) para iniciar sesión e ingrese al módulo de bienvenida, mostrar el resultado de la orientación psicológica recomendada para el usuario y chatbot. mediante su usuario y contraseña.
  - En el módulo que muestre el resultado de la orientación psicoterapéutica recomendada para el usuario: muestra el resultado de la orientación psicoterapéutica más efectiva para el paciente, obtenida por el chatbot durante el diagnóstico en base a el tipo de trastorno detectado.
  - En el módulo de chatbot: en esta sección, se desarrollará un chatbot que interactúe con el usuario a través de una entrevista personalizada. Utilizando esta

entrevista, el chatbot recopilará la información sobre el problema o las dificultades que el paciente describe y recomendará la orientación psicoterapéutica más efectiva.

CAPÍTULO I: MARCO TEÓRICO

1.1 Tecnología de desarrollo

En la Tabla 1 se detalla el concepto de cada una de las cuatro tecnologías compuesta en el stack MERN según (Naranjo Heredia, 2021).

Tabla 1

Stack Mern.

Tecnología	Descripción
MongoDB	Para el autor ya mencionado explica que MongoDB se especializa en documentos y es una base de datos NoSQL que guarda los datos al formato BSON y documentos similares al formato JSON con esquemas dinámicos. Esto significa que puedes almacenar y recuperar datos de manera flexible sin tener una estructura fija de tabla.
Express	Como menciona el autor, está pensado en el levantamiento de las aplicaciones móviles en Node.js con el fin de minimizar el tiempo requerido para la creación y puesta en marcha. En sus aspectos destacados se encuentra la facilidad para la creación de servidores web, manejar rutas, middleware, solicitudes y respuestas HTTP.
React	El autor menciona que React es una biblioteca diseñada para la creación de interfaces de usuario interactivas, a través de la construcción de componentes reutilizables, los cuales se integran para conformar interfaces de usuario complejas.
Node.js	Como entorno de ejecución en el servidor, Node.js posibilita ejecutar código JavaScript fuera del navegador, destacándose por su capacidad para construir aplicaciones web que pueden crecer de manera eficiente y presentar un rendimiento estable.

Nota. Descripción breve de cada tecnología que compone el stack MERN. Fuente: (Naranjo Heredia, 2021).

Como menciona (Naranjo Heredia, 2021), se puede afirmar que el stack MERN demuestra una versatilidad destacada en distintos proyectos y aplicaciones. Actualmente, la arquitectura predominante es REST. En este enfoque, la lógica del servidor se configura como

un servicio REST, gestionando las solicitudes del usuario y proporcionando resultados en formato JSON al cliente. El Front-End, construido con ReactJS, consume este conjunto de datos. Esta arquitectura posibilita la creación de aplicaciones web más simples y ágiles, conllevando a un resultado similar a una aplicación de escritorio. En resumen, el stack MERN sigue una arquitectura para crear APIs Restful para aplicaciones, lo que favorece el modularidad en los proyectos y facilita el mantenimiento a lo largo del tiempo.

1.2 Metodologías ágiles

Según la información recogida en la Tabla 2 se detalla la comparativa de las diferentes metodologías frente a la metodología SCRUM con sus ventajas y desventajas.

Tabla 2

Tabla comparativa de las diferentes metodologías.

Metodología	Descripción	Ventajas	Desventajas
Scrum	Promover la colaboración en proyectos. Se destaca la realización de sprints para evaluar observaciones del cliente y la importancia de reuniones diarias breves para coordinar eficientemente el progreso del proyecto y asignar roles claros al equipo de desarrollo.	Adaptarse a proyectos con cambios frecuentes en los requisitos y se ha adoptado en prácticas orientadas a CMMI o PSP/TSP.	El uso continuo de sprints puede generar estrés en el equipo de trabajo.
XP	La metodología se basa en cinco valores clave para obtener retroalimentación rápida y ser adaptable: simplicidad, comunicación, respeto y coraje.	Fomenta la productividad, es apta para proyectos con requisitos cambiantes y promueve relaciones	Los desarrolladores aprueban los códigos, pero presenta desafíos en grandes organizaciones y no es apropiada

		interpersonales y aprendizaje continuo.	para proyectos extensos o equipos numerosos.
Proceso unificado ágil (AUP)	El AUP utiliza técnicas ágiles, como TDD y modelado ágil, para mejorar la productividad en un ciclo de vida de cuatro fases: concepción, elaboración, construcción y despliegue.	Fácil de aprender, modificar y adaptable.	Mayor disponibilidad para trabajar en equipo entre desarrolladores.

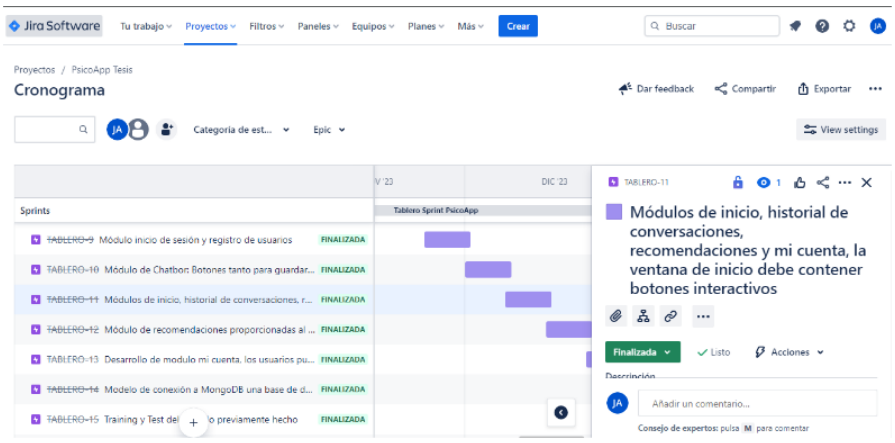
Nota. Se realiza una comparación con otras metodologías de desarrollo de software frente a la metodología SCRUM. Fuente: (Velásquez et al., 2019).

En base a las descripciones mencionadas por (Velásquez et al., 2019) se eligió SCRUM, siendo la estructura en iteraciones ideal a una duración específica, en este caso, cada una semana para tarea incumplida y dos semanas para una tarea cumplida. Al concluir cada sprint, se revisa los logros alcanzados y se retroalimenta el método para mejorar en el siguiente ciclo.

Para alcanzar estos objetivos, se seleccionó una herramienta que facilitará la preparación de los sprints, la visualización de las tareas en distintas etapas del flujo de trabajo y el seguimiento del progreso. Por esta razón, se eligió utilizar la herramienta JIRA.

Figura 1

Cronograma de actividades utilizando JIRA.



Nota. Desarrollo de las actividades en curso dentro del proyecto técnico. Elaborado por los autores.

En el ámbito del enfoque scrum, un sprint se divide en varias fases, cada una con su propio conjunto de actividades. A continuación, se describen las fases clave de un sprint en scrum:

1. **Planificación del sprint:** Se reúne al equipo para seleccionar y priorizar las tareas más importantes que se abordarán durante el sprint. Durante esta sesión, el objetivo del sprint se define y como se medirá el éxito. En este proceso, el scrum master, asumiendo el papel de tutor, verifica el tiempo estimado para cada tarea y su asignación, asegurando una planificación efectiva para alcanzar los objetivos del sprint.

2. **Desarrollo (Development):** En esta etapa, el equipo se enfoca en la implementación de los requerimientos de los usuarios. Los desarrolladores colaboran estrechamente para ejecutar las tareas asignadas, trabajando de manera conjunta para alcanzar con éxito los objetivos del sprint.

3. **Reuniones diarias:** El equipo debe completar las tareas asignadas. Realizan reuniones semanales cortas para compartir actualizaciones y coordinar esfuerzos. Adicionalmente, cada sábado se reúnen para revisar el progreso de éxito y abordar posibles inconvenientes con el cumplimiento del sprint.

4. **Revisión del sprint:** Al finalizar el sprint, se reúne al equipo para realizar una revisión exhaustiva del trabajo completado. Durante esta sesión, demuestran las funcionalidades desarrolladas y recopilan valiosos comentarios del product owner y otros stakeholders. Esta reunión no solo evalúa el cumplimiento del objetivo del sprint, sino que también permite discutir mejoras potenciales para futuros sprints, facilitando así un proceso de desarrollo continuo y adaptativo.

5. **Retrospectiva del sprint:** Al finalizar cada sprint, el equipo debe evaluar el desempeño, aplicar mejoras y planificar ajustes para futuros ciclos. Este enfoque se centra en la mejora continua. Al finalizar el trabajo, se entrega al tutor encargado, tras varias pruebas realizadas a detalle con la intención de cumplir los objetivos previamente establecidos.

1.3 Arquitectura en capas

Como menciona (Zottola, 2023), la arquitectura de n capas se encuentra representada como una estructura de tres niveles: la base, el cuerpo y la cima. Esta metáfora captura la esencia de cómo se descompone una aplicación en componentes esenciales.

1. **Capa de presentación:** Está conformada por el entorno visual del aplicativo que es utilizado desde aplicaciones web hasta dispositivos móviles, esta capa da vida a la experiencia al presentar datos y capturar acciones.

2. **Capa de lógica de negocio:** En el nivel intermedio, la lógica de negocio adquiere independencia. Aquí residen las reglas empresariales, el flujo de procesos y los cálculos. Este estrato es el núcleo vital de la aplicación, donde la funcionalidad y la inteligencia convergen para dar vida a la aplicación.

3. **Capa de acceso a datos:** En la última capa, se establece conexión con la base de datos, vinculándola al almacén de datos o fuente de datos externos. Aquí, se obtienen, modifican y almacenan datos, completando así el ciclo esencial de la gestión de datos en la aplicación.

1.4 Herramientas de desarrollo

Visual Studio Code: Como se menciona (Microsoft, 2023), se trata de un entorno de edición de código fuente, caracterizado por ser software libre y compatible con diversas plataformas como Windows, GNU/Linux y macOS.

ChatGPT: Según (Rahimi & Bezmin Abadi, 2023), es una plataforma de aprendizaje profundo y se somete a un aprendizaje continuo y reentrenamiento mediante la referencia al

extenso cuerpo de texto disponible en internet. Es capaz de generar respuestas escritas de alta calidad, plausibles y parecidas a las humanas. Además, puede generar análisis estadísticos, letras de canciones, programas informáticos y resúmenes o introducciones de artículos científicos.

Android Studio: Como menciona (Gupta et al., 2018), es un software dedicado al desarrollo de aplicaciones Android. En Android Studio se puede escribir código, depurar, probar y empaquetar aplicaciones para dispositivos Android.

Android Debug Bridge (ADB): Según (Developers, 2023), es una herramienta de comandos extremadamente versátil que permite la comunicación con dispositivos Android. Este comando ofrece funciones, permitiéndote realizar acciones como instalar y depurar aplicaciones de manera sencilla y eficiente en tus dispositivos.

NoxPlayer: Como menciona (Linares, 2021), es un emulador de Android compatible tanto con Mac como con PC, proporcionando todas las funciones esenciales para operar en la computadora de manera similar a un dispositivo móvil con Android.

GitHub: Como menciona (Lima et al., 2014)**,** es plataforma de alojamiento para proyectos de software. En GitHub, los usuarios pueden crear repositorios de código y subir sus contribuciones. Cada repositorio cuenta con una lista de colaboradores, quienes tienen la capacidad de realizar cambios en el contenido del repositorio.

Flask: Según (Grinberg, 2014), se trata de un conjunto de herramientas para desarrollar aplicaciones web de manera directa pero eficiente, sin imponer una estructura o conjunto en específico.

Python: Para el autor (González Duque, 2011), se define como un lenguaje de programación distintivo debido a sus tipos de datos dinámicos, su compatibilidad con múltiples plataformas y su enfoque orientado a objetos.

React-Native: Según (Danielsson, 2016), es una herramienta de desarrollo que facilita la generación de aplicaciones para dispositivos móviles. Al emplear JavaScript y React, permite escribir código y desplegar aplicaciones. La característica clave es su capacidad para utilizar componentes nativos y un "puente" que conecta el código JavaScript con las API nativas, garantizando una experiencia nativa en ambas plataformas.

1.5 Trabajos relacionados con la psicoterapia

En el trabajo mencionado de (Sedlakova & Trachsel, 2023), examina el uso de la inteligencia artificial conversacional (CAI) en psicoterapia. Se exploran los beneficios potenciales y los riesgos éticos que pueden surgir. Los autores sugieren que la CAI puede ser beneficiosa para brindar apoyo terapéutico a aquellos sin acceso a la atención convencional, pero enfatizan la necesidad de establecer límites claros en sus funciones. Además, abordan cuestiones éticas sobre la relación entre humanos y la inteligencia artificial, la confidencialidad de los datos y la integridad de los usuarios.

Dentro de la revisión realizada por (Boucher et al., 2021), exploran la demanda constante referente a los servicios de salud mental y cómo los chatbots de IA pueden contribuir a abordar esta necesidad. Se discuten los beneficios potenciales, como la accesibilidad y la personalización de la atención, así como los desafíos, como la privacidad y la confidencialidad de los datos. Además, se presentan ejemplos de estudios y proyectos que utilizan chatbots para brindar apoyo en salud mental.

1.6 Métodos de prueba

Caja Negra: El autor explica que un modelo o sistema que funciona de manera opaca, es decir, cuyo proceso de toma de decisiones o generación de resultados no es fácil de comprender, plantea un desafío durante la toma de decisiones. (González Loyola, 2019).

Caja Blanca: El autor explica que un modelo transparente al trabajar en la toma decisiones resulta ser una alternativa preferida, especialmente en aplicaciones críticas, donde la

necesidad de comprender y justificar las conclusiones obtenidas son fundamentales para garantizar la fiabilidad y la confianza en el sistema. (González Loyola, 2019).

CAPÍTULO II: MARCO METODOLÓGICO

2.1 Metodología de desarrollo

La metodología a poner en práctica en este proyecto fue: "SCRUM", ya que permite trabajar en las tareas a través de iteraciones o sprints de acuerdo a la duración de las actividades a desarrollar. Después de completar las actividades asignadas en cada sprint, se lleva a cabo un análisis con el fin de alcanzar los logros requeridos, proporcionando retroalimentación sobre los procesos y la mejora de las siguientes iteraciones.

Roles scrum:

Product owner: Holger Raúl Ortega Martínez (HO).

Scrum team:

Developer 1: Jhon Poll Idrovo Tituaña (JI).

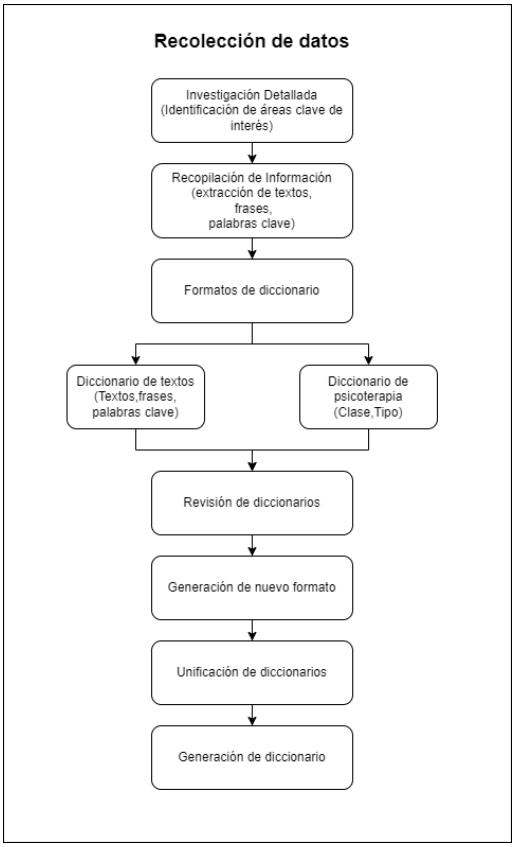
Developer 2: Boris Arnaldo Ruano Vasques (BR).

2.2 Propuesta de solución

I. Durante la primera fase se establece el enfoque en la exploración y recopilación inicial de temas relevantes de psicoterapia que aporten valor en el desarrollo del aplicativo móvil. Se lleva a cabo una investigación detallada para identificar áreas clave de interés y se establecen las bases iniciales del proyecto. Se recopila información esencial para alimentar nuestro diccionario de datos, esta tarea implica la identificación y extracción de textos significativos, frases clave y palabras clave específicas presentes en documentos relacionados con psicoterapia. Después de la recopilación de datos, se estableció un formato con tres parámetros para segmentar la información de manera perceptible y organizada facilitando así su uso en procesos de entrenamiento de modelos y desarrollo continuo de la aplicación.

Figura 2

Proceso de recopilación de información.



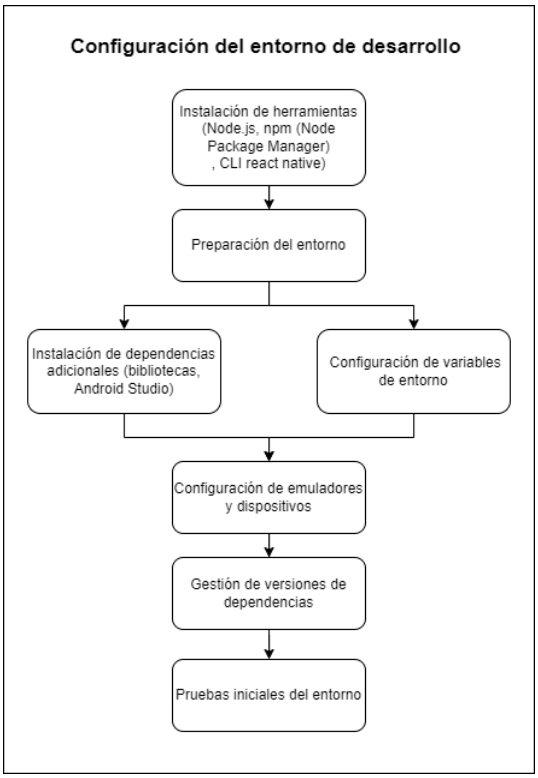
Nota. Se menciona los pasos a seguir en la recolección de datos. Elaborado por los autores.

Según la información recogida en la Figura 2**,** se detallan las fases planificadas y ejecutadas para recolectar la información requerida para el entrenamiento del modelo.

II. Durante la segunda fase se inicia la configuración del entorno de desarrollo en React Native. Esto incluye la instalación de herramientas y la preparación del entorno, abarcando la configuración de variables de entorno, ajustes específicos del proyecto y la instalación de dependencias adicionales. Con respecto al desarrollo para la interfaz de usuario abarcó la creación de maquetas y prototipos que visualizan la apariencia y la estructura de la aplicación, considerando aspectos como la usabilidad, accesibilidad y estética.

Figura 3

Ajuste del ambiente de desarrollo.



Nota. Se menciona la configuración del entorno de desarrollo. Elaborado por: Los autores.

Según la información recogida en la Figura 3, se muestran las fases de configuración del entorno de desarrollo y ofrece una visión clara y estructurada de los pasos clave involucrados.

Figura 4

Maquetado de aplicación móvil.

Nota. El diseño inicial del maquetado para la aplicación móvil. Elaborado por: Los autores.

Según la información recogida en la Figura 4 se muestra el maquetado de la aplicación móvil con una visión general y estructura inicial de la aplicación como punto de referencia para evaluar y ajustar elementos clave del diseño durante las fases de desarrollo.

III. La tercera fase corresponde al desarrollo Front-End del módulo de bienvenida, que incluye botones interactivos para facilitar la navegación y ofrece la opción de registrarse e iniciar sesión. Con la implementación del formulario de registro de usuarios, permitirá capturar la información ingresada durante el proceso de registro y por defecto se guardará en la base de datos. Se incorpora validaciones en el Front-End con mensajes claros y comprensibles para informar al usuario sobre la acción que realiza, lo que permite verificar y realizar la corrección de datos oportunamente.

IV. La cuarta fase corresponde al desarrollo del módulo de inicio de sesión, para el módulo mencionado, se crea un formulario con campos destinado al ingreso de datos de usuario y contraseña., botones interactivos para el inicio de sesión y recuperar contraseña, en el formulario se incorporan validaciones de datos en el Front-End y mensajes de alerta para errores de usuario no registrado o contraseña incorrecta, el botón de recuperación de contraseña se encarga de abrir una ventana emergente con campos para validar los datos del usuario que olvido su contraseña, En el Front-End de la ventana modal, se incluye la funcionalidad para ingresar la nueva contraseña, así como botones interactivos para cambiar la contraseña o cancelar el proceso. Además, se implementan validaciones para verificar la existencia del usuario y garantizar el ingreso correcto de los datos.

V. La quinta fase corresponde al desarrollo de la interfaz del módulo del chatbot mediante la biblioteca “GriftedChat”. Se utiliza las capacidades y componentes proporcionados por “GriftedChat” para definir la apariencia y la disposición visual del chatbot. La interfaz del chatbot incluye botones interactivos para guardar y realizar la recomendación, el botón guardar

permitirá al usuario asignar un nombre descriptivo a la conversación que se desea guardar a través de una ventana modal y posteriormente almacenar el contenido del chat en el repositorio de datos. El botón recomendar ofrece la capacidad de despliegue de una ventana modal que ofrece una recomendación específica de la orientación psicoterapéutica adecuada para el paciente, basada en la interacción y la conversación que se ha guardado en el repositorio de datos.

VI. La sexta fase corresponde al desarrollo de los módulos de inicio, historial de conversaciones, recomendaciones y mi cuenta, la ventana de inicio contiene los botones interactivos para acceder a las diferentes ventanas que se detallan a continuación: chatbot, conversaciones, recomendaciones, mi cuenta y cierre de sesión. El desarrollo del módulo de conversaciones ofrece a los usuarios un espacio en el que pueden revisar y gestionar sus interacciones pasadas con el chatbot, la ventana contiene botones interactivos con los títulos que le corresponden a cada chat almacenado por el usuario, los botones tienen la capacidad de revisar el registro de conversiones almacenadas y mostrarlas en una nueva ventana en la que puede interactuar nuevamente en la conversación guardada.

El desarrollo del módulo de recomendaciones permite acceder al usuario a las recomendaciones personalizadas basadas en su historial de conversaciones, la ventana contiene botones interactivos con los títulos de la conversación almacenadas en la base de datos en ocasiones pasadas y los botones pueden acceder al resultado de la recomendación y mostrarlo a través de una ventana modal.

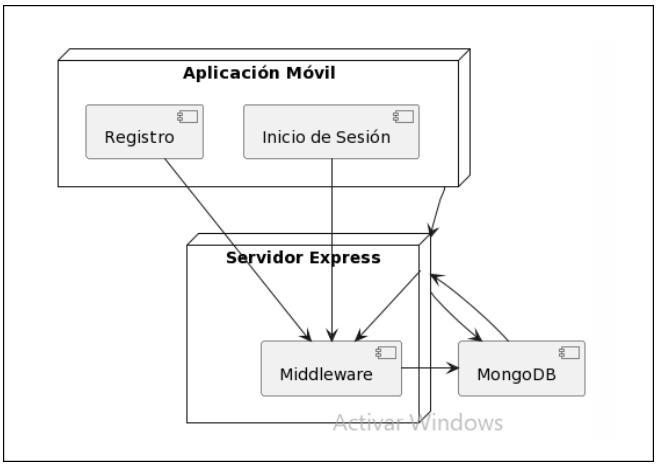
En el desarrollo de modulo mi cuenta, los usuarios pueden visualizar y gestionar la información de su perfil, se proporciona la capacidad para que los usuarios actualicen su información de perfil para ello se incorpora botones interactivos para facilitar la actualización de información como la contraseña y el nombre de usuario, lo que ofrece a los usuarios una

forma fácil de realizar cambios en su cuenta. Se incluye el botón interactivo de cierre de sesión permitiendo a los usuarios salir de su cuenta de manera segura.

VII. La séptima fase corresponde a la implementación del modelo para la vinculación con MongoDB y la creación del servidor en Express. Este componente tiene el papel central al actuar como el punto de conexión entre el Back-End y Front-End de la aplicación móvil para los módulos detallados a continuación: registro, inicio de sesión, chatbot, conversaciones y mi cuenta. Se han definido modelos que especifican la forma y la relación entre los datos almacenados en MongoDB, facilitando así la representación organizada y estructurada de la información Se crea un middleware (Intermediario para solicitudes HTTP) para los módulos conformados por: registro de usuario e inicio de sesión. En el momento de iniciar sesión, este middleware realiza la validación de las contraseñas.

Figura 5

Conexión entre la aplicación, servidor y base de datos.



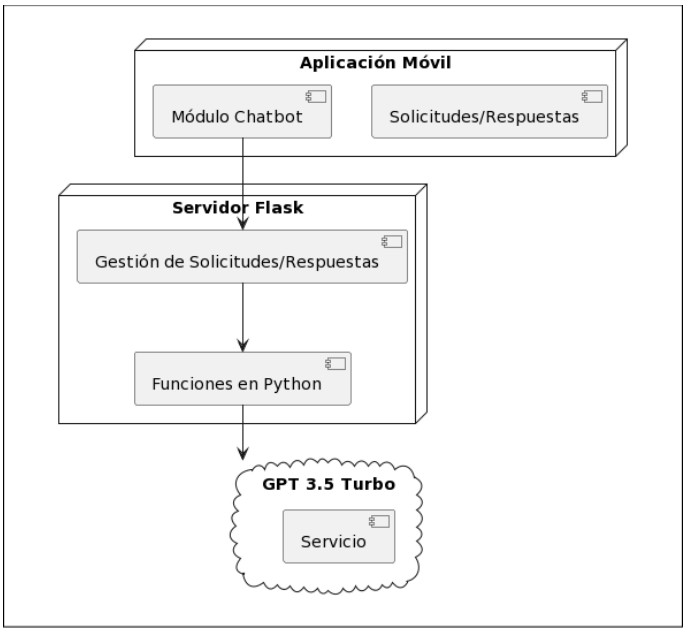
Nota. Se observa el diagrama de despliegue que muestra la interacción entre la capa de aplicación móvil, servidor en Express y base de datos. Elaborado por los autores.

VIII. En la octava fase se lleva a cabo la implementación del servidor Flask y la configuración de la conexión para consumir el servicio de OpenAI (GPT 3.5 Turbo) mediante Python. Se ha creado el servidor en Flask con la finalidad de gestionar las solicitudes y respuestas a través de la aplicación móvil, específicamente en el módulo de chatbot. Además,

se han desarrollado funciones en Python para consumir de manera eficiente el servicio de OpenAI (GPT 3.5 Turbo). Se ha trabajado en la configuración de roles específicos para OpenAI (GPT 3.5 Turbo) en su interacción con el usuario, permitiendo aprovechar su avanzada capacidad para procesar texto y mejorar las respuestas del chatbot frente al texto ingresado por el usuario.

Figura 6

Conexión entre la aplicación y servidor Flask para consumir servicio GPT 3.5.



Nota. Se implementa el servidor Flask y se configura la conexión para utilizar el servicio de GPT-3.5 Turbo mediante Python. Elaborado por los autores.

IX. La novena fase, se enfoca en el desarrollo del lenguaje natural utilizando la biblioteca transformers y el modelo BERT pre-entrenado en español ('dccuchile/bert-base-spanish-wwm-cased'). Se realiza la preparación de los datos antes de su procesamiento para ello se requiere el diccionario de datos en formato (CSV) que se describió en el primer sprint, para pre-procesar las etiquetas y la distribución de datos en dos grupos distintos: uno destinado al entrenamiento y otro par las prueba, se utiliza el tokenizador BERT para convertir textos en representaciones numéricas, posteriormente se crea modelos BERT específicos para las tareas

de clasificación de clase y tipo, estos modelos BERT están diseñados para abordar una tarea específica y aprender patrones relevantes en los datos.

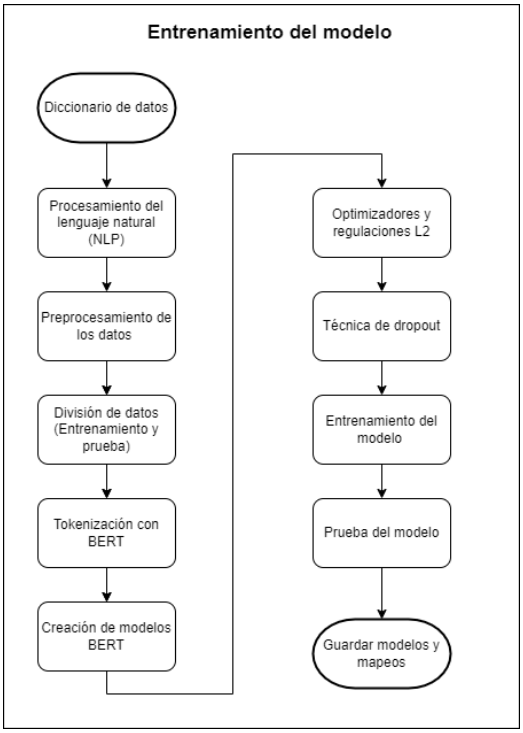
Durante la fase de entrenamiento, el modelo es alimentado con datos de entrenamiento y ajusta los pesos de sus parámetros para adaptarse a los patrones presentes en esos datos. Durante esta fase, el modelo aprende a llevar a cabo la tarea específica de clasificación de texto. Posteriormente, en la fase de prueba, se mide la adaptabilidad del modelo a datos no vistos previamente examinados, utilizando un conjunto de prueba. Esta evaluación proporciona una valoración del modelo en función de métricas de rendimiento como la precisión.

Con el fin de aumentar la eficacia del modelo, se emplea la técnica de dropout, que implica apagar de manera aleatoria un cierto porcentaje de las neuronas (nodos) en la red en cada iteración de entrenamiento. Esta estrategia desempeña una función en prevenir el sobreajuste al impedir que el modelo se apoye demasiado en ciertas características o combinaciones específicas y así establece una mejor adaptabilidad a diferentes datos.

Se establecen los optimizadores con regularización L2 utilizando el optimizador AdamW en los modelos. Esta configuración busca mejorar la eficiencia del proceso de optimización y prevenir el sobreajuste. Además, se guarda el mapeo de las clases etiquetadas en formato JSON. Una vez completados las épocas establecidas previamente para el proceso de entrenamiento, los modelos son guardados.

Figura 7

Proceso de entrenamiento del modelo.



Nota. Diagrama donde describe el proceso de entrenamiento del modelo. Elaborado por los autores.

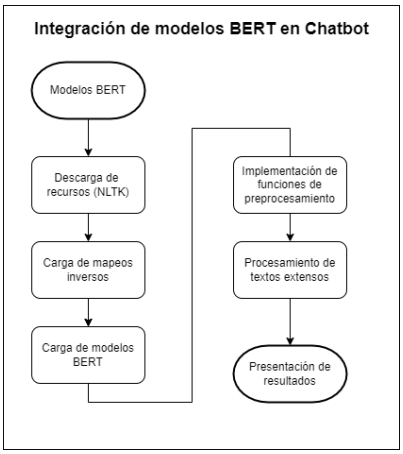
X. La Décima Fase, corresponde a la integración de los modelos BERT pre-entrenados, descritos durante la novena fase, los modelos se integran en el servidor Flask para que sean accesibles desde el módulo de chatbot, permitiendo así obtener resultados de predicción. Este proceso se compone de dos etapas. Primero, se descargan los recursos adicionales del NLTK en inglés (Natural Language Toolkit) para facilitar el procesamiento de texto, incluyendo tareas como tokenización, lematización, stemming, eliminación de stop words y conversión de textos a minúsculas.

En la segunda etapa, se cargan los mapeos inversos desde archivos JSON, que contienen las clases y tipos previamente definidos. Además, se incorporan los modelos BERT pre-entrenados específicos para la clasificación de clase y tipo. Se implementan funciones de pre-procesamiento del texto que se encargan de realizar las predicciones de clase y tipo. Se

introduce un método de procesamiento especial para textos extensos, utilizando una técnica de ventana deslizante. Esto permite dividir textos largos en segmentos cortos y manejables. Posteriormente, se combinan las predicciones generadas para cada segmento, obteniendo las predicciones finales que se presentan a través del módulo de chatbot, en la ventana modal de los resultados de recomendación.

Figura 8

Proceso de integración del modelo BERT en el servidor Flask y chatbot.



Nota. Se enfoca en la integración de los modelos al chatbot. Elabora por los autores.

2.3 Elaboración de la lista sprints

Después de describir el diseño predefinido, se procede a la elaboración de la lista de sprints con actividades alineadas a los objetivos específicos del proyecto.

2.4 Lista de actividades

OE1: Realizar una revisión de la efectividad de las orientaciones psicoterapéuticas en base a los diferentes tipos de trastornos.

Tabla 3

Lista de actividades. Objetivo específico 1.

No.	Actividades
1	Exploración y recopilación inicial de temas relevantes de psicoterapia.

2	Elaboración del diccionario de datos, conformada por la identificación y extracción de textos significativos, frases clave y palabras clave de psicoterapia.
3	Establecer un formato para el diccionario con tres parámetros, para segmentar la información de manera perceptible y organizada

Nota. Se toma en consideración los objetivos específicos del proyecto para realizar las diferentes actividades. Elaborado por los autores.

OE2: Realizar un análisis de aplicaciones y documentos relacionados al desarrollo de chatbots para detección de trastornos psicológicos.

Tabla 4

Lista de actividades. Objetivo específico 2.

No.	Actividades
1	Exploración y recopilación de documentos relacionados al desarrollo de chatbot para detección de trastornos psicológicos.
2	Estudio de análisis de aplicaciones de chatbot para tratamientos psicológicos.
3	Configuración del entorno de desarrollo en React-Native.
4	Creación de maquetas y prototipos de la interfaz de usuario.
5	Evaluación y ajustes de la interfaz de usuario.

Nota. Se toma en consideración los objetivos específicos del proyecto para realizar las diferentes actividades. Elaborado por los autores.

OE3: Construir diferentes módulos para los usuarios que permitan interactuar con la aplicación: módulo de bienvenida, registro de usuario, inicio de sesión, módulo que muestre el resultado de la orientación psicoterapéutica recomendada para el usuario y chatbot.

Tabla 5

Lista de actividades. Objetivo específico 3.

No.	Actividades
1	Desarrollo del módulo de bienvenida, que incluye botones interactivos para inicio de sesión y registro de usuarios.

2	Desarrollo del módulo de inicio de sesión, en él se implementa el formulario con campos para el ingreso de datos de usuario y contraseña, botones interactivos con la funcionalidad de inicio de sesión y recuperación de contraseña.
3	Desarrollo del módulo del chatbot mediante la biblioteca “GriftedChat”, en él se implementan botones interactivos con la funcionalidad guardar conversación y recomendar la orientación psicoterapéutica.
4	Desarrollo de los módulos de inicio e historial de conversaciones, en la ventana de inicio se implementa los botones interactivos para acceder a las diferentes modulo que se detallan a continuación: chatbot, conversaciones recomendaciones, mi cuenta y cierre de sesión, en el módulo de conversaciones se implementan botones interactivos para acceder las conversaciones guardadas.
5	Desarrollo de los módulos de recomendaciones y mi cuenta, en el módulo de recomendaciones se implementan botones interactivos para para acceder a las recomendaciones guardadas, y en el módulo mi cuenta se implementa un apartado para mostrar la información y botones interactivos para actualizar información y contraseña.
6	Desarrollo del servidor Express e implementación del modelo para la conexión a la base de datos MongoDB utilizando JavaScript.
7	Desarrollo de los modelos en el servidor Express para inicio de sesión, registro de usuarios, conversaciones y recomendaciones.
8	Desarrollo de API utilizando el framework Express para enviar y recuperar información de la base de datos MongoDB a través de la aplicación móvil.
9	Desarrollo e integración de los API creados en Express.
10	Desarrollo del servidor Flask y configuración para consumir servicio de OpenAi utilizando Python.
11	Implementar el procesamiento del lenguaje natural (NLP) utilizando la biblioteca transformers y el modelo BERT.
12	Integración de los mapeos inversos y modelos BERT pre-entrenados, los mapeos y modelos se integran en el servidor Flask para que sean accesibles desde el módulo de chatbot.

13	Desarrollo de una función de procesamiento especial para textos extensos, utilizando una técnica de ventana deslizante. Esto permite dividir textos largos en segmentos cortos y manejables.
14	Desarrollo e integración de los API creados en el servidor Flask para el módulo de chatbot

Nota. Se toma en consideración los objetivos específicos del proyecto para realizar las diferentes actividades. Elaborado por los autores.

Se implementan 9 sprints de acuerdo a los objetivos y actividades definidas, con revisiones al final de cada sprint para evaluar si se procede con el siguiente o si es necesario corregir posibles errores en alguna actividad. Se detallan a continuación los sprints establecidos.

Tabla 6

Progreso del sprint 1.

Sprint: No.1
Objetivo específico No.1
ACT.1: Exploración y recopilación inicial de temas relevantes de psicoterapia.
ACT.2: Elaboración del diccionario de datos, conformada por la identificación y extracción de textos significativos, frases clave y palabras clave de psicoterapia.
ACT.3: Establecer un formato para el diccionario con tres parámetros, para segmentar la información de manera perceptible y organizada.

Nota. Actividades realizadas para el objetivo específico 1. Elaborado por los autores.

Tabla 7

Progreso del sprint 2.

Sprint: No.2
Objetivo específico No.2
ACT.1: Exploración y recopilación de documentos relacionados al desarrollo de chatbot para detección de trastornos psicológicos.
ACT.2: Estudio de análisis de aplicaciones de chatbot para tratamientos psicológicos.
ACT.3: Configuración del entorno de desarrollo en React-Native.
ACT.4: Creación de maquetas y prototipos de la interfaz de usuario.

ACT.5: Evaluación y ajustes de la interfaz de usuario.

Nota. Actividades realizadas para el objetivo específico 2. Elaborado por los autores.

Tabla 8

Progreso del sprint 3.

Sprint: No.3
Objetivo específico No.2
ACT3: Configuración del entorno de desarrollo en React-Native.
ACT4: Creación de maquetas y prototipos de la interfaz de usuario.
ACT5: Evaluación y ajustes de la interfaz de usuario.
Objetivo específico No.3
ACT.1: Desarrollo del módulo de bienvenida, que incluye botones interactivos para inicio de sesión y registro de usuarios.
ACT.2: Desarrollo del módulo de inicio de sesión, en él se implementa el formulario con campos para el ingreso de datos de usuario y contraseña, botones interactivos con la funcionalidad de inicio de sesión y recuperación de contraseña.
ACT.3: Desarrollo del módulo del chatbot mediante la biblioteca "GriftedChat", en él se implementan botones interactivos con la funcionalidad guardar conversación y recomendar la orientación psicoterapéutica.

Nota. Actividades realizadas objetivo específico 3 e iteraciones de las actividades del objetivo específico 2. Elaborado por los autores.

Tabla 9

Progreso del sprint 4.

Sprint: No.4
Objetivo específico No.3
ACT.2: Desarrollo del módulo de inicio de sesión, en él se implementa el formulario con campos para el ingreso de datos de usuario y contraseña, botones interactivos con la funcionalidad de inicio de sesión y recuperación de contraseña.
ACT.3: Desarrollo del módulo del chatbot mediante la biblioteca "GriftedChat", en él se implementan botones interactivos con la funcionalidad guardar conversación y recomendar la orientación psicoterapéutica.

ACT.4: Desarrollo de los módulos de inicio e historial de conversaciones, en la ventana de inicio se implementa los botones interactivos para acceder a las diferentes modulo que se detallan a continuación: chatbot, conversaciones recomendaciones, mi cuenta y cierre de sesión, en el módulo de conversaciones se implementan botones interactivos para acceder las conversaciones guardadas.
ACT.5: Desarrollo de los módulos de recomendaciones y mi cuenta, en el módulo de recomendaciones se implementan botones interactivos para para acceder a las recomendaciones guardadas, y en el módulo mi cuenta se implementa un apartado para mostrar la información y botones interactivos para actualizar información y contraseña.

Nota. Actividades realizadas objetivo específico 3. Elaborado por los autores.

Tabla 10

Progreso del sprint 5.

Sprint: No.5
Objetivo específico No.3
ACT.3: Desarrollo del módulo del chatbot mediante la biblioteca "GriftedChat", en él se implementan botones interactivos con la funcionalidad guardar conversación y recomendar la orientación psicoterapéutica.
ACT.5: Desarrollo de los módulos de recomendaciones y mi cuenta, en el módulo de recomendaciones se implementan botones interactivos para para acceder a las recomendaciones guardadas, y en el módulo mi cuenta se implementa un apartado para mostrar la información y botones interactivos para actualizar información y contraseña.
ACT.6: Desarrollo del servidor Express e implementación del modelo para la conexión a la base de datos MongoDB utilizando JavaScript.
ACT.7: Desarrollo de los modelos en el servidor Express para inicio de sesión, registro de usuarios, conversaciones y recomendaciones.
ACT.8: Desarrollo de API utilizando el framework Express para enviar y recuperar información de la base de datos MongoDB a través de la aplicación móvil.

Nota. Actividades realizadas objetivo específico 3. Elaborado por los autores.

Tabla 11

Progreso del sprint 6.

Sprint: No.6
Objetivo específico No.3
ACT.3: Desarrollo del módulo del chatbot mediante la biblioteca "GriftedChat", en él se implementan botones interactivos con la funcionalidad guardar conversación y recomendar la orientación psicoterapéutica.
ACT.8: Desarrollo de API utilizando el framework Express para enviar y recuperar información de la base de datos MongoDB a través de la aplicación móvil.
ACT.9: Desarrollo e integración de los API creados en Express.
ACT.10: Desarrollo del servidor Flask y configuración para consumir el servicio de OpenAI utilizando Python.

Nota. Actividades realizadas objetivo específico 3. Elaborado por los autores.

Tabla 12

Progreso del sprint 7.

Sprint: No.7
Objetivo específico No.3
ACT.3: Desarrollo del módulo del chatbot mediante la biblioteca "GriftedChat", en él se implementan botones interactivos con la funcionalidad guardar conversación y recomendar la orientación psicoterapéutica.
ACT.9: Desarrollo e integración de los API creados en Express.
ACT.11: Implementar el procesamiento del lenguaje natural (NLP) utilizando la biblioteca transformers y el modelo BERT.
ACT.12: Integración de los mapeos inversos y modelos BERT pre-entrenados, los mapeos y modelos se integran en el servidor Flask para que sean accesibles desde el módulo de chatbot.

Nota. Actividades realizadas objetivo específico 3. Elaborado por los autores.

Tabla 13

Progreso del sprint 8.

Sprint: No.8
Objetivo específico No.3
ACT.3: Desarrollo del módulo del chatbot mediante la biblioteca "GriftedChat", en él se implementan botones interactivos con la funcionalidad guardar conversación y recomendar la orientación psicoterapéutica.
ACT.11: Implementar el procesamiento del lenguaje natural (NLP) utilizando la biblioteca transformers y el modelo BERT.
ACT.12: Integración de los mapeos inversos y modelos BERT pre-entrenados, los mapeos y modelos se integran en el servidor Flask para que sean accesibles desde el módulo de chatbot.

Nota. Actividades realizadas objetivo específico 3. Elaborado por los autores.

Tabla 14

Progreso del sprint 9.

Sprint: No.9
Objetivo específico No.3
ACT.11: Implementar el procesamiento del lenguaje natural (NLP) utilizando la biblioteca transformers y el modelo BERT.
ACT.12: Integración de los mapeos inversos y modelos BERT pre-entrenados, los mapeos y modelos se integran en el servidor Flask para que sean accesibles desde el módulo de chatbot.
ACT.13: Desarrollo de una función de procesamiento especial para textos extensos, utilizando una técnica de ventana deslizante. Esto permite dividir textos largos en segmentos cortos y manejables.
ACT.14: Desarrollo e integración de los API creados en el servidor Flask para el módulo de chatbot.

Nota. Actividades realizadas objetivo específico 3. Elaborado por los autores.

2.5 Requerimientos funcionales y no funcionales

2.5.1 Requerimientos funcionales

En la Tabla 15 se mostrará todas las funcionalidades utilizadas en el proyecto:

Tabla 15

Requerimientos funcionales.

N°	Código	Descripción
1	RF01	**Registro de usuarios:** Al proporcionar información personal, los usuarios podrán incluir datos como nombres, apellidos, número de cédula, género, dirección de correo electrónico, número de contacto, nombre de usuario y contraseña. Además, se aplicarán diversas validaciones, como verificar la longitud y composición de la contraseña, así como comprobar si la cédula, correo electrónico, número de contacto y nombre de usuario ya están registrados en el sistema. Este proceso asegurará que la información proporcionada sea precisa y única. Tras completar este proceso de registro, los usuarios podrán iniciar sesión en la aplicación utilizando tanto su nombre de usuario como su contraseña.
2	RF02	**Inicio de sesión:** Este proceso permitirá a los usuarios autenticarse mediante credenciales previamente establecidas, como nombre de usuario y contraseña.
3	RF03	**Cambiar nombre de usuario:** En este proceso permite cambiar el nombre de usuario utilizando la cedula como validador para el cambio.
4	RF04	**Cambiar contraseña:** En este proceso el usuario podrá cambiar la contraseña mediante el nombre del usuario actual y la contraseña actual.
5	RF05	**Chatbot:** Este proceso se basa en realizar preguntas relevantes sobre la situación emocional o mental del usuario, sus experiencias, síntomas, y otros factores relevantes. Basándose en las respuestas proporcionadas, el chatbot utilizaría algoritmos y reglas predefinidas para sugerir una orientación psicoterapéutica específica, como terapia cognitivo-conductual, psicoanálisis, terapia de pareja, entre otras.
6	RF06	**Recomendar la orientación psicoterapéutica:**

	Este procedimiento empleará algoritmos de procesamiento del lenguaje natural (NLP), que analizan la información del usuario, es decir, problemas psicológicos, eventos significativos o malestar del paciente. Su objetivo es ofrecer recomendaciones que se asemejen al diagnóstico que realizaría un profesional de la salud. La aplicación se dedicará a brindar sugerencias personalizadas, ajustadas a las necesidades individuales de cada usuario, garantizando así una experiencia terapéutica que favorece la interacción entre el paciente y el asistente virtual.

Nota. Requerimientos funcionales del proyecto. Elaborado por los autores.

2.5.2 Requerimientos no funcionales

Usabilidad: Se requiere que la interfaz de usuario en el aplicativo móvil sea de fácil comprensión y fácil de utilizar durante la interacción. Las maquetas y prototipos deben tener en cuenta la usabilidad, accesibilidad y estética, y se deben proporcionar mensajes claros en el Front-End para guiar a los usuarios durante el registro y otras interacciones.

Seguridad: El aplicativo móvil debe resguardar la información sensible de los usuarios, especialmente durante el Registro e Inicio de Sesión. Se enfatiza el uso de encriptación para almacenar contraseñas y asegurar la confidencialidad de los datos. Además, se subraya la importancia de validaciones y mensajes de alerta en el Front-End para mejorar la seguridad, identificando errores como usuario no registrado o contraseña incorrecta.

Responsive: El aplicativo móvil debe ser adaptable a diversos tamaños de pantalla, asegurando una experiencia coherente en dispositivos móviles. Se destaca la necesidad de que los botones y elementos interactivos sean fácilmente seleccionables y utilizables en pantallas de diferentes dimensiones.

Escalable: El aplicativo móvil debe adaptarse al crecimiento futuro de usuarios y datos. Para lograr esto, la arquitectura debe incluir un espacio para almacenar los datos. Para ello, se establece la base de datos MongoDB y se implementan servidores Express y Flask para la interacción. La aplicación móvil debe ser diseñada con escalabilidad en mente, se destaca la

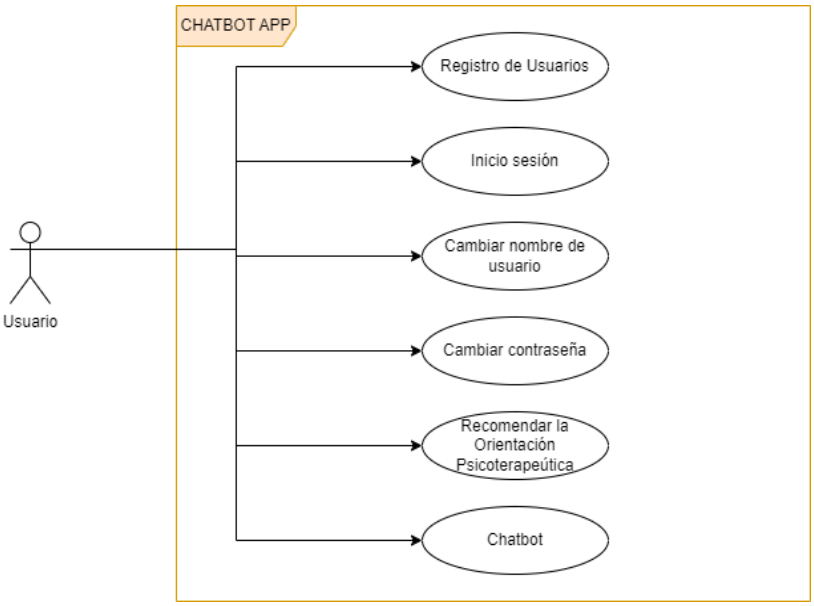
importancia de implementar prácticas eficientes para gestionar recursos y garantizar una respuesta rápida del servidor durante el uso del chatbot al consumir el servicio de OpenAI (GPT 3.5 Turbo).

2.6 Diagrama de casos de uso

2.6.1 Módulo principal de la aplicación móvil

Figura 9

Módulo principal de la aplicación móvil.



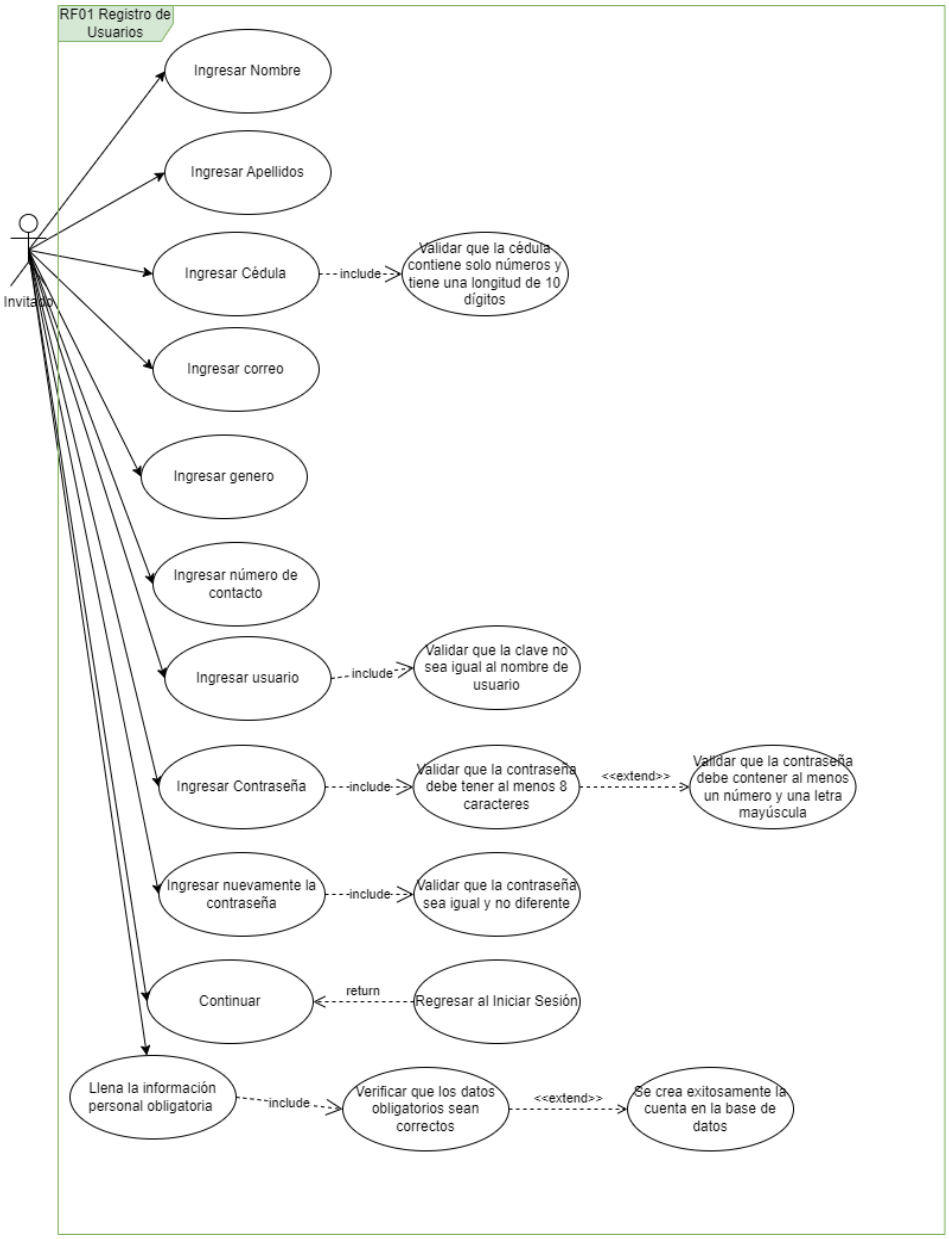
Nota. Diagrama de caso de uso módulo principal. Elaborado por los autores.

De acuerdo con lo representado en la Figura 9 el usuario podrá ingresar a cada uno de los módulos mostrados en la imagen anterior en el cual se compone: registro de usuario, inicio sesión, cambiar nombre de usuario, cambiar contraseña, recomendar la orientación psicoterapéutica y chatbot.

2.6.2 RF01 Registro de usuario

Figura 10

Casos de usos. RF01 Registro de usuarios.



Nota. Diagrama de caso de uso registrar usuarios. Elaborado por los autores.

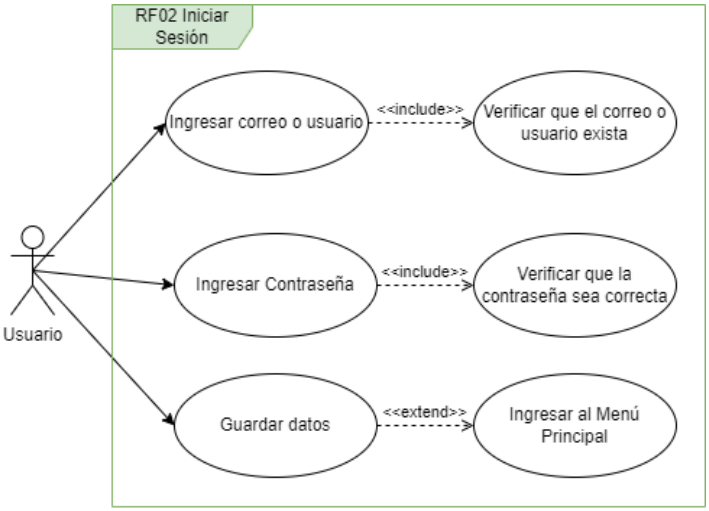
De acuerdo con lo representado en la Figura 10, el usuario invitado podrá registrarse en "PsicoApp" al seleccionar el módulo de registro de usuarios que se encuentra en el segmento principal de la aplicación móvil que aparece al desplegar la plataforma. El usuario deberá completar la información personal solicitada en el módulo. Si la información es correcta, se

crea la cuenta del usuario y por consiguiente podrá ingresar a la aplicación a través del módulo iniciar sesión.

2.6.3 RF02 Iniciar sesión

Figura 11

Casos de usos. RF02 Iniciar sesión.



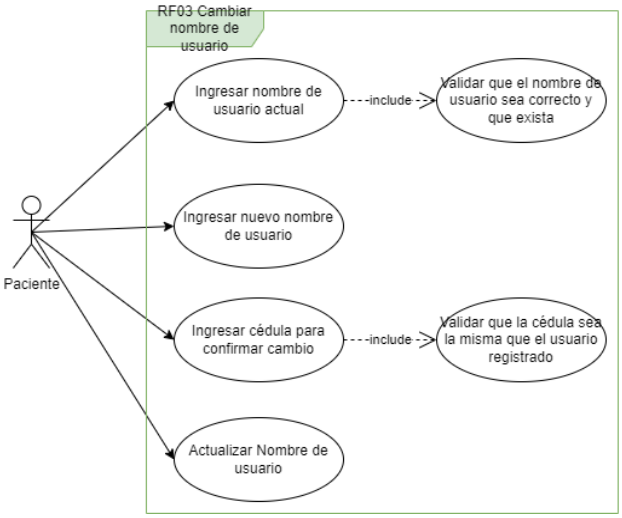
Nota. Diagrama de caso de uso iniciar sesión. Elaborado por los autores.

De acuerdo con lo representado en la Figura 11, introduciendo su nombre de usuario y contraseña, el usuario accede al formulario de inicio de sesión. Posteriormente, el aplicativo realiza una validación de los datos y determinar si son correctos mediante la comparación con la información obtenida desde la base de datos.

2.6.4 RF03 Cambiar nombre de usuario

Figura 12

Casos de usos. RF03 Cambiar nombre de usuario.



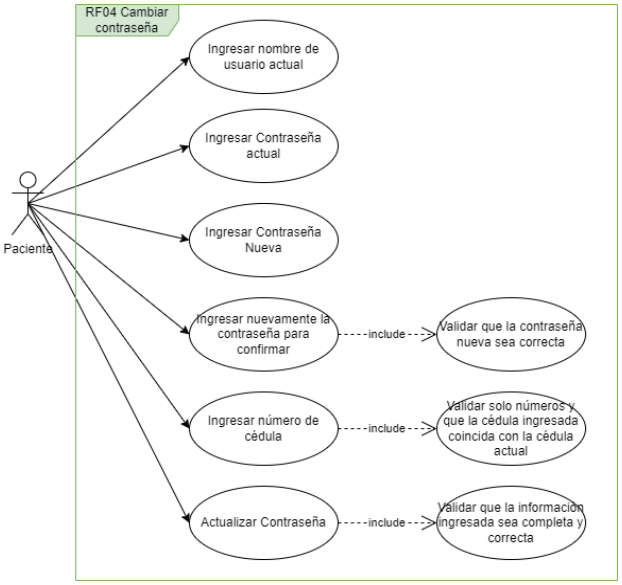
Nota. Diagrama de caso de uso cambiar nombre de usuario. Elaborado por los autores.

De acuerdo con lo representado en la Figura 12, el usuario podrá actualizar el nombre de usuario utilizando como validador el número de cédula. Adicional, si el usuario quiere actualizar el nombre de usuario, deberá completar todos los campos de forma correcta.

2.6.5 RF04 Cambiar contraseña

Figura 13

Casos de usos. RF04 Cambiar contraseña.



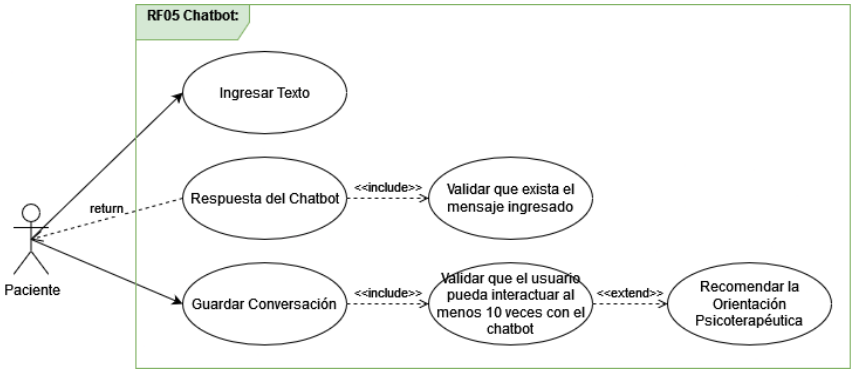
Nota. Diagrama de caso de uso cambiar contraseña. Elaborado por los autores.

De acuerdo con lo representado en la Figura 13, el usuario podrá cambiar la contraseña siempre y cuando ingrese su nombre del usuario, su número de cédula y contraseña actual. También se valida que el usuario complete la información requerida para que el usuario pueda actualizar la contraseña.

2.6.6 RF05 Chatbot

Figura 14

Casos de usos. RF05 Chatbot.



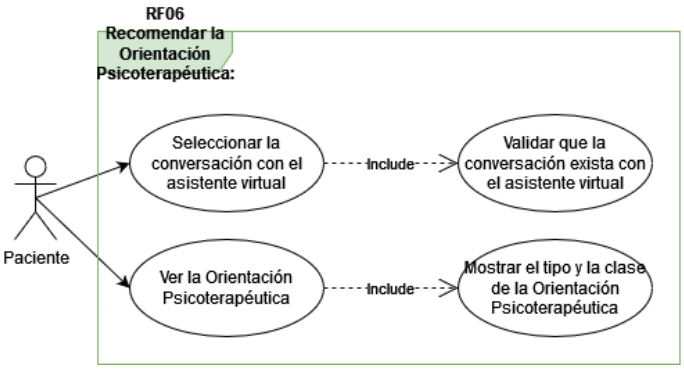
Nota. Diagrama de caso de uso chatbot. Elaborado por los autores.

De acuerdo con lo representado en la Figura 14, el usuario podrá acceder al módulo de chatbot al ingresar sus síntomas o problemas psicológicos en la caja de texto proporcionada. Posteriormente, el chatbot responderá a la consulta del usuario, buscando comprender la situación en la que se encuentra. Además, se requieren al menos 10 interacciones con el chatbot para que pueda recomendar una orientación psicoterapéutica adecuada. Para recibir la recomendación de orientación psicoterapéutica, el usuario debe guardar la conversación, asignándole un título para su referencia. Finalmente, se proporciona un botón que permite al usuario acceder a la orientación psicoterapéutica recomendada.

2.6.7 RF06 Recomendar la orientación psicoterapéutica

Figura 15

Casos de usos. RF06 Recomendar la orientación psicoterapéutica.



Nota. Diagrama de caso de uso recomendar la orientación psicoterapéutica. Elaborado por los autores.

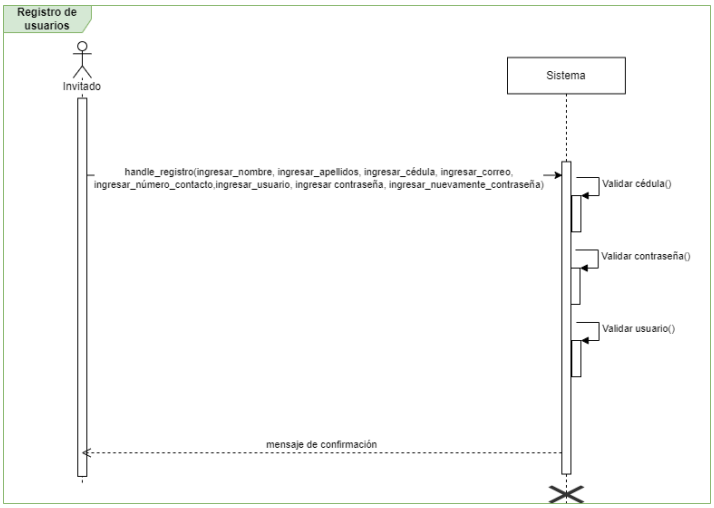
Según la Figura 15, el usuario podrá visualizar su orientación psicoterapéutica a través de las conversaciones previas mantenidas con el asistente virtual. Por tanto, es crucial que el usuario describa detalladamente su situación al asistente virtual, para recibir recomendaciones más cercanas y adaptadas a su problemática.

2.7 Diagrama de secuencia

2.7.1 RF01 Registro de usuario

Figura 16

Diagrama de secuencia. Registro de usuarios.



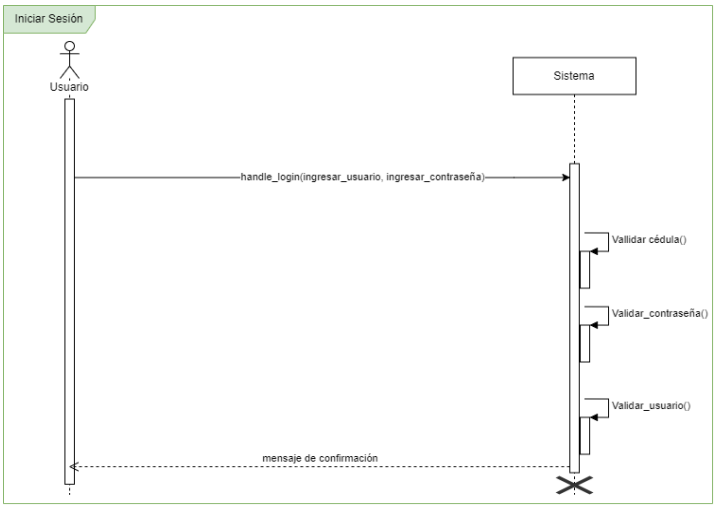
Nota. Diagrama de secuencia registrar usuario. Elaborado por los autores.

Según la Figura 16, los usuarios pueden registrarse en PsicoApp (Nombre del aplicativo) seleccionando el módulo "registro de usuarios" en la pantalla principal. Al completar la información personal requerida, se valida si es correcta y pasa a la creación de la cuenta, y posteriormente permitiendo acceder al aplicativo.

2.7.2 RF02 Iniciar sesión

Figura 17

Diagrama de secuencia. Iniciar sesión.



Nota. Diagrama de secuencia iniciar sesión. Elaborado por los autores.

En la Figura 17, introduciendo su nombre de usuario y contraseña, el usuario accede al formulario de inicio de sesión. Posteriormente, el aplicativo realiza una validación de los datos y determinar si son correctos mediante la comparación con la información obtenida desde la base de datos.

2.7.3 RF03 Cambiar nombre de usuario

Figura 18

Diagrama de secuencia. Cambiar nombre de usuario.

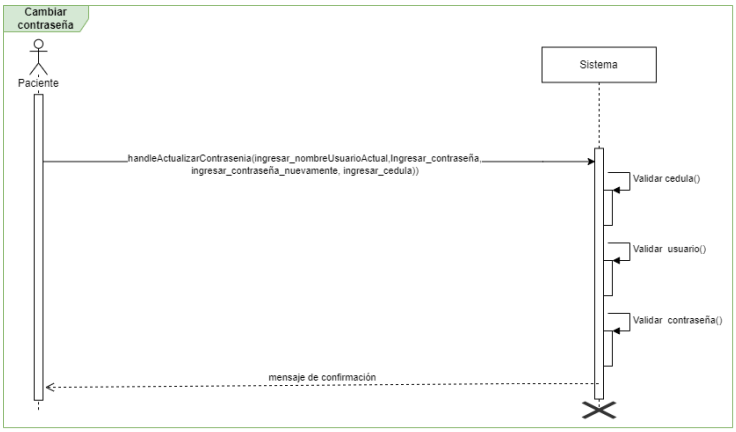
Nota. Diagrama de secuencia cambiar el nombre de usuario. Elaborado por los autores.

En la Figura 18, el usuario tiene la opción de actualizar su nombre de usuario utilizando como validador el número de cédula. Además, para actualizar exitosamente el nombre de usuario, es necesario completar todos los campos de manera precisa.

2.7.4 RF04 Cambiar contraseña

Figura 19

Diagrama de secuencia. Cambiar contraseña.



Nota. Diagrama de secuencia cambiar contraseña. Elaborado por los autores.

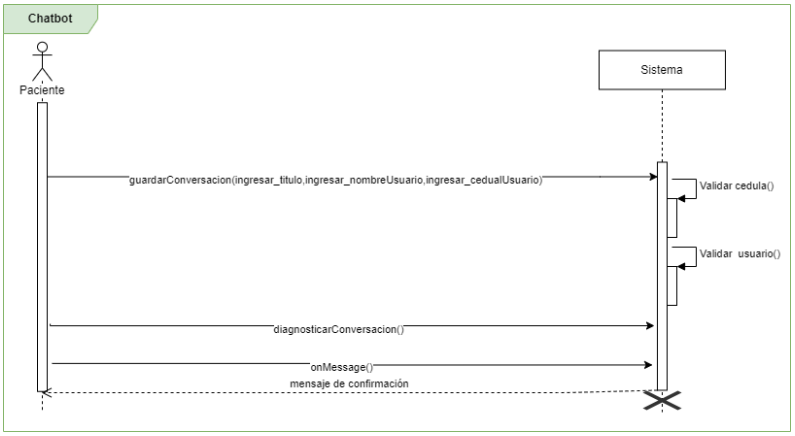
En la Figura 19, la actualización de la contraseña está disponible para el usuario al ingresar la contraseña actual con la que inicio sesión, luego ingresa la contraseña y utiliza la

cédula junto con la contraseña que inicio sesión como validador para realizar el cambio. Se verifica que la información necesaria esté completa antes de permitir la actualización de la contraseña.

2.7.5 RF05 Chatbot

Figura 20

Diagrama de secuencia. Chatbot.



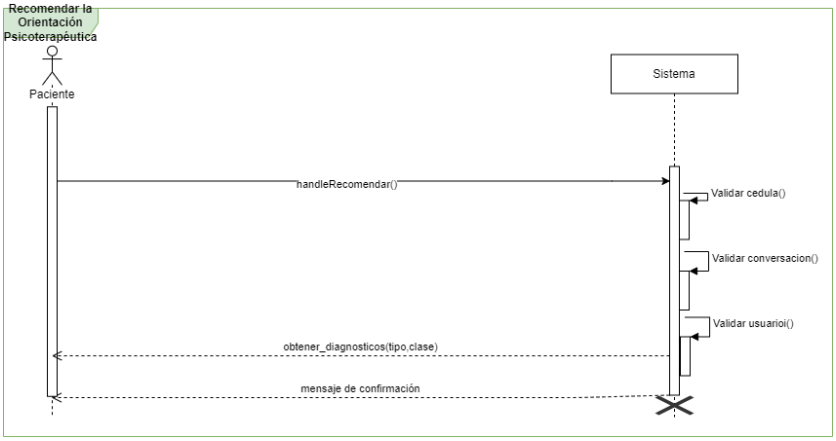
Nota. Diagrama de secuencia chatbot. Elaborado por los autores.

En la Figura 20, El usuario interactúa con el chatbot al ingresar sus síntomas. Tras al menos 10 interacciones, el chatbot recomienda orientación psicoterapéutica. Guardar la conversación y utilizar un botón permite al usuario acceder a la recomendación.

2.7.6 RF06 Recomendar la orientación psicoterapéutica

Figura 21

Diagrama de secuencia. Recomendar la orientación psicoterapéutica.



Nota. Diagrama de secuencia recomendar orientación psicoterapéutica. Elaborado por los autores.

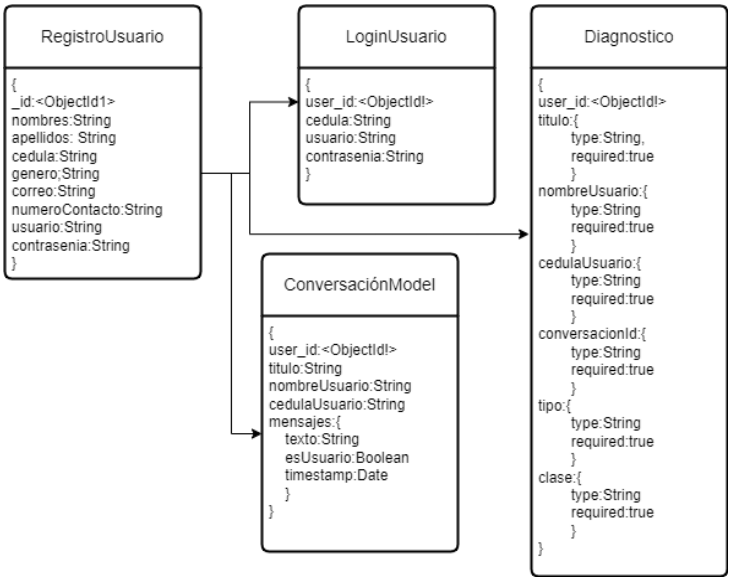
Según la Figura 21, el usuario puede acceder a su orientación psicoterapéutica mediante las conversaciones previas con el asistente virtual. Es fundamental que el usuario describa detalladamente su situación al asistente para recibir recomendaciones más precisas y personalizadas acorde a sus necesidades.

2.8 Modelado

Modelado para MongoDB

Figura 22

Modelado MongoDB.



Nota. Esquemas de tablas utilizado en la base de datos del proyecto. Elaborado por los autores.

2.9 Diccionario de datos

A continuación, se detallan todas las entidades NoSQL utilizadas, incluyendo información sobre cada campo, su tipo de dato y su descripción correspondiente.

Tabla 16

Tabla LoginUsuario. Diccionario de datos.

LoginUsuario		
Campo	**Tipo de dato**	**Descripción**
cedula	String	Cédula registrada en la aplicación
usuario	String	Nombre o alias de usuario registrado
contrasenia	String	Contraseña registrada

Nota. Se detalla las entidades NoSQL como los nombres de las variables utilizadas, el tipo de dato y su descripción. Elaborado por los autores.

Tabla 17

Tabla ConversionModel. Diccionario de datos.

ConversacionModel		
Campo	**Tipo de dato**	**Descripción**
titulo	String	Nombre que se va a guardar la conversación al asistente virtual
nombreUsuario	String	Nombre del usuario que quiere guardar la conversación
cedulaUsuario	String	Cédula del usuario que quiere guardar la conversación
mensaje:texto	String	Conversación del chatbot y el usuario
mensaje:esUsuario	Boolean	Define si la conversación corresponde al chatbot sea igual a 0. Si la conversación corresponde al usuario sea igual a 1.

mensaje:timestamp	date	Fecha de creación de las conversaciones entre el chatbot y el usuario.

Nota. Se detalla las entidades NoSQL como los nombres de las variables utilizadas, el tipo de dato y su descripción. Elaborado por los autores.

Tabla 18

Tabla Diagnostico. Diccionario de datos.

Diagnostico		
Campo	**Tipo de dato**	**Descripción**
titulo:type	String	Nombre de la conversación guardada por el usuario
nombreUsuario:type	String	Nombre del usuario
cedulaUsuario:type	String	Cédula del usuario
conversacionId:type	String	Identificador único de la conversación guardada por el usuario
tipo:type	String	Tipo de orientación psicoterapéutica
clase:type	String	Clase de orientación psicoterapéutica

Nota. Se detalla las entidades NoSQL como los nombres de las variables utilizadas, el tipo de dato y su descripción. Elaborado por los autores.

Tabla 19

Tabla RegistroUsuario. Diccionario de datos.

RegistroUsuario		
Campo	**Tipo de dato**	**Descripción**
nombres	String	El nombre del usuario
apellidos	String	Apellidos del usuario
cedula	String	Cédula del usuario

genero	String	Género del usuario (Masculino o Femenino)
correo	String	Correo del usuario
numeroContacto	String	Número de celular o convencional del usuario
usuario	String	Nombre o alias de usuario
contrasenia	String	Contraseña que debe ingresar el usuario

Nota. Se detalla las entidades NoSQL como los nombres de las variables utilizadas, el tipo de dato y su descripción. Elaborado por los autores.

CAPÍTULO III: DISEÑO Y DESPLIGUE

3.1 Arquitectura

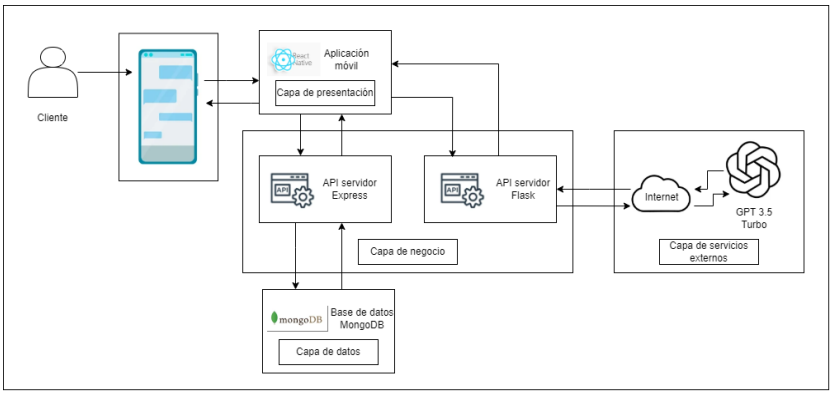
En el contexto de la arquitectura del proyecto es diseñado para dispositivos Android, el objetivo es describir el proceso de implementación para la solución del problema, se detalla la manera de organización y construcción del prototipo de aplicación móvil, con la finalidad de entender el funcionamiento y como se relaciona e interactúan las distintas partes del software.

3.1.1 Entorno de desarrollo

Con el fin de ejecutar el prototipo, la solución propuesta es un diseño basado en una arquitectura de n capas, para el caso se utilizarán 3 capas conformadas por presentación, negocio y datos. La arquitectura establecida permite dividir el sistema de la siguiente forma: lograr la separación entre la capa de presentación y la capa de negocio, consiguiendo de este modo la segregación de la capa de negocio con respecto a la capa de datos. De acuerdo al orden jerárquico de comunicación entre las capas durante el desarrollo, se implantarán los componentes necesarios para que cada capa cumpla la función requerida.

Figura 23

Arquitectura n capas. Vista de desarrollo.



Nota. Arquitectura n capas propuesto. Elaborado por los autores.

La estructura del prototipo de la aplicación móvil abarca una capa de datos que contiene un sistema de gestión de bases de datos NoSQL, empleando MongoDB para este propósito específico. La capa de negocio está compuesta por dos servidores uno en Express y otro en

Flask, se desarrolla el API en Express para la conexión a MongoDB y a través de ello poder almacenar y recuperar los datos para visualizarlos a través de la aplicación móvil.

Se desarrolla un servidor en Flask para conectar con el servicio de OpenAI (GPT 3.5 Turbo) utilizando Python para facilitar la interacción con el servicio de OpenAI y utilizarlo en el módulo de chatbot de la aplicación móvil.

En el servidor Flask se implementan las librerías para procesamiento del lenguaje natural NLTK, mapeos inversos y modelos pre-entrenados de BERT utilizando Python para realizar las predicciones en base a una conversación almacena en MongoDB.

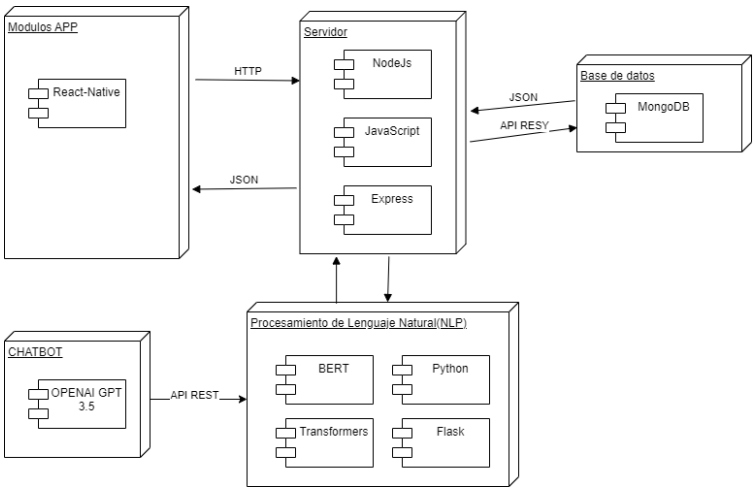
Utilizando React-Native, se ha desarrollado la capa de presentación de la interfaz de usuario implementando el CLI (Command Line Interface) de React-Native para el desarrollo de los siguientes módulos: bienvenida, registro de usuario, inicio de sesión y chatbot.

3.1.2 Diagrama de despliegue de la aplicación

El esquema de despliegue de la aplicación móvil muestra la interacción entre los distintos elementos que se aprecian a continuación en la Figura 24.

Figura 24

Diagrama de despliegue.



Nota. Diagrama de despliegue propuesto. Elaborado por los autores.

Módulos App (React-Native): La interfaz de usuario constituye la interacción entre los usuarios y la aplicación. Incluye componentes fundamentales como registro, inicio de sesión, bienvenida, gestión de cuentas, conversaciones, chatbot, entre otros. Estos elementos están disponibles mediante una aplicación móvil que utiliza la plataforma React-Native, simplificando la comunicación con la API del Back-End mediante el framework React.

La estructura del proyecto se organiza en diversos componentes, cada uno con su propio conjunto de estilos proporcionados por React-Native, aprovechando el lenguaje de programación JavaScript. Además, con el fin de crear los modelos, se utiliza Python. Este enfoque ayuda con la creación de una interfaz intuitiva y eficiente para los usuarios, respaldada por la potencia y versatilidad de las tecnologías empleadas.

Servidor (Node.js): Al ser un entorno de ejecución dedicado a JavaScript, Node.js proporciona una API que permite la relación entre el cliente y el servidor, ejecutando funciones a través de peticiones GET, PUT y POST. El servidor establece una conexión con MongoDB.

Servidor (Express): Dado que es un marco para aplicaciones en Node.js, desempeña el papel fundamental de un servidor web para gestionar las solicitudes HTTP del cliente y las direcciona hacia el servidor correspondiente. Simplifica la creación y gestión de rutas, permitiendo asociar funciones de controlador con rutas específicas, lo que optimiza el manejo de solicitudes y respuestas.

Base de datos (MongoDB): Simplifica la interacción desde el servidor de aplicaciones mediante un API REST desarrollado con la biblioteca “Mongoose” (Soporta Node.js). Además, el servidor puede recopilar de manera eficiente los datos solicitados por el cliente, asegurando una comunicación fluida y optimizada entre MongoDB y la aplicación.

Chatbot: Busca mejorar las conversaciones naturales con el usuario. Su objetivo es comprender y generar respuestas coherentes en diversos temas. La integración del API de

OpenAI facilita la interacción entre el usuario y la aplicación con el fin de mejorar la experiencia del usuario.

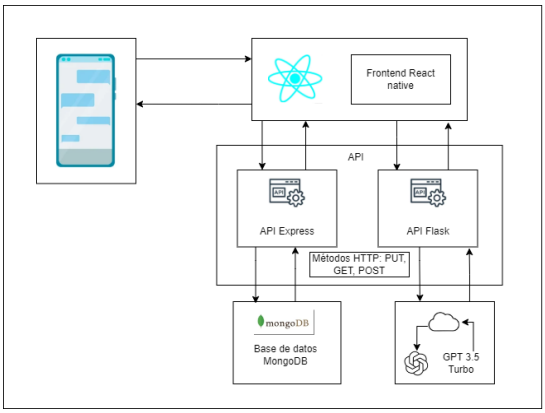
Procesamiento natural del lenguaje (NLP): Se implementa el servidor Flask para gestionar solicitudes y respuestas en el módulo de chatbot, así como la conexión con el servicio de OpenAI (GPT 3.5 Turbo) mediante Python. Además, se integra procesamiento del lenguaje natural (NLP) utilizando la biblioteca transformers y el modelo BERT pre-entrenado en español. Se efectúa la etapa de preprocesamiento de datos, continuando con el entrenamiento del modelo y el ajuste de los pesos, se realiza la evaluación de su rendimiento. Se emplea dropout y optimizadores con regularización L2 que mejoran la capacidad de generalizar los datos. La incorporación de modelos BERT pre-entrenados en el servidor Flask habilita la realización de predicciones utilizando sofisticadas técnicas de procesamiento de texto. Este proceso culmina con la presentación de los resultados a través del módulo de chatbot, que se materializa mediante una ventana modal de recomendación.

3.1.3 Arquitectura API

En la Figura 25**,** se define la estructura que facilita la interacción entre la base de datos y el servicio de OpenAI en la aplicación móvil a través de la API.

Figura 25

Arquitectura de la API.



Nota. Arquitectura en API propuesto. Elaborado por los autores.

3.2 Desarrollo de la aplicación

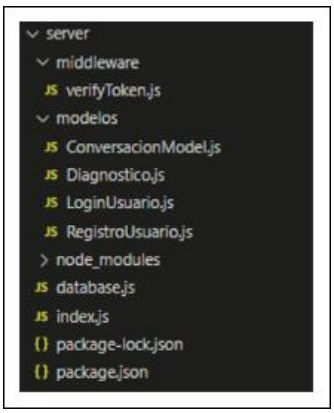
En el siguiente apartado, se presentará el desarrollo de la aplicación, estructurándolo en distintos componentes: Back-End, Front-End y aprendizaje automático (Machine learning). En cada elemento se mencionará su funcionalidad y progreso en proyecto.

3.2.1 Back-End

En la infraestructura del Back-End, se emplea Express como servicio para gestionar rutas por API y facilitar la interacción dinámica entre el cliente y el servidor permitiendo conectar con la base de datos para recuperar o almacenar datos. MongoDB compass se elige como el gestor de bases de datos NoSQL para el almacenamiento local de datos. Flask se emplea como servicio para el manejo de solicitudes por API y gestionar la interacción entre OpenAI (GPT 3.5 Turbo) permitiendo aprovechar su capacidad avanzada de procesamiento de lenguaje natural, mejorando la coherencia y la relevancia de las respuestas generadas en las conversaciones entre el usuario y el módulo de chatbot.

Figura 26

Servidor Express. Vista desde Visual Studio Code.



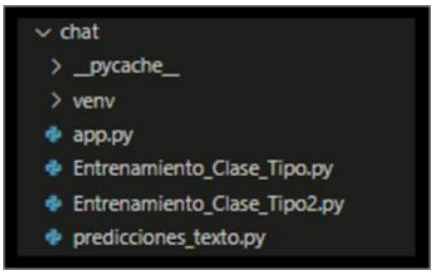
Nota. Archivos Javacript montados en el servidor Express. Elaborado por los autores.

Como se menciona en la Figura 26, el servidor Express además de conectarse con la base de datos también contiene la verificación de autenticación del token. Este token es creado

cuando el usuario crea su contraseña en el módulo de registro. Dentro del servidor también tiene dependencias instaladas como: bycrypt, jwt y verifytoken.

Figura 27

Servidor Flask. Vista desde Visual Studio Code.



Nota. Archivos de Python montados en el servidor de Flask. Elaborado por los autores.

En la Figura 27**,** se muestra la importancia de Flask en el entrenamiento del modelo de clasificación de texto, funciones de preprocesamiento y predicción o resultado del tipo y clase de la orientación psicoterapéutica. Para este apartado se utilizó Python.

3.2.2 Códigos importantes Back-End

Figura 28

Códigos importantes. Conexión base de datos.

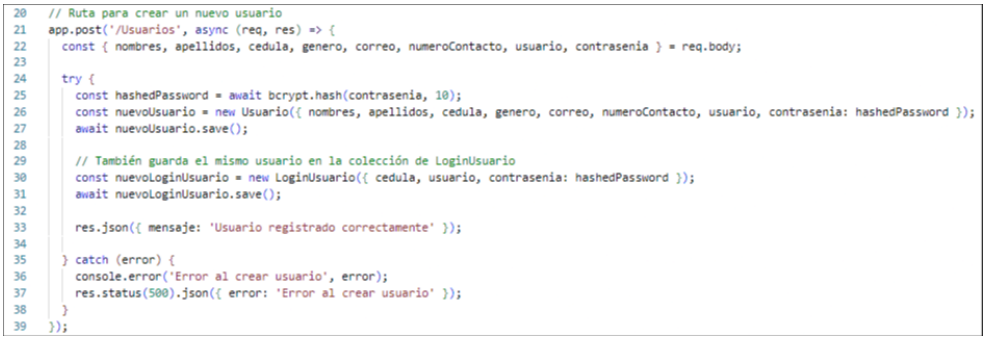
```
1  const mongoose = require('mongoose');
2
3  const connectDB = async () => {
4    try {
5      await mongoose.connect('mongodb://127.0.0.1:27017/Psicoapp', {
6        useNewUrlParser: true,
7        useUnifiedTopology: true,
8      });
9      console.log('Conexión a MongoDB exitosa');
10   } catch (error) {
11     console.error('Error al conectar a MongoDB', error);
12   }
13 };
14
15 module.exports = connectDB;
```

Nota. Código fuente acerca de la conexión hacia la base de datos. Elaborado por los autores.

En la línea 3 de la Figura 28, se define la función asíncrona con la finalidad de conectar con la base de datos. Dentro de la función, se utiliza ‘moongse.connect’ para establecer conexión. Los parámetros requeridos son la URL de la base de datos.

Figura 29

Códigos importantes. Tokenización de la contraseña del usuario.



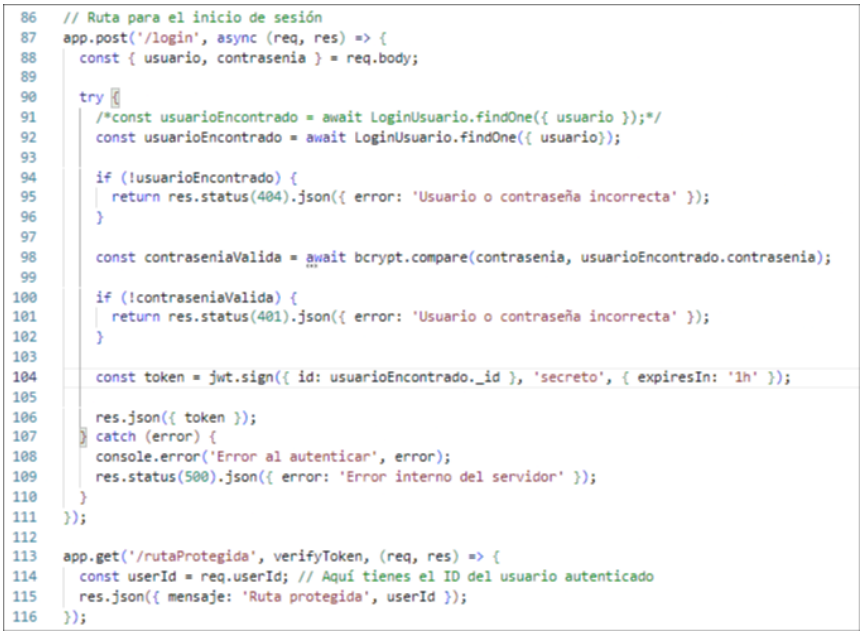
```
20  // Ruta para crear un nuevo usuario
21  app.post('/Usuarios', async (req, res) => {
22    const { nombres, apellidos, cedula, genero, correo, numeroContacto, usuario, contrasenia } = req.body;
23
24    try {
25      const hashedPassword = await bcrypt.hash(contrasenia, 10);
26      const nuevoUsuario = new Usuario({ nombres, apellidos, cedula, genero, correo, numeroContacto, usuario, contrasenia: hashedPassword });
27      await nuevoUsuario.save();
28
29      // También guarda el mismo usuario en la colección de LoginUsuario
30      const nuevoLoginUsuario = new LoginUsuario({ cedula, usuario, contrasenia: hashedPassword });
31      await nuevoLoginUsuario.save();
32
33      res.json({ mensaje: 'Usuario registrado correctamente' });
34
35    } catch (error) {
36      console.error('Error al crear usuario', error);
37      res.status(500).json({ error: 'Error al crear usuario' });
38    }
39  });
```

Nota. Código fuente acerca de la tokenización por usuario. Elaborado por los autores.

En la línea 21 se define la ruta POST para crear un nuevo usuario. En la línea 25, se utiliza para “hashear” la contraseña del usuario. El valor es el número de rondas de hash. A partir de la línea 1inea 26 hasta la línea 31, se crean instancias para crear y guardar el usuario en las tablas “Usuario” y “LoginUsuario” en la base de datos.

Figura 30

Códigos importantes. Iniciar sesión validaciones.



```
86  // Ruta para el inicio de sesión
87  app.post('/login', async (req, res) => {
88    const { usuario, contrasenia } = req.body;
89
90    try {
91      /*const usuarioEncontrado = await LoginUsuario.findOne({ usuario });*/
92      const usuarioEncontrado = await LoginUsuario.findOne({ usuario});
93
94      if (!usuarioEncontrado) {
95        return res.status(404).json({ error: 'Usuario o contraseña incorrecta' });
96      }
97
98      const contraseniaValida = await bcrypt.compare(contrasenia, usuarioEncontrado.contrasenia);
99
100     if (!contraseniaValida) {
101       return res.status(401).json({ error: 'Usuario o contraseña incorrecta' });
102     }
103
104     const token = jwt.sign({ id: usuarioEncontrado._id }, 'secreto', { expiresIn: '1h' });
105
106     res.json({ token });
107   } catch (error) {
108     console.error('Error al autenticar', error);
109     res.status(500).json({ error: 'Error interno del servidor' });
110   }
111 });
112
113 app.get('/rutaProtegida', verifyToken, (req, res) => {
114   const userId = req.userId; // Aquí tienes el ID del usuario autenticado
115   res.json({ mensaje: 'Ruta protegida', userId });
116 });
```

Nota. Código que valida el inicio de sesión por usuario. Elaborado por los autores.

En la línea 87 se define la ruta para que el usuario inicie sesión. En la línea 92, el modelo busca un usuario que este almacenado en la base de datos y que coincida con el nombre de usuario proporcionado. Si no encuentra ningún usuario con el nombre proporcionado, se devuelve una respuesta de error, indicando que el usuario o contraseña son incorrectos.

Figura 31

Códigos importantes. Verificación de los tokens de autentificación.

```
const jwt = require('jsonwebtoken');

const verifyToken = (req, res, next) => {
  const token = req.header('Authorization');
  if (!token) return res.status(403).json({ error: 'Acceso denegado' });

  try {
    const decoded = jwt.verify(token, 'secreto');
    req.userId = decoded.id;
    next();
  } catch (error) {
    res.status(401).json({ error: 'Token inválido' });
  }
};

module.exports = { verifyToken };
```

Nota. Código que verifica los tokens de autentificación. Elaborado por los autores.

En este código, su funcionalidad es verificar la validez de los tokens de autentificación, Si un token válido existe, el middleware decodifica y asigna el ID del usuario para su posterior uso. Si no tiene un token, se envía una respuesta de acceso denegado.

3.2.3 Front-End

En el proceso de construcción del Front-End, se optó por emplear el lenguaje de programación JavaScript, junto con frameworks como React-Native, específicamente diseñados para el desarrollo de la interfaz de usuario en entornos móviles.

La interfaz de usuario del Front-End utiliza los componentes Nativos de React-Native y CSS para el diseño del sitio web. También se utilizó librerías como “Moongose” para el modelado de datos diseñada para MongoDB, “TouchableOpacity” para botones táctiles y Modal para ventanas modales.

3.2.4 Aprendizaje automático (Machine learning)

En el proceso de desarrollo se ha creado una solución que combina aprendizaje automático (Machine learning) y procesamiento del lenguaje natural (NLP) para la clasificación de textos en el ámbito de la psicoterapia. La primera fase implica el preprocesamiento de datos mediante bibliotecas como: “Pandas” y “NLTK”, la asignación de etiquetas numéricas, manejo de valores nulos y división del conjunto de datos en la tarea de clasificación. La segunda fase se enfoca en la aplicación de tecnologías avanzadas, entre ellas transformers y BERT, conocidas por su habilidad para comprender el lenguaje en contextos bidireccionales. Esta capacidad permite una comprensión profunda de las relaciones semánticas presentes en el discurso, en nuestro caso se utiliza para ámbito psicoterapéutico.

En nuestro enfoque de entrenamiento con BERT, hemos utilizado el modelo pre-entrenado "dccuchile/bert-base-spanish-wwm-cased". Se adapta la arquitectura de clasificación de secuencias de BERT para las tareas específicas de 'Class' y 'Type', implementando estrategias de regularización, como el dropout, con el fin de prevenir el sobreajuste del modelo en los datos de entrenamiento y mejorar su capacidad para generalizar los datos que no fueron previamente parte del entrenamiento del modelo. Asimismo, se emplea el optimizador AdamW, para evitar el sobreajuste, ya que penaliza los pesos más grandes y favorece modelos más simples y generalizables. Como información adicional, se consideró la integración de la plataforma DialogFlow en el proyecto. Sin embargo, se encontró complejidad en el desarrollo de flujos de conversación más naturales, lo cual resultó ser bastante limitante para el progreso del proyecto.

3.2.5 Códigos importantes Back-End con Python

Figura 32

Carga de los conjuntos de datos.

```
19  # Cargar el archivo CSV
20  ruta_csv = "../../diccionarios/Dataset_Typo_Picoterapia.csv"
21  df = pd.read_csv(ruta_csv, delimiter=';')
```

Nota. Código que carga el modelo de entrenamiento. Elaborado por los autores.

Según la Figura 32**,** se sube el conjunto de datos creado en formato (CSV) y lo almacena en un DataFrame de “Pandas”.

Figura 33

Mapeo de las clases tanto para clase y tipo.

```
# Crear un mapeo de clases a etiquetas numéricas para 'Class'
class_mapping_class = {label: idx for idx, label in enumerate(df['Class'].unique())}
df['Class'] = df['Class'].map(class_mapping_class)

# Crear un mapeo de clases a etiquetas numéricas para 'Type'
class_mapping_type = {label: idx for idx, label in enumerate(df['Type'].unique())}
df['Type'] = df['Type'].map(class_mapping_type)
```

Nota. Código que realiza un mapeo para la clase y el tipo. Elaborado por los autores.

En la Figura 33**,** se aprecia la realización del mapeo de las clases ('Class' y 'Type') a etiquetas numéricas, para que el modelo de aprendizaje automático pueda procesar y aprender de los datos.

Figura 34

Clase PsicoterapiaDataset. Creación de conjuntos de datos y dataloaders.

```
# Crear conjuntos de datos y dataloaders para la clasificación de clase y tipo
class PsicoterapiaDataset(Dataset):
    def __init__(self, text_list, labels_class, labels_type):
        self.text_list = text_list
        self.labels_class = labels_class
        self.labels_type = labels_type

    def __len__(self):
        return len(self.text_list)

    def __getitem__(self, idx):
        idx = idx % len(self.text_list)  # Asegurarse de que el índice sea válido
        try:
            return {
                'text': str(self.text_list[idx]),
                'labels_class': torch.tensor(self.labels_class[idx], dtype=torch.long).to(device),
                'labels_type': torch.tensor(self.labels_type[idx], dtype=torch.long).to(device)
                }
        except IndexError as e:
            print(f"Error: {e}. Index: {idx}, Length: {len(self.labels_class)}")
            raise e
```

Nota. Se crea el conjunto de datos y dataloaders con fin de clasificar la clase y tipo. Elaborado por los autores.

Según la Figura 34**,** se observa cómo se define la clase PsicoterapiaDataset para la creación de conjuntos de datos y dataloaders para la clase y tipo:

- El método __init__ de inicialización recibe tres listas: text_list que contiene los textos, labels_class que tiene las etiquetas para la clasificación de clase, y labels_type que tiene las etiquetas para la clasificación de tipo. Estas listas son almacenadas como atributos de la clase.

- El método __len__(self) devuelve la longitud del conjunto de datos, que se define como la longitud de la lista text_list.
- El método __getitem__(self, idx) se utiliza para obtener un elemento del conjunto de datos al proporcionar un índice específico, idx. Se asegura de que el índice sea válido usando el operador % (módulo). Luego, se devuelve un diccionario que contiene el texto ('text'), las etiquetas para la clasificación de clase ('labels_class'), y las etiquetas para la clasificación de tipo ('labels_type').

Figura 35

Proceso de división en conjuntos de entrenamiento y prueba.

```
# Ajustar el tamaño de test_size para garantizar que haya suficientes datos
# para crear conjuntos de entrenamiento y prueba
test_size = 0.2
if len(df) > 0:
    textos_train, textos_test, labels_train, labels_test = train_test_split(
        df['Texto'].tolist(), df[['Class', 'Type']].values, test_size=test_size, random_state=42)
else:
    print("No data remaining after removing NaN values.")
```

Nota. División en conjuntos de entrenamiento y prueba. Elaborado por los autores.

Según la Figura 35**,** se realiza el proceso de división en conjuntos de entrenamiento y prueba. La división de datos se realiza a través de: 'train_test_split' de 'scikit-learn'.

Figura 36

Creación del conjunto de datos y dataloaders para la clasificación de tipo.

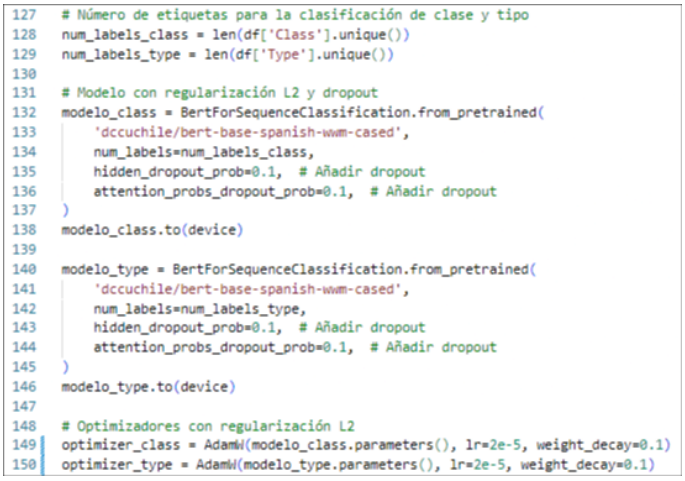
```
# Crear conjuntos de datos y dataloaders para la clasificación de tipo
train_dataset_class = PsicoterapiaDataset(textos_train, labels_train[:, 0], labels_train[:, 1])  # Columna 'Class' y 'Type'
test_dataset_class = PsicoterapiaDataset(textos_test, labels_test[:, 0], labels_test[:, 1])  # Columna 'Class' y 'Type'

train_dataloader_class = DataLoader(train_dataset_class, batch_size=16, shuffle=True)
test_dataloader_class = DataLoader(test_dataset_class, batch_size=16, shuffle=False)
```

Nota. El conjunto de datos puede clasificar el tipo. Elaborado por los autores.

Según la Figura 36**,** se generan los conjuntos de datos ('train_dataset_class' y 'test_dataset_class') y dataloaders ('train_dataloader_class' y 'test_dataloader_class') para la tarea de clasificación de clase. Estos dataloaders cargan datos en lotes de 16, y se mezclan los datos de entrenamiento con el propósito de mejorar la capacidad de generalización del modelo. La clase "PsicoterapiaDataset" personalizada maneja la estructura de los datos, tomando textos y etiquetas correspondientes.

Figura 37

Etiquetas para la clasificación del tipo y clase.



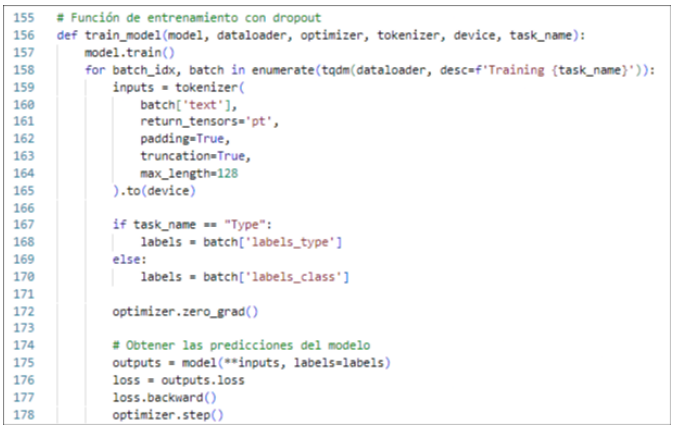
```
# Número de etiquetas para la clasificación de clase y tipo
num_labels_class = len(df['Class'].unique())
num_labels_type = len(df['Type'].unique())

# Modelo con regularización L2 y dropout
modelo_class = BertForSequenceClassification.from_pretrained(
    'dccuchile/bert-base-spanish-wwm-cased',
    num_labels=num_labels_class,
    hidden_dropout_prob=0.1,  # Añadir dropout
    attention_probs_dropout_prob=0.1,  # Añadir dropout
)
modelo_class.to(device)

modelo_type = BertForSequenceClassification.from_pretrained(
    'dccuchile/bert-base-spanish-wwm-cased',
    num_labels=num_labels_type,
    hidden_dropout_prob=0.1,  # Añadir dropout
    attention_probs_dropout_prob=0.1,  # Añadir dropout
)
modelo_type.to(device)

# Optimizadores con regularización L2
optimizer_class = AdamW(modelo_class.parameters(), lr=2e-5, weight_decay=0.1)
optimizer_type = AdamW(modelo_type.parameters(), lr=2e-5, weight_decay=0.1)
```

Nota. Se define el tipo y la clase para la clasificación. Elaborado por los autores.

En Figura 37 se observa cómo se define y configura la arquitectura del modelo de clasificación utilizando BERT para el idioma español. Se crean modelos separados para la clasificación de clase (modelo_class) y tipo (modelo_type). Se utiliza la técnica de dropout para regularización y se asignan los modelos al dispositivo de ejecución (GPU o CPU). Además, se trabaja con optimizadores AdamW valores específicos de aprendizaje y regularización de peso para ajustar los pesos durante el entrenamiento.

Figura 38

Función de entrenamiento.



```
# Función de entrenamiento con dropout
def train_model(model, dataloader, optimizer, tokenizer, device, task_name):
    model.train()
    for batch_idx, batch in enumerate(tqdm(dataloader, desc=f'Training {task_name}')):
        inputs = tokenizer(
            batch['text'],
            return_tensors='pt',
            padding=True,
            truncation=True,
            max_length=128
        ).to(device)

        if task_name == "Type":
            labels = batch['labels_type']
        else:
            labels = batch['labels_class']

        optimizer.zero_grad()

        # Obtener las predicciones del modelo
        outputs = model(**inputs, labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
```

Nota. Se realiza el entrenamiento del modelo. Elaborado por los autores.

En la Figura 38 se observa la función de entrenamiento (‘train_model’) que realiza el siguiente proceso:

- Inicia el modelo en el modo de entrenamiento.
- Itera sobre los lotes de datos de entrenamiento.
- Procesa los lotes utilizando el modelo y calcula la pérdida.
- Realiza la retro propagación y actualiza los pesos del modelo.
- Utiliza un optimizador para ajustar los parámetros del modelo.
- Repite estos pasos para cada época de entrenamiento.

Figura 39

Evaluación del rendimiento del modelo.

```python
# Función de evaluación para la clasificación de clase
def evaluate_model_class(model, dataloader, tokenizer, device):
    model.eval()
    predictions = []
    all_labels = []

    with torch.no_grad():
        try:
            for batch in tqdm(dataloader, desc='Evaluating Class'):
                inputs = tokenizer(
                    batch['text'],
                    return_tensors='pt',
                    padding=True,
                    truncation=True,
                    max_length=128
                ).to(device)
                outputs = model(**inputs)
                logits_class = outputs.logits
                predictions.extend(torch.argmax(logits_class, dim=1).cpu().numpy())
                all_labels.extend(batch['labels_class'].cpu().numpy())

            accuracy = accuracy_score(all_labels, predictions)
            print(f'Accuracy for Class: {accuracy * 100:.2f}%')
            return accuracy

        except Exception as e:
            print(f"Error during evaluation: {e}")
            return 0.0
```

Nota. Se evalúa el rendimiento del modelo. Elaborado por los autores.

En la Figura 39 observa la función de evaluación (‘evaluate_model_class’ y ‘evaluate_model_type’) que se encarga de evaluar el rendimiento del modelo de clasificación BERT después de cada época de entrenamiento a través siguiente proceso:

- Inicia el modelo en modo de evaluación.
- Iteran sobre lotes de datos de prueba.
- Procesan los lotes y obtienen las predicciones del modelo.

- Se calculan la precisión del modelo comparando las predicciones con las etiquetas reales.
- Se imprimen y devuelven la precisión del modelo para la clasificación de clase o tipo.

Figura 40

Guardar el modelo de clasificación de la clase.

```
# Guardar el modelo después de completar todas las épocas
ruta_guardado_class = "../../diccionarios/Modelo_App_Class"
modelo_class.save_pretrained(ruta_guardado_class)
```

Nota. Se guarda el modelo de la clasificación para la clase. Elaborado por los autores.

En la Figura 40 se observa cómo se guarda el modelo de clasificación de clase (‘modelo_class’ o ‘modelo_Type’) después de completar todas las épocas de entrenamiento. El modelo se guarda en un directorio especifico lo que permite utilizar el modelo entrenado en futuras predicciones sin necesidad de volver a entrenarlo.

3.2.6 Códigos importantes Back-End en el servidor Flask

Figura 41

Función para el preprocesamiento de texto.

```
# Función para preprocesar texto
def preprocess_text(text):
    if isinstance(text, str):
        text = text.lower()
        text = re.sub(r'[^a-zA-Z\s/áéíóúüñ]', '', text)

        tokenizer_local = BertTokenizer.from_pretrained('dccuchile/bert-base-spanish-wwm-cased')

        ventana_deslizante = tokenizer_local.model_max_length - 2
        segmentos = [text[i:i + ventana_deslizante] for i in range(0, len(text), ventana_deslizante // 2)]

        tokens = []

        for segmento in segmentos:
            segmento_tokens = [stemmer.stem(token) for token in word_tokenize(segmento) if token not in stop_words]
            tokens.extend(segmento_tokens)

        return ' '.join(tokens)
    else:
        return ''

```

Nota. Se realiza el pre procesamiento del texto. Elaborado por los autores.

En la Figura 41 se observa la función “preprocess_text” que realiza el preprocesamiento de texto, ajustes y limpiezas al texto de entrada, siguiendo el proceso detallado a continuación:

- Toma un texto como entrada.
- Convierte el texto a minúsculas.

- Elimina caracteres especiales, dejando solo letras y espacios.
- Divide el texto en segmentos utilizando un tokenizador BERT específico para el español con la técnica de ventana deslizante.
- Lematiza cada segmento, eliminando stop words y generando una lista de tokens.
- Concatena los tokens de todos los segmentos.
- Finalmente devuelve el texto pre procesado.

Figura 42

Función para la predicción de la clase y tipo.

```
# Función para predecir clase y tipo
def predict_class_and_type(text):
    processed_text = preprocess_text(text)
    encoded_text = tokenizer(processed_text, padding=True, truncation=True, return_tensors='pt', max_length=512)

    with torch.no_grad():
        outputs_class = modelo_class(**encoded_text)

    with torch.no_grad():
        outputs_type = modelo_type(**encoded_text)

    predicted_class_idx = torch.argmax(outputs_class.logits[0]).item()
    predicted_class_label = class_mapping_inv[predicted_class_idx]

    predicted_type_idx = torch.argmax(outputs_type.logits[0]).item()
    predicted_type_label = class_mapping_type_inv[predicted_type_idx]

    return predicted_class_label, predicted_type_label
```

Nota. Realiza la predicción para la clase y el tipo. Elaborado por los autores.

En la Figura 42 se observa la función 'predict_class_and_type' que recibe como parámetro un texto, realiza el preprocesamiento del texto, lo codifica y utiliza dos modelos pre-entrenados ('modelo_class' y 'modelo_type') para predecir la clase y el tipo relacionados al texto ingresado. Luego, mapea los índices predichos a etiquetas mediante mapeos inversos y devuelve las etiquetas de clase y tipo predichas.

Figura 43

Función para generar respuesta usando OpenAI.

```
# Función para generar respuesta usando OpenAI ChatGPT
def generate_openai_response(prompt):
    try:
        chat_completion = clientGPT.chat.completions.create(
            model="gpt-3.5-turbo",
            messages=[
                {"role": "system", "content": "Eres un psicoterapeuta que brinda apoyo emocional."},
                {"role": "user", "content": prompt}
            ]
        )
        return chat_completion.choices[0].message.content
    except Exception as e:
        print(f"Error al conectarse a OpenAI: {e}")
        return f"Lo siento, hay un problema al intentar responder en este momento. Detalles: {str(e)}"
```

Nota. Obtener respuesta por parte de OpenAI. Elaborado por los autores.

Según la Figura 43, se observa la función que utiliza el servicio OpenAI para generar respuestas en un contexto de psicoterapeuta. La función toma un prompt (Pregunta, declaración o cualquier texto) del usuario, establece el rol del sistema y del usuario, y se envía una solicitud de generación de texto al modelo “GPT-3.5 Turbo” para obtener una respuesta del modelo mencionado.

CAPÍTULO IV: RESULTADOS

4.1 Resultado final de la aplicación móvil

Implementación del prototipo de aplicación móvil conformado por los módulos de bienvenida, registro, inicio de sesión, inicio, conversaciones, recomendaciones, mi cuenta y chatbot. A continuación, se detalla los resultados de la implementación de cada módulo.

Tabla 20

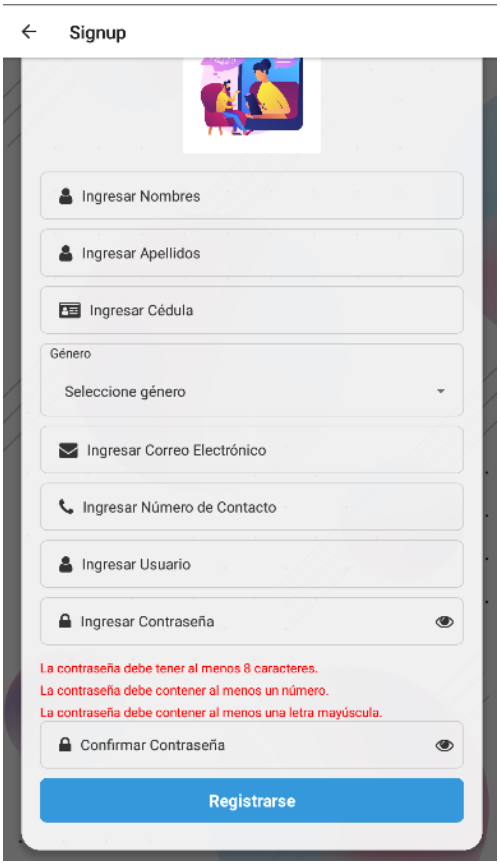
Implementación del módulo de bienvenida.

Implementación del módulo de bienvenida	
	En la ilustración se muestra la implementación del módulo de bienvenida utilizando React-Native y ejecutado en el emulador NoxPayer, el módulo consta de los siguientes componentes: • Mensaje de bienvenida. • Botón interactivo para acceder al registro. • Botón interactivo para acceder al inicio de sesión.

Nota. Se muestra el resultado del módulo de bienvenida. Elaborado por los autores.

Tabla 21

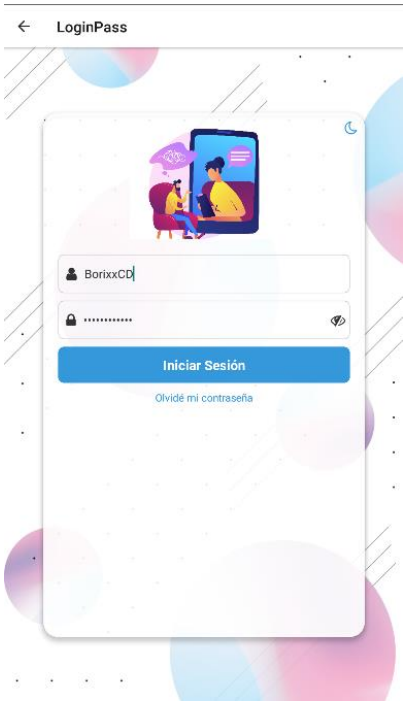
Implementación del módulo de registro.

Implementación del módulo de registro	
	En la ilustración se muestra la implementación del módulo de registro utilizando React-Native y ejecutado en el emulador NoxPayer, el módulo consta de los siguientes componentes y validaciones: • Formulario para el ingreso de datos. • Validación de correo. • Validaciones para campos de texto y numéricos. • Validación de cedulas ya registradas en la base de datos. • Validación de número de contacto ya registrado en la base de datos. • Validación de confirmación de contraseña.

Nota. Se muestra el resultado del módulo de registro. Elaborado por los autores.

Tabla 22

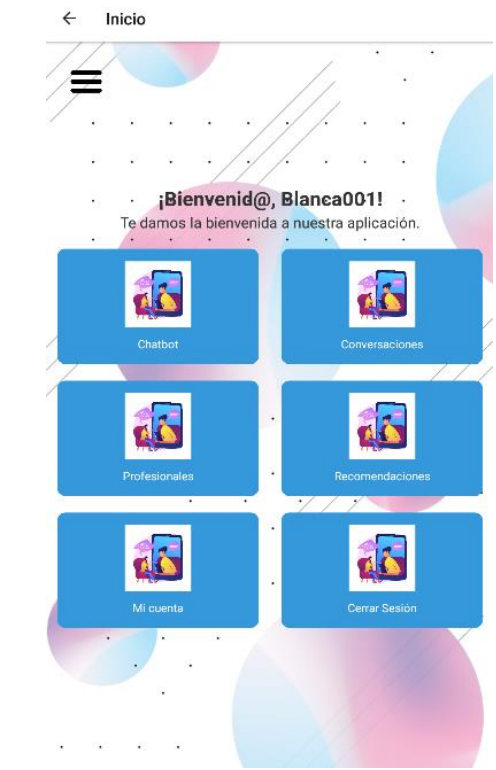
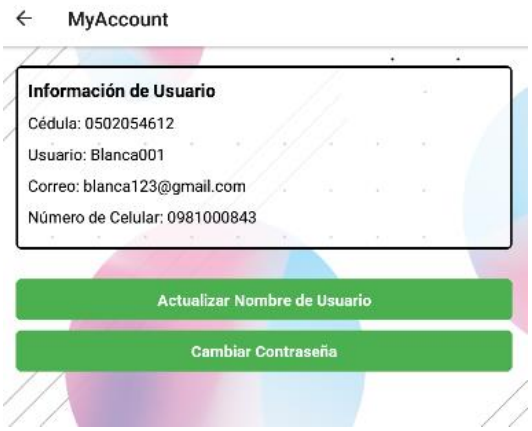
Implementación del módulo de inicio de sesión.

Implementación del módulo de inicio de sesión	
	En la ilustración se muestra la implementación del módulo de inicio de sesión utilizando React-Native y ejecutado en el emulador NoxPayer, el módulo consta de los siguientes componentes y validaciones: • Formulario de ingreso de datos. • Validaciones usuario. • Validaciones de contraseña. • Botón interactivo de inicio de sesión. • Botón interactivo de cambio de contraseña.
	En la ilustración se muestra la implantación de una ventana modal para el cambio de contraseña que consta de los siguientes componentes y validaciones: • Validación de usuario. • Validación de cedula. • Validaciones para campos de texto y numéricos. • Validación de confirmación de contraseña.

Nota. Se muestra el resultado del módulo de inicio de sesión. Elaborado por los autores.

Tabla 23

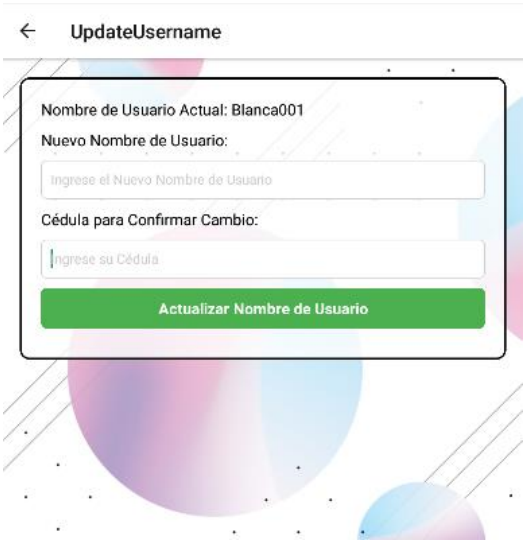
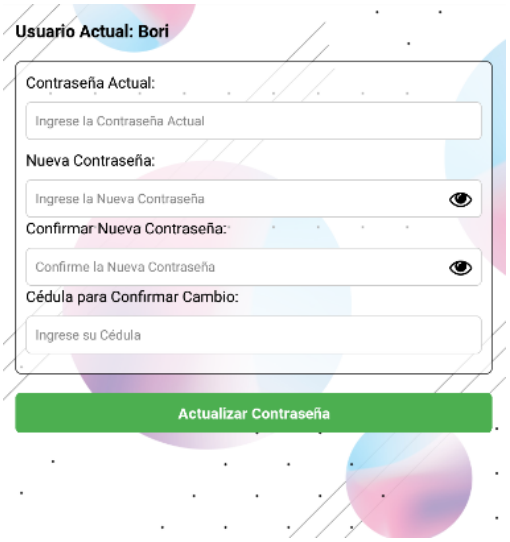
Implementación del módulo de inicio.

Implementación del módulo de inicio
   
En la ilustración se muestra la implementación del módulo de inicio de sesión utilizando React-Native y ejecutado en el emulador NoxPayer, el módulo consta de los siguientes componentes: • Botón interactivo de chatbot • Botón interactivo de conversaciones. • Botón interactivo de recomendaciones. • Botón interactivo de mi cuenta • Botón interactivo de cierre de sesión. En la ilustración se muestra la implantación modulo mi cuenta para visualizar la información personal del usuario que consta de los siguientes componentes: • Datos de usuario. • Botón interactivo para actualizar nombre de usuario. • Botón interactivo para actualizar contraseña.

Nota. Se muestra el resultado del módulo de inicio. Elaborado por los autores.

Tabla 24

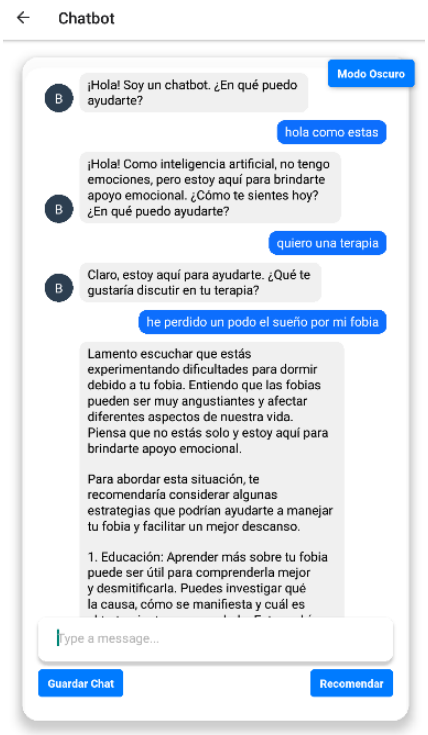
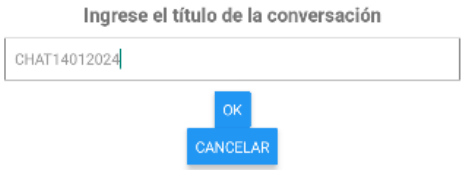
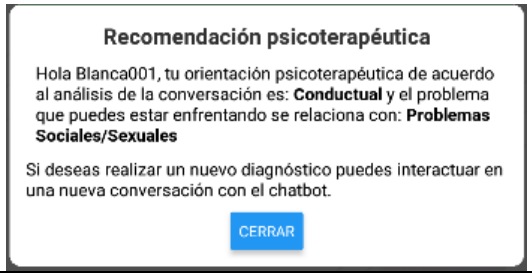
Implementación del módulo para actualizar usuario y contraseña asociados al módulo mi cuenta.

Implementación del módulo para actualizar usuario y contraseña asociados al módulo mi cuenta	
	En la ilustración se muestra la implantación modulo actualizar nombre de usuario que consta de los siguientes componentes y validaciones: • Formulario para ingreso de datos. • Validación de existencia de usuario. • Validación de cedula para realizar el cambio. • Botón interactivo para actualizar nombre de usuario.
	En la ilustración se muestra la implantación modulo actualizar contraseña que consta de los siguientes componentes y validaciones: • Formulario para ingreso de datos. • Validación para confirmación de contraseña. • Validación cedula para confirmar los cambios. • Botón interactivo para actualizar contraseña.

Nota. Se muestra el resultado del módulo de actualizar contraseña y usuario. Elaborado por los autores.

Tabla 25

Implementación del módulo de chatbot.

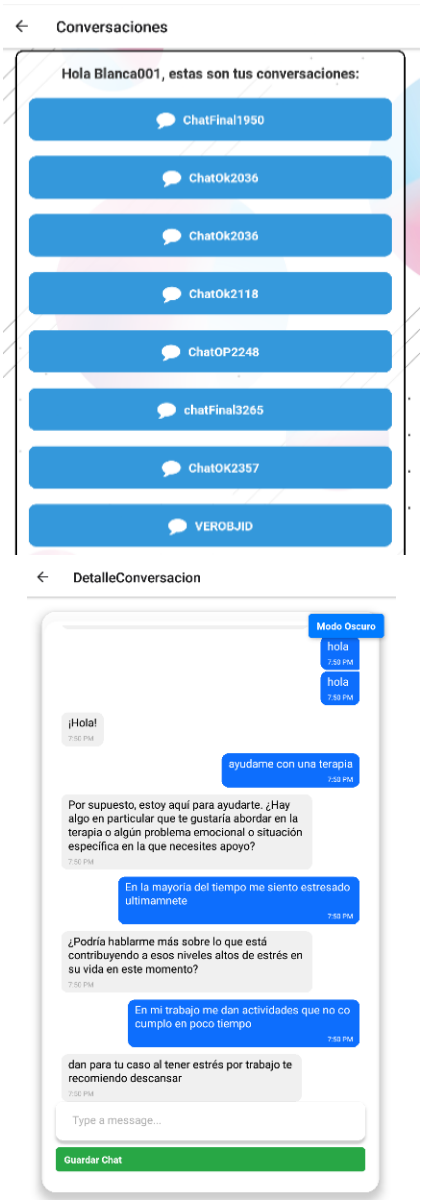
Implementación del módulo de chatbot	
	En la ilustración se muestra la implementación del módulo de chatbot utilizando React-Native y ejecutado en el emulador NoxPayer, el módulo consta de los siguientes componentes: • Interfaz para interactuar con el chatbot que consume el servicio de OpenAI. • Botón interactivo para guardar la conversación • Botón interactivo para obtener y guardar la recomendación psicoterapéutica.
	En la ilustración se muestra la implantación de una ventana modal para guardar la conversación, componentes: • Caja de texto para el nombre de la conversación. • Botones interactivos para guardar y cerrar ventana modal.
	En la ilustración se muestra la implantación de una ventana modal para visualizar la recomendación psicoterapéutica, componentes:

- Mensaje de recomendación con la clase y el tipo de orientación psicoterapéutica
- Botón interactivo para cerrar la ventana modal.

Nota. Se muestra el resultado del módulo de chatbot. Elaborado por los autores.

Tabla 26

Implementación del módulo de conversaciones.

Implementación del módulo de conversaciones	
	En la ilustración se muestra la implementación del módulo de conversaciones utilizando React-Native y ejecutado en el emulador NoxPayer, el módulo consta de los siguientes componentes: • Botones interactivos con el nombre de la conversación guardad en base de datos. En la ilustración se muestra la implantación módulo de historial de conversaciones para visualizar la conversación almacenada en la base de datos, el módulo consta de los siguientes componentes: • Ventana de chatbot para visualizar conversación. • Caja de texto para interactuar seguir interactuando con el chatbot.

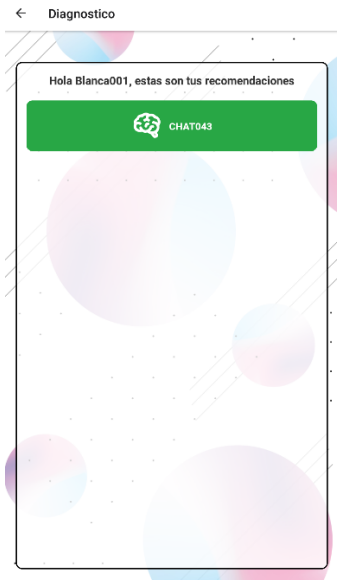
- Botón interactivo para guardar la conversación.

Nota. Se muestra el resultado del módulo de conversaciones. Elaborado por los autores.

Tabla 27

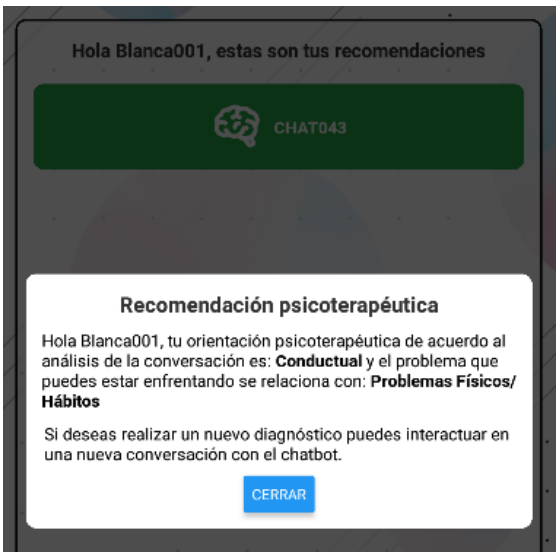
Implementación del módulo de recomendaciones.

Implementación del módulo de recomendaciones



En la ilustración se muestra la implementación del módulo de recomendaciones utilizando React-Native y ejecutado en el emulador NoxPayer, el módulo consta de los siguientes componentes:

- Botones interactivos con el nombre de la conversación para desplegar la recomendación almacenada en la base de datos.



En la ilustración se muestra la implantación de una ventana modal para visualizar la recomendación psicoterapéutica almacenada en la base de datos después de analizar una conversación, la ventana modal consta de los siguientes componentes:

- Mensaje de recomendación con la clase y el tipo de orientación psicoterapéutica
- Botón interactivo para cerrar la ventana modal.

Nota. Se muestra el resultado del módulo de recomendaciones. Elaborado por los autores.

4.2 Pruebas funcionales

Tabla 28

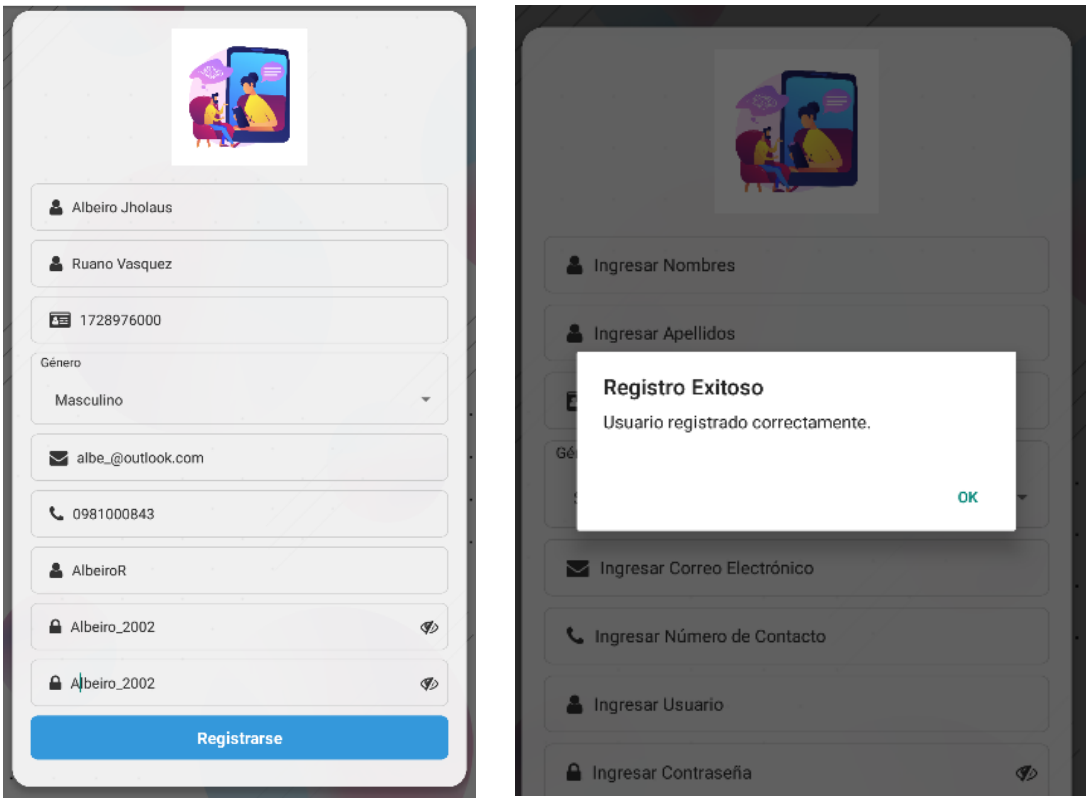
Prueba funcional 1.

Prueba funcional No.1	**RF01**	**12/01/2024**	**Responsable: Albeiro Ruano**

Registro de usuario: Los usuarios tendrán la capacidad de ingresar información personal, incluyendo nombre, apellidos, número de cédula, género, dirección de correo electrónico, número de contacto, nombre de usuario y contraseña. Además, se aplicarán diversas validaciones, como verificar la longitud y composición de la contraseña, así como comprobar si la cédula, correo electrónico, número de contacto y nombre de usuario ya están registrados en el sistema. Este proceso asegurará que la información proporcionada sea precisa y única. Como resultado de este procedimiento de registro, los usuarios podrán acceder a la aplicación utilizando tanto su nombre de usuario como su contraseña.

Proceso y resultado: El usuario accede a la aplicación móvil se dirige a la ventana de registro e ingresa sus datos, selecciona el botón registrar y posterior se despliega la ventana de registro exitoso.

RF01 aprobado



Nota. Se muestra el resultado de la primera prueba funcional. Elaborado por los autores.

Tabla 29

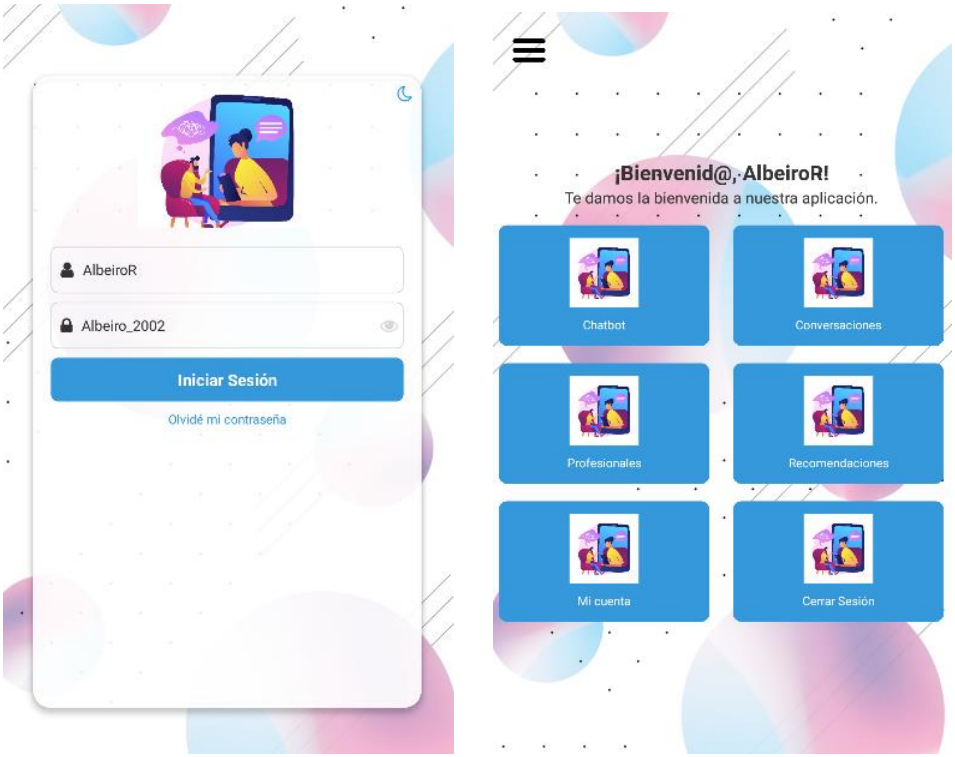
Prueba funcional 2.

Prueba funcional No.2	**RF02**	**12/01/2024**	**Responsable: Albeiro Ruano**

Inicio de sesión: Este proceso permitirá a los usuarios autenticarse mediante credenciales previamente establecidas durante el proceso de registro, para ello se requiere el nombre de usuario y contraseña.

Proceso y resultado: El usuario accede a la aplicación móvil se dirige a la ventana de inicio de sesión e ingresa sus datos, secciona el botón iniciar sesión y accede a la ventana de inicio.

RF02 aprobado



Nota. Se muestra el resultado de la segunda prueba funcional. Elaborado por los autores.

Tabla 30

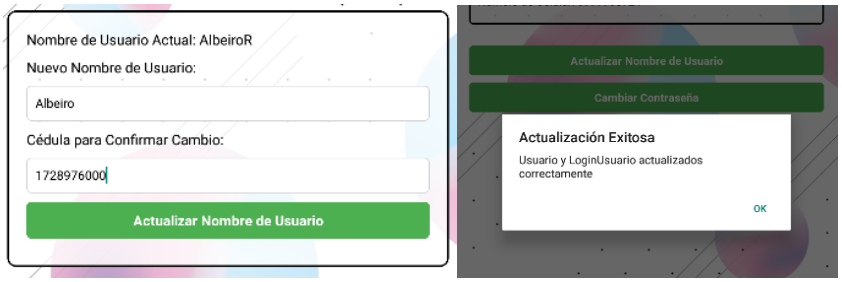
Prueba funcional 3.

Prueba funcional No.3	**RF03**	**12/01/2024**	**Responsable: Albeiro Ruano**

Cambiar nombre de usuario: En este proceso permite cambiar el nombre de usuario utilizando la cedula como validador para el cambio.

Proceso y resultado: El usuario accede a la aplicación móvil, se dirige a la sección mi cuenta, selecciona actualizar nombre de usuario, accede al formulario, ingresa el nuevo nombre de usuario, la cedula para validar y realizar el cambio. Finalmente obtiene el mensaje indicando que el cambio fue exitoso.

RF03 aprobado



Nota. Se muestra el resultado de la tercera prueba funcional. Elaborado por los autores.

Tabla 31

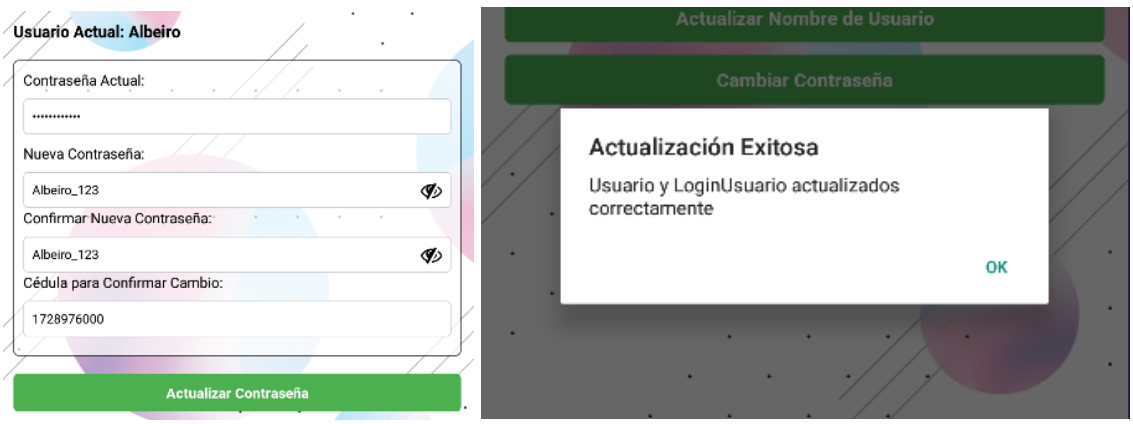
Prueba funcional 4.

Prueba funcional No.4	**RF04**	**12/01/2024**	**Responsable: Albeiro Ruano**

Cambiar contraseña: En este proceso el usuario podrá cambiar la contraseña mediante el uso de la contraseña actual y la cedula se utilizará como validador para el cambio.

Proceso y resultado: El usuario accede a la aplicación móvil, se dirige a la sección mi cuenta, selecciona actualizar contraseña, accede al formulario, ingresa contraseña actual, la nueva contraseña y la cedula para validar y realizar el cambio. Finalmente obtiene el mensaje indicando que el cambio fue exitoso.

RF04 aprobado



Nota. Se muestra el resultado de la cuarta prueba funcional. Elaborado por los autores.

Tabla 32

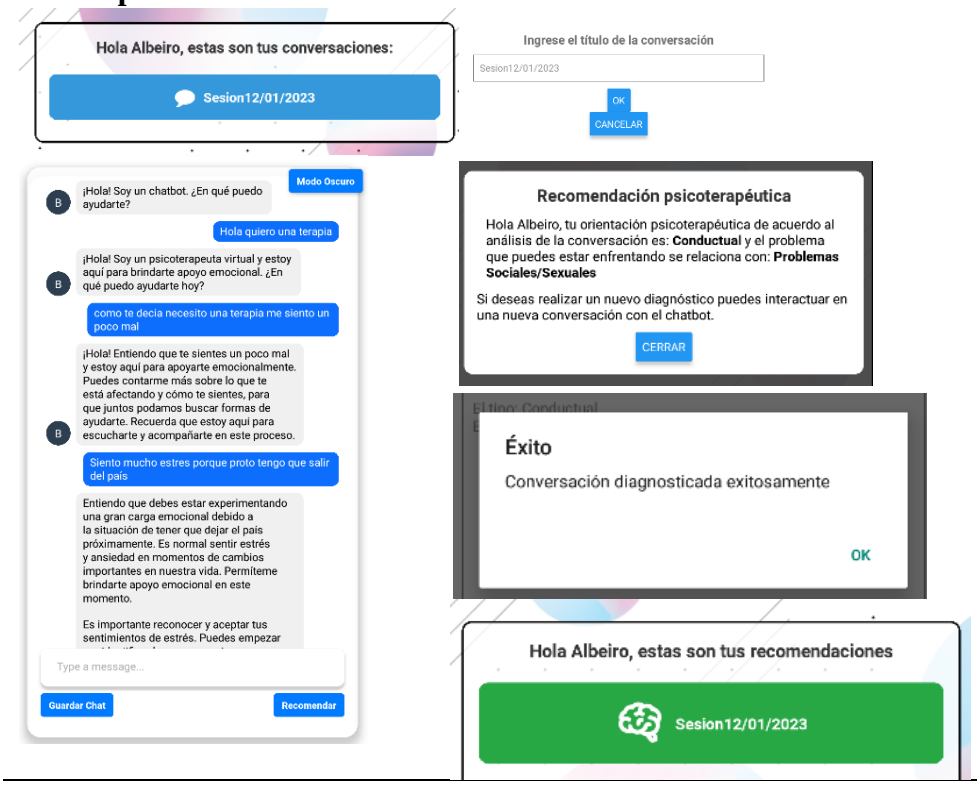
Prueba funcional 5.

Prueba funcional No.5	**RF05**	**12/01/2024**	**Responsable: Albeiro Ruano**

Chatbot: Este proceso se basa en realizar preguntas relevantes sobre la situación emocional o mental del usuario, sus experiencias, síntomas, y otros factores relevantes. Basándose en las respuestas proporcionadas, el chatbot utilizaría algoritmos y reglas predefinidas para sugerir una orientación psicoterapéutica específica, como terapia cognitivo-conductual, psicoanálisis, terapia de pareja, entre otras.

Proceso y resultado: El usuario accede a la aplicación móvil se dirige a la sección de chatbot, accede e interactúa con el chatbot, guarda la conversación, y selecciona recomendar lo que devolverá la orientación psicoterapéutica más adecuada, en base al análisis de la conversación guardada.

RF05 aprobado



Nota. Se muestra el resultado de la quinta prueba funcional. Elaborado por los autores.

4.3 Fase de aceptación

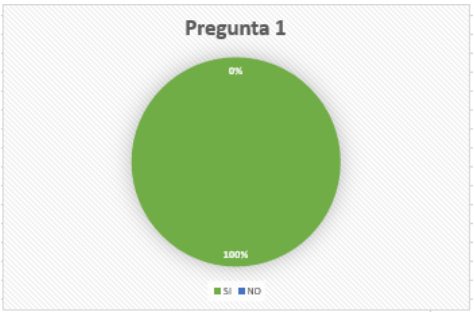
En la fase actual para conocer el nivel de aceptación del prototipo de aplicación móvil para recomendar la orientación psicoterapéutica se ha realizado una encuesta a diez diferentes

usuarios después de haber utilizado la aplicación móvil, A continuación, se detalla las preguntas y los resultados obtenidos.

1 ¿Te sentiste respaldado/a y comprendido/a durante las interacciones con el chatbot?

Figura 44

Pregunta 1. Prueba de aceptación.



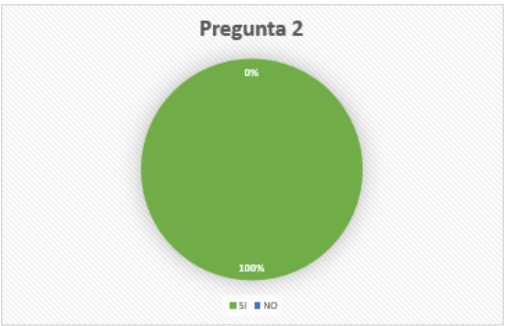
Nota. Porcentaje de participantes en relación con la pregunta 1. Se asigna el color verde a "Sí" y el color azul a "No". Elaborado por los autores.

En esta primera pregunta, todos los usuarios entendieron cada una de las interacciones realizadas por el chatbot. No hubo comentarios referentes a la pregunta por parte de los encuestados.

2 ¿Recomendarías esta aplicación a otros usuarios que buscan orientación psicoterapéutica?

Figura 45

Pregunta 2. Prueba de aceptación.



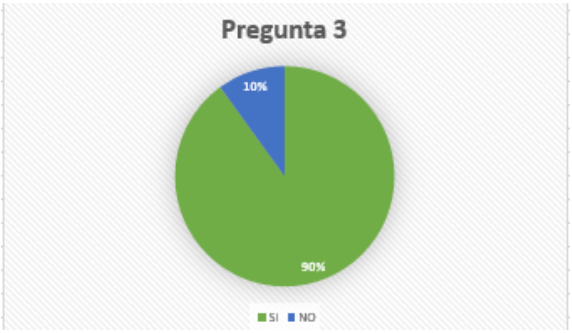
Nota. Porcentaje de participantes en relación con la pregunta 2. Se asigna el color verde a "Sí" y el color azul a "No". Elaborado por los autores.

Al igual que en la pregunta anterior, todos los usuarios respondieron afirmativamente cuando se les preguntó si recomendarían este aplicativo a otras personas o familiares. Como observación, señalaron que preferirían una mejora en el aspecto visual de la aplicación, ya que indicaban que se requieren algunas mejoras en la aplicación.

3 ¿La aplicación proporcionó información clara y comprensible de la orientación psicoterapéutica?

Figura 46

Pregunta 3. Prueba de aceptación.



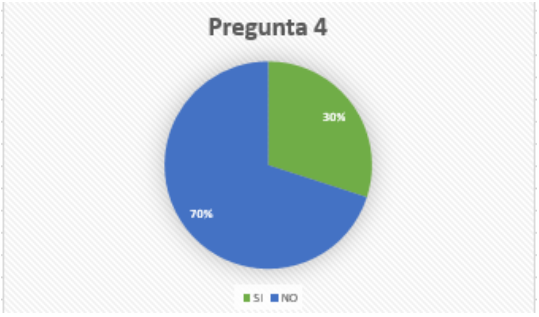
Nota. Porcentaje de participantes en relación con la pregunta 3. Se asigna el color verde a "Sí" y el color azul a "No". Elaborado por los autores.

En esta pregunta, la mayoría de los usuarios que interactuaron con la aplicación, comprendieron el resultado proporcionado a excepción de un porcentaje mínimo.

4 ¿Hubo algún problema técnico o dificultad que encontraste durante el uso de la aplicación?

Figura 47

Pregunta 4. Prueba de aceptación.



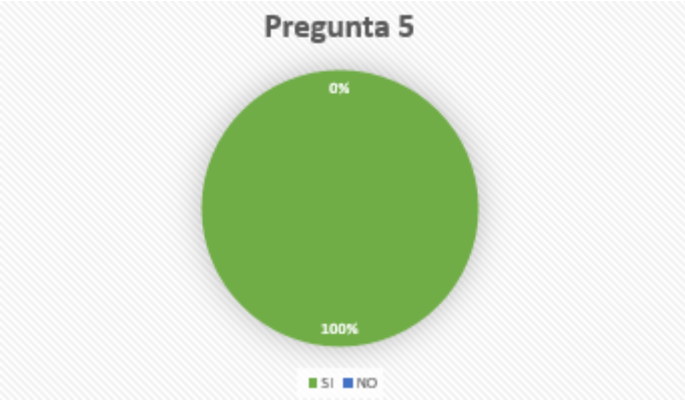
Nota. Porcentaje de participantes en relación con la pregunta 4. Se asigna el color verde a "Sí" y el color azul a "No". Elaborado por los autores.

En esta pregunta, muchos usuarios no tuvieron dificultades con la aplicación. Un porcentaje experimentó demoras en la visualización de algunas ventanas como: inicio de sesión y registro. Los usuarios recomendaron mejorar los tiempos de respuesta en las ventanas mencionadas y la mejora de interfaz de usuario.

5 ¿Encuentras útiles las funciones de recomendación proporcionadas por la aplicación?

Figura 48

Pregunta 5. Prueba de aceptación.



Nota. Porcentaje de participantes en relación con la pregunta 5. Se asigna el color verde a "Sí" y el color azul a "No". Elaborado por los autores.

En la última pregunta, los usuarios coincidieron en que las funciones de la aplicación móvil resultan útiles. Incluso expresaron que la recomendación proporcionada por la aplicación podría ser de utilidad al momento de discutir aspectos relevantes con su psicoterapeuta de confianza.

Tabla 33

Resumen de pruebas de aceptación.

Pregunta	Porcentaje de aceptación	Porcentaje de no aceptación
1 ¿Te sentiste respaldado/a y comprendido/a durante las interacciones con el chatbot?	100%	0%
2 ¿Recomendarías esta aplicación a otros usuarios que buscan orientación psicoterapéutica?	100%	0%
3 ¿La aplicación proporcionó información clara y comprensible de la orientación psicoterapéutica?	90%	10%
4 ¿Hubo algún problema técnico o dificultad que encontraste durante el uso de la aplicación?	70%	30%
5 ¿Encuentras útiles las funciones de recomendación proporcionadas por la aplicación?	100%	0%

Nota. Se muestra un resumen general del porcentaje de aceptación de las preguntas realizadas a los usuarios después de utilizar la aplicación móvil. Elaborado por los autores.

En la Tabla 33 se evidencia la aceptación de la aplicación móvil posterior a la interacción con los usuarios y la evaluación de la misma a través de una encuesta de aceptación.

4.4 Pruebas estructurales

Para la sección siguiente, se llevaron a cabo seis pruebas estructurales o de caja blanca, enfocadas en aspectos como el registro del usuario, la obtención de recomendaciones, la actualización del perfil del usuario y el proceso de inicio de sesión. Estas pruebas fueron realizadas en Postman empleando los métodos HTTP: “GET” y POST”.

Tabla 34

Prueba estructural 1.

ID de prueba	Descripción de la prueba	Método/Endpoint
TCB001	Registro de usuario exitoso	POST /Usuarios
Pasos de ejecución:		

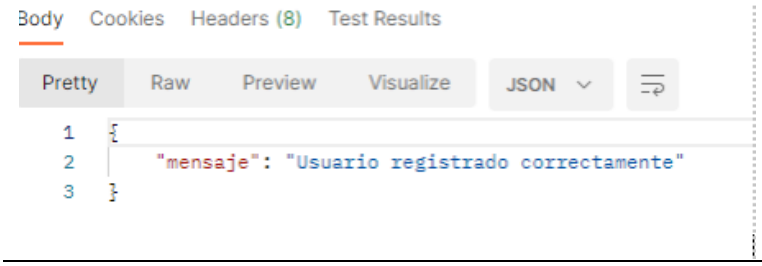
1.Enviar una solicitud POST a “/Usuarios” con datos válidos.

2.Verificar el código de estado 200. 3. Verificar el mensaje de éxito en la respuesta.

Datos de entrada: Formato JSON.

```
{
  "nombres": "John",
  "apellidos": "Doe",
  "cedula": "123456789",
  "genero": "Masculino",
  "correo": "john.@example.com",
  "numeroContacto": "1234567890",
  "usuario": "john_doe",
  "contrasenia": "password123"
}
```

Resultado esperado: {"mensaje": "Usuario registrado correctamente"}



Nota. Se muestra el resultado de la prueba estructural 1. Elaborado por los autores.

Tabla 35

Prueba estructural 2.

ID de prueba	Descripción de la prueba	Método/Endpoint
TCB002	Validación de existencia de cédula	GET /existe-cedula/:cedula

Pasos de ejecución:

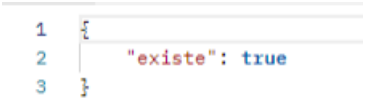
1.Enviar una solicitud GET a “/existe-cedula/:cedula” con una cédula existente.

2.Verificar el código de estado 200.

3.Verificar que el campo "existe" en la respuesta sea true.

Datos de entrada: API.

Resultado esperado: {"existe": true}



Nota. Se muestra el resultado de la prueba estructural 2. Elaborado por los autores.

Tabla 36

Prueba estructural 3.

ID de prueba	Descripción de la prueba	Método/Endpoint
TCB003	Inicio de sesión exitoso	POST /login

Pasos de ejecución:

1.Enviar una solicitud POST a "/login" con datos válidos de usuario y contraseña.

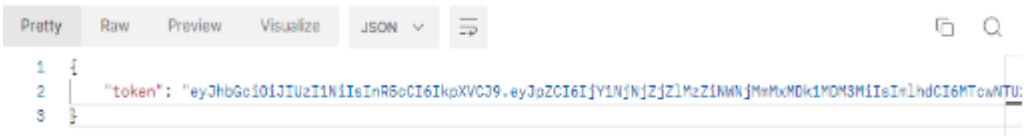
2.Verificar el código de estado 200 en la respuesta.

3.Verificar que la respuesta contenga un token de autenticación.

Datos de entrada: Formato JSON.

```
{
  "usuario": "Blanca001",
  "contrasenia": "Blanca001"
}
```

Resultado esperado: {"token": " CJ9.eyJpZCI6IjY1NjNjZjZlMz "}



Nota. Se muestra el resultado de la prueba estructural 3. Elaborado por los autores.

Tabla 37

Prueba estructural 4.

ID de prueba	Descripción de la prueba	Método/Endpoint
TCB004	Actualización de usuario exitosa	PUT /cedula_valida

Pasos de ejecución:

1.Enviar una solicitud PUT a "/Usuarios/:cedula" con una cédula válida y nuevos datos de usuario.

2.Verificar el código de estado 200 en la respuesta.

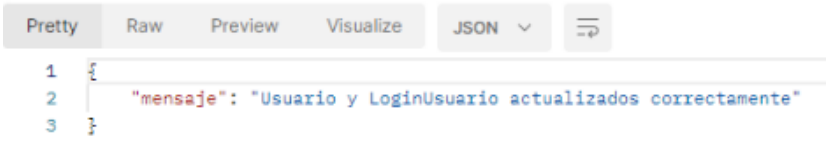
3.Verificar que la respuesta contenga el mensaje "Usuario y LoginUsuario actualizados correctamente".

Datos de entrada: Formato JSON**.**

```
{
  "correo": "Prueba.1@example.com",
  "numeroContacto": "0968404562",
```

```
  "usuario": "borisds",
  "contraseniaActual": "Blanca001",
  "nuevaContrasenia": "Clave1234",
  "confirmarContrasenia": "Clave1234"
}
```


Resultado esperado: {"mensaje": "Usuario y LoginUsuario actualizados correctamente"}



Nota. Se muestra el resultado de la prueba estructural 4. Elaborado por los autores.

Tabla 38

Prueba estructural 5

ID de prueba	Descripción de la prueba	Método/Endpoint
TCB005	Obtener conversaciones	GET / conversaciones

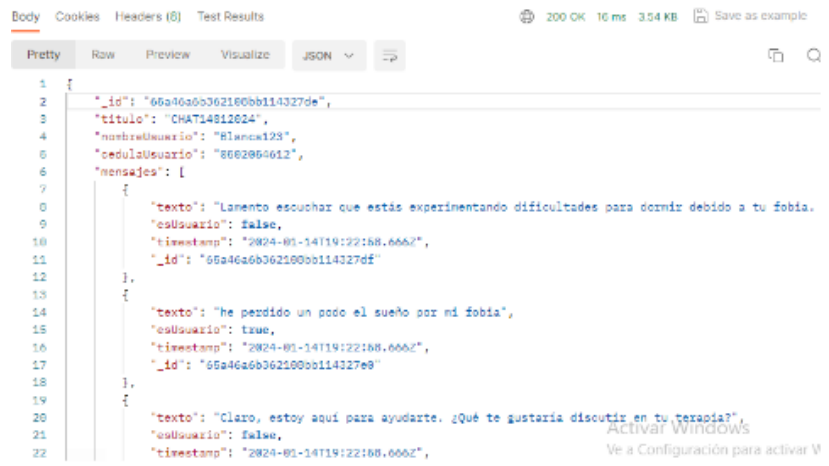
Pasos de ejecución:

1.Enviar una solicitud GET a /conversaciones.

2.Verificar que la respuesta contenga la información de las conversaciones del usuario.

Datos de entrada: API.

Resultado esperado: Formato JSON conversación de usuario.



Nota. Se muestra el resultado de la prueba estructural 5. Elaborado por los autores.

Tabla 39

Prueba estructural 6.

ID de prueba	Descripción de la prueba	Método/Endpoint
TCB006	Obtener recomendaciones	GET /diagnostico

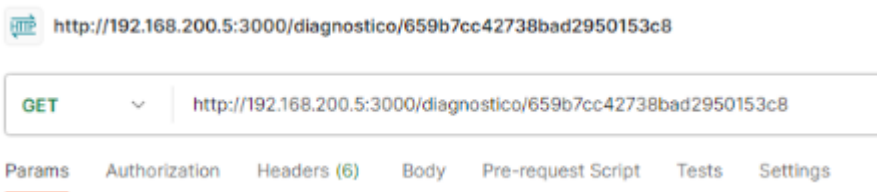
Pasos de ejecución:

1.Enviar una solicitud GET a "/diagnostico/:diagnosticoId".

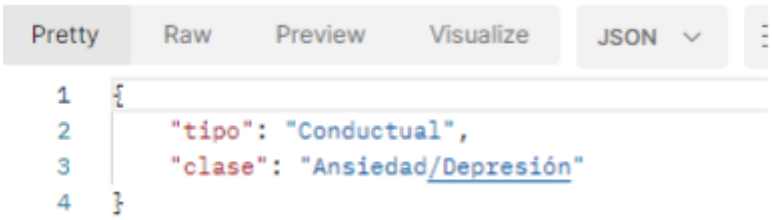
2.Reemplazar 'diagnosticoId' con un ID válido de diagnóstico existente en tu base de datos.

3.Verificar que la respuesta contenga los detalles del diagnóstico.

Datos de entrada: API.



Resultado esperado: Formato JSON recomendaciones de usuario.



Nota. Se muestra el resultado de la prueba estructural 6. Elaborado por los autores.

En conclusión, las pruebas estructurales en la aplicación fueron bastantes buenas, destacando las correctas validaciones realizadas a la existencia de las diferentes cédulas registradas en la aplicación. También el correcto funcionamiento de las recomendaciones dadas por el chabot, la actualización del usuario y el registro de un nuevo usuario.

CONCLUSIONES

La investigación realizada sobre orientaciones psicoterapéuticas permitió llevar a cabo un análisis sobre los contenidos y temas más relevantes en el ámbito de la psicoterapia, lo que brindo un entendimiento más profundo sobre los enfoques terapéuticos y los tipos de trastornos más relevantes en área de la salud mental. El logro significativo de la investigación fue la creación de un diccionario de datos conformado por palabras clave, frases y textos relevantes recopilados de la literatura psicoterapéutica, el diccionario de datos creado aporta valor al ser el componente principal durante el entrenamiento del modelo de clasificación.

La recopilación de documentos relacionados con el desarrollo de chatbots permitió identificar las tendencias, prácticas, herramientas y servicios más actuales para el diseño y desarrollo de aplicaciones móviles enfocadas en el área de la salud mental, así como la exploración de tecnologías e innovaciones que emplean el procesamiento del lenguaje natural (NLP) y aprendizaje automático (Machine learning). El resultado obtenido durante el proceso de desarrollo del prototipo, fue la implementación de los diferentes componentes de procesamiento del lenguaje natural y aprendizaje automático, que tuvieron lugar en el análisis y selección adecuada de algoritmos con el fin de realizar una clasificación en base a un texto dado. Además, se gestionaron grandes conjuntos de datos y se considera la necesidad de adaptarse a las particularidades del contexto psicoterapéutico. No obstante, hemos observado áreas de mejora y optimización que podrían ser implementadas a futuro en el algoritmo desarrollado.

El proceso de entrenamiento ha sido parte fundamental para desarrollar modelos de clasificación capaces de predecir la orientación psicoterapéutica a partir del análisis de una conversación realizada entre el usuario y el chatbot. Al implementar Transformers y BERT con técnicas de procesamiento del lenguaje natural (NLP) se ha mejorado significativamente la

comprensión de los textos psicoterapéuticos, al tener la capacidad de capturar relaciones semánticas más complejas en textos extensos como el caso de la interacción entre el usuario y el chatbot que puede llegar a tener una extensión significativa.

El desarrollo e implementación de los módulos diseñados para la interacción con el usuario son un paso crucial para cumplir con los objetivos propuestos, para ello se establece la creación de un módulo de bienvenida, registro de usuario, inicio de sesión, chatbot que integra el servicio OpenAI y el módulo de recomendación de la orientación psicoterapéutica que trabaja en conjunto con los modelos entrenados por transformers y BERT. El logro obtenido durante las pruebas con los usuarios, evidenciaron la capacidad de la aplicación para proporcionar un espacio de interacción entre el usuario y el asistente virtual a través de una conversación. Uno de los hallazgos significativos fue el módulo de recomendación de la orientación psicoterapéutica que, al realizar un análisis de la conversación, utilizando los modelos entrenados por transformers y BERT, revelaron su utilidad al interpretar la información proveniente de la interacción entre el usuario y el chatbot, presentando la orientación psicoterapéutica al usuario.

Para concluir, el prototipo de aplicación móvil desarrollado ha alcanzado con éxito los objetivos propuestos, al permitir que los usuarios interactúen con el asistente virtual del chatbot, con la finalidad de generar una conversación que puede ser extensa o corta y posteriormente analizada utilizando los modelos entrenados previamente por las técnicas de procesamiento natural del lenguaje (NLP) y aprendizaje automático (Machine learning). Los logros alcanzados una vez que el usuario haya obtenido la orientación psicoterapéutica más adecuada, en base a las pruebas de aceptación, permitieron conocer en la primera pregunta que el cien por ciento de los usuarios se sintieron respaldados con la interacción del chatbot, en la segunda pregunta se identificó que el cien por ciento recomendaría la aplicación a otros usuarios, en la tercera

pregunta se idéntico que un diez por ciento no estaba de acuerdo con la orientación psicoterapéutica recomendada pero el noventa por ciento estaba de acuerdo con el resultado obtenido, en la cuarta pregunta se evidenció que el treinta por ciento de los usuarios tuvo dificultades durante el uso de la aplicación, esto indica que se deben realizar ajustes con la interfaz de usuario en futuras actualizaciones, por otro lado el setenta por ciento de los usuarios indico que no tuvo dificultades durante la interacción con la aplicación, en la pregunta cinco se evidencio que el cien por ciento de los usuarios encontraron útiles las funciones de recomendación proporcionadas por la aplicación. Los resultados obtenidos en las pruebas de aceptación reflejan que la aplicación móvil podría ser una herramienta de apoyo a futuro.

RECOMENDACIONES

Para garantizar la robustez y adaptabilidad del modelo de clasificación, se sugiere actualizar de forma continua el diccionario de datos. La propuesta consiste en incorporar de manera constante y sistemática términos y conceptos emergentes en la literatura psicoterapéutica, Esta práctica aseguraría que el modelo esté preparado para enfrentar de manera proactiva los nuevos enfoques y desarrollos que continuamente surgen en el ámbito de la psicoterapia.

Aunque el proceso de entrenamiento actualmente se aproxima al ámbito psicoterpéutico, se sugiere realizar mejoras continuas para optimizar más su rendimiento. Se propone explorar diversas estrategias de fine-tuning para los modelos transformers y BERT, incluyendo la optimización de hiperparámetros y la incorporación de conjuntos de datos adicionales pertinentes. Estos ajustes pretenden garantizar un entrenamiento más preciso y adaptado a las necesidades específicas de la aplicación.

Es importante mantener un enfoque modular y gradual en el desarrollo de la aplicación para asegurar una organización, reutilización y escalabilidad del código. Así mismo es necesario documentar regularmente los cambios en cada módulo, lo que simplifica las configuraciones del entorno de desarrollo y contribuye al mantenimiento continuo. Este método no solo facilita el manejo y la evolución del proyecto, sino que también permite desarrollar cada componente de manera independiente antes de integrarlo en la aplicación principal.

Se aconseja seleccionar minuciosamente las librerías utilizadas en el desarrollo de la aplicación, asegurándose de elegir aquellas que se encuentre actualizadas y tengan soporte. Además, es importante conocer las mejoras y revisar la documentación oficial de las librerías para garantizar su correcta implementación.

BIBLIOGRAFÍA

Boucher, E. M., Harake, N. R., Ward, H. E., Stoeckl, S. E., Vargas, J., Minkel, J., . . . Zilca, R. (2021). Artificially intelligent chatbots in digital mental health interventions: a review, Expert Review of Medical Devices. *Taylor & Francis Online*, 37-49. https://doi.org/https://doi.org/10.1080/17434440.2021.2013200

Danielsson, W. (2016). *React Native application development.* Linköpings universitet. https://www.diva-portal.org/smash/get/diva2:998793/FULLTEXT02.pdf

Developers, A. (27 de 03 de 2023). *Android Debug Bridge (ADB)*. https://developer.android.com/studio/command-line/adb?hl=es-419

Ezequiel, B. (Abril de 2009). *Redalyc.* Retrieved 13 de Julio de 2023, from Las psicoterapias: https://www.redalyc.org/pdf/3331/333127084005.pdf

Fernández Trejo, D., Lucas, M., & Rodríguez, J. L. (04 de Diciembre de 2019). *Llorenteycuenca.* Retrieved 13 de Julio de 2023, from CHATBOTS Y AGENTES INTELIGENTES. EL DESAFÍO DE LA CONVERSACIÓN ARTIFICIAL: https://ideas.llorenteycuenca.com/2019/12/chatbots-y-agentes-inteligentes-el-desafio-de-la-conversacion-artificial/

González Duque, R. (2011). *Python para todos.* https://persoal.citius.usc.es/eva.cernadas/informaticaparacientificos/material/libros/Python%20para%20todos.pdf

González Loyola, O. (2019). Black-Box vs. White-Box: Understanding Their Advantages and Weaknesses From a Practical Point of View. *IEEE Xplore*. https://doi.org/10.1109/ACCESS.2019.2949286

Grinberg, M. (2014). *Flask Web Development.* O'Reilly Media Inc. https://coddyschool.com/upload/Flask_Web_Development_Developing.pdf

Gupta, A., Kumar Panda, D., & Pande, M. (2018). *Development of Mobile Application for Laundry Services Using Android Studio.* International Journal of Modern Computer Science. https://www.ripublication.com/ijaer18/ijaerv13n12_71.pdf

Lamas, C. M., Distefano, J. M., Cataldo, G., Mongelo, C. M., & Mesurado, B. (2018). *Redalyc*. Retrieved 15 de Julio de 2023, from Conocimiento y uso de tecnologías digitales en psicoterapia entre los psicólogos de Buenos: https://www.redalyc.org/journal/4835/483555971007/483555971007.pdf

Lima, A., Rossi, L., & Musolesi, M. (2014). Coding Together at Scale: GitHub as a Collaborative Social Network. *Proceedings of the International AAAI Conference on Web and Social Media*, 1-8. https://arxiv.org/abs/1407.2535

Linares, I. (25 de 11 de 2021). *NoxPlayer, el emulador Android para el ordenador con el que arrancar tus apps y juegos favoritos.* Xataka: https://www.xatakandroid.com/aplicaciones-android/apps-juegos-android-ordenador-noxplayer-uno-mejores-emuladores-que-puedes-descargar-gratis

Microsoft. (27 de 11 de 2023). *Visual Studio: IDE y editor de código para desarrolladores de software y teams.* https://visualstudio.microsoft.com/es/#:~:text=Visual%20Studio%20Code%20es%20un,para%20Windows%2C%20macOS%20y%20Linux.

Montaño, S., & Fernando, A. (2019). *Repositorio UTP*. Retrieved 14 de Julio de 2023, from Aplicación web para el centro psicoterapéutico renacer: https://repositorio.utp.edu.co/items/0492ad17-4f3f-4caf-b5a1-6ac967c10349

Muela Aparicio, A., & Méndez Eneko, S. (2020). *Scielo.* Retrieved 10 de Julio de 2023, from Tratamientos psicológicos personalizados: orientaciones clínicas: https://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S0214-78232020000100016

Naranjo Heredia, S. J. (2021). *Análisis del stack Mern y su uso en las aplicaciones web.* Babahoyo: Universidad técnica de Babahoyo. http://dspace.utb.edu.ec/bitstream/handle/49000/9530/E-UTB-FAFI-SIST-000191.pdf?sequence=1&isAllowed=y

Rahimi, F., & Bezmin Abadi, A. T. (2023). ChatGPT and Publication Ethics. https://doi.org/0188-4409

Ramírez Benítez, Y. (Febrero de 2007). *Pepsic.* Retrieved 10 de Julio de 2023, from La Orientación Psicológica, un Espacio de Búsqueda y Reflexión Necesario para Todos: http://pepsic.bvsalud.org/scielo.php?script=sci_arttext&pid=S1665-75272007000200005

Romero, M., Casadevante, C., & Montoro, H. (2020). *Scielo.* Retrieved 11 de Julio de 2023, from Cómo construir un psicólogo Chatbot: https://scielo.isciii.es/pdf/pappsicol/v41n1/0214-7823-pappsicol-41-1-27.pdf

Sedlakova, J., & Trachsel, M. (2023). Conversational Artificial Intelligence in Psychotherapy: A New Therapeutic Tool or Agent? *Taylor & Francis Online*, 4-13. https://doi.org/10.1080/15265161.2022.2048739

Smith, M. L., & Glass, G. V. (Septiembre de 1977). *Researchgate.* Retrieved 13 de Julio de 2023, from Meta-analysis of comparative therapy outcome studies a replication and refinement: https://www.researchgate.net/publication/22234005_Meta-Analysis_of_Psychotherapy_Outcome_Studies

Velásquez, S. M., Vahos Montoya, J. D., Gómez Adasme, M. E., Restrepo Zapata, E. J., Pino, A. A., & Londoño Marín, S. (2019). Una revisión comparativa de la literatura acerca de metodologías tradicionales y modernas de desarrollo de software. *Revista CINTEX*, 13-23. https://doi.org/https://doi.org/10.33131/24222208.334

Zottola, R. (14 de 09 de 2023). *Arquitectura N-Tier y el Enfoque de 3 Capas*. https://dr-zottola.medium.com/arquitectura-n-tier-y-el-enfoque-de-3-capas-f7f91cd6dcab