UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE COMPUTACIÓN

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA EL AGENDAMIENTO DE CITAS DE CONSULTAS MÉDICAS UTILIZANDO TÉCNICAS DE PROCESAMIENTO DE LENGUAJE NATURAL APLICADAS A UN ASISTENTE VIRTUAL

Trabajo de titulación previo a la obtención del
título de Ingeniero en Ciencias de la Computación

AUTORES: ALEX JESSIEL REINOSO GONZÁLEZ
CHRISTIAN DANIEL ZHIRZHAN CABRERA
TUTOR: ING. GABRIEL ALEJANDRO LEÓN PAREDES, Ph.D.

Cuenca - Ecuador
2022

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Alex Jessiel Reinoso González con documento de identificación N° 1400919302 y Christian Daniel Zhirzhan Cabrera con documento de identificación N° 0151027299; 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.

Cuenca, 08 de marzo del 2022.

Atentamente,

Alex Jessiel Reinoso González	Christian Daniel Zhirzhan Cabrera
1400919302	0151027299

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

Nosotros, Alex Jessiel Reinoso González con documento de identificación N° 1400919302 y Christian Daniel Zhirzhan Cabrera con documento de identificación N° 0151027299, 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: "Desarrollo de una aplicación móvil para el agendamiento de citas de consultas médicas utilizando técnicas de procesamiento de lenguaje natural aplicadas a un asistente virtual", el cual ha sido desarrollado para optar por el título de: Ingeniero 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.

Cuenca, 08 de marzo del 2022.

Atentamente,

Alex Jessiel Reinoso González	Christian Daniel Zhirzhan Cabrera
1400919302	0151027299

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Gabriel Alejandro León Paredes con documento de identificación N° 0103652186, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO DE UNA APLICACIÓN MÓVIL PARA EL AGENDAMIENTO DE CITAS DE CONSULTAS MÉDICAS UTILIZANDO TÉCNICAS DE PROCESAMIENTO DE LENGUAJE NATURAL APLICADAS A UN ASISTENTE VIRTUAL, realizado por Alex Jessiel Reinoso González con documento de identificación N° 1400919302 y por Christian Daniel Zhirzhan Cabrera con documento de identificación N° 0151027299, 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.

Cuenca, 08 de marzo del 2022.

Atentamente,

Ing. Gabriel Alejandro León Paredes, Ph.D
0103652186

DEDICATORIA

Yo, Alex J. Reinoso, en reciprocidad al apoyo, consejos y amor brindado. Dedico este trabajo a mis padres y abuelitos, ya que su amor y consejos me impulsaron en mi carrera universitaria. Este logro lo comparto con ellos ya que fueron de vital importancia para mí.

Yo, Christian D. Zhirzhan en reciprocidad al apoyo, dedicación, sacrificio, amor y comprensión recibidos. Dedico este trabajo a mi madre, mis abuelitos y tíos, dado que gracias a sus consejos tuve la fuerza y motivación para concluir mi carrera universitaria, no lo pude haber logrado sin ellos por lo que este triunfo también les pertenece.

AGRADECIMIENTOS

Yo Alex J. Reinoso, expreso mi infinita gratitud con Dios por el cuidado y bendiciones brindadas a mi persona de manera incondicional.

A mis amados padres, Claudio y Sonia, por haberme apoyado y creído en mí en cada una de las metas que me he propuesto. Así también, a mis padres de intercambio, Glenn y Lynn, por su amor y apoyo incondicional.

A mi hermana Sharon y tías Nadia y Diana, por su amor, apoyo y ayuda en momentos de dificultad, al igual que sus consejos invaluables. Mil gracias de corazón.

De igual manera, expreso mi agradecimiento a mis amigos por pasar juntos los buenos y malos momentos. Gracias: Eduardo Reinozo, Edisson Reinozo, Freddy Roldan, Pablo Calle, Macarena Villaroel y Manolo Morocho.

Yo Christian D. Zhirzhan, en primera instancia agradezco a mi madre Martha que a pesar de la distancia siempre estuvo apoyándome y nunca me dejó solo a lo largo de este camino.

A mis abuelos, Rosa y Rafael, por brindarme su amor y apoyo en los momentos más difíciles, les agradezco infinitamente.

A mis tíos, Alfredo, Yolanda y Susana, por siempre creer en mí y brindarme sus valiosos consejos que me ayudaron a cumplir esta meta, muchas gracias.

También expreso mi agradecimiento a mis amigos, Rosa Orellana, Joel Zuña, Daniel Coronel, John Barzallo, Evelyn Pintado y Edwin Marquez, muchas gracias por impulsarme a cumplir esta meta y animarme en mis momentos difíciles.

De igual manera, nuestro agradecimiento a la Universidad Politécnica Salesiana del Ecuador, a todos quienes conforman la carrera de Computación, a nuestros profesores, en especial al Dr. Gustavo Bravo y Dr. Remigio Hurtado quienes no solo son ejemplo de profesionalismo sino también de humanismo; agradecemos a cada uno de ellos por las enseñanzas y retos que nos han ayudado a ser mejores seres humanos y contribuyeron en nuestra formación profesional.

Finalmente, deseamos expresar nuestra más sincera gratitud al Dr. Gabriel Paredes, nuestro tutor, quien, con su guianza, enseñanzas, paciencia, amistad y excelencia en docencia nos brindó los conocimientos que permitieron la culminación de este trabajo de titulación.

RESUMEN

Debido al crecimiento apresurado de tecnologías de la información y por ende la digitalización de procesos y actividades, es de vital importancia la automatización de procesos repetitivos con el afán de hacer mejor uso del tiempo, esto acompañado con la pandemia que ha forzado a varios sectores productivos de la sociedad, como lo es la salud, a mudar parte de sus operaciones a medios digitales. Por consecuencia, esta migración a medios digitales a forzado a varios usuarios el ser más dependiente de aplicaciones y plataformas digitales, lo que para algunos representa un reto puesto que carecen de la experticia manejando estas.

Por esta razón y la falta de aplicaciones que simplifiquen el proceso de agendamiento de citas médicas, se ha propuesto la implementación de un chatbot, asistente virtual, basado en reglas que facilite la búsqueda y agendamiento de citas médicas mediante el Procesamiento del Lenguaje Natural (PLN) dentro del área geográfica de la ciudad de Cuenca, provincia del Azuay – Ecuador. Consideramos que la implementación de este acortaría la brecha entre el usuario y la tecnología, al proveer una interacción más cercana a la humana.

En la parte de la interfaz de usuario, el sistema se encuentra dividido en dos aplicaciones multiplataforma desarrolladas con Ionic; una destinada para el profesional de la salud, y otra para los pacientes. En el caso de la aplicación de los pacientes, esta hace uso de los servicios provistos por el servidor de RASA para procesar las entradas del usuario y responder según estas; y a su vez, tanto la aplicación del paciente como la del profesional de la salud consumen servicios de un servidor desarrollado con Spring.

En la parte de la lógica se ha optado por una arquitectura de microservicios por su flexibilidad, capacidad de escalado e independencia, todo esto haciendo uso de software libre, como lo es el Open JDK 11 de java y RASA Open Source Conversational A.I., y las aplicaciones móviles con el marco de trabajo multiplataforma IONIC, de esta forma puede ser implementada para iOS, Android e inclusive la web como una aplicación web progresiva.

Palabras clave

RASA, Ionic, PLN, Spring, Backend, Frontend, profesional de la salud, paciente, CLN, Interfaz, médico

ABSTRACT

Due to the rapid growth of information technologies, the transition to digital platforms and the digitalization of processes and activities, the automation of repetitive tasks is of vital importance to make better use of human resources and time, this accompanied by the pandemic that has forced several productive sectors of society, such as health, to move part of its operations to digital platforms.

Consequently, this migration to digital platforms has forced several users to be more dependent of digital applications and platforms, which for some users represents a challenge due to lack of expertise in handling such platforms.

For this reason and the lack of applications that simplify the process of scheduling medical appointments, the implementation of a chatbot, virtual assistant, based on rules that facilitates the search and scheduling of medical appointments through Natural Language Processing (NLP) has been proposed to be used within the geographic area of the city of Cuenca, province of Azuay – Ecuador.

We believe that the implementation of this would shorten the gap between the user and technology, by providing an interaction closer to the human.

For the user interface, the system is divided into two cross-platform applications developed with Ionic; one intended for the health professional, and another for patients. In the case of the patient application, it makes use of the services provided by the RASA server to process the user's inputs and respond to them; both the patient application and the health professional application consume services from a server developed with Spring.

In the logic part, a microservices architecture has been chosen due to its flexibility, scalability, and independence, all this using Open-Source software, such as Java's Open JDK 11 and RASA Open-Source Conversational A.I, and the mobile applications with the IONIC cross-platform framework, so it can be implemented for iOS, Android and even the web as a progressive web application.

Índice de contenido

Índice de figuras

1. Introducción

El crecimiento exponencial del uso de dispositivos "inteligentes" y los avances tecnológicos, han hecho que estos cada vez sean más indispensables y que dependamos de ellos para realizar ciertas actividades como: ordenar comida, verificar nuestro estado bancario, consultar el clima, entre otros. Sin embargo, las aplicaciones enfocadas en el agendamiento de citas médicas no han tenido una gran acogida en el país, debido a que algunas se centran en citas médicas telemáticas o carecen de características que mejoren la interacción con el usuario, esto sumado con la necesidad de maneras no presenciales de agendar citas médicas debido a la pandemia presenta un verdadero problema.

Se tiene como objetivo desarrollar una aplicación móvil que facilite el agendamiento de citas médicas, gestión de estas y triaje de una manera sencilla y amigable con el usuario y el profesional de la salud. Dicha aplicación móvil contará con un asistente virtual, el cual hace uso de técnicas de procesamiento del lenguaje natural con la finalidad de que la interacción con la aplicación sea similar a la de paciente – doctor; además, eliminar la necesidad de que el agendamiento y gestión de citas médicas sea supervisado por un humano, permitiendo la optimización de los recursos humanos.

Este asistente virtual esta implementado en la aplicación dedicada a los pacientes, esto con la finalidad de que se agenden las citas médicas mediante una conversación por mensajes de texto. Por otro lado, en la aplicación del médico se permite visualizar las citas médicas e información relevante de la misma.

Este asistente está desarrollado con tecnologías de software libre, con el afán de que pueda ser utilizado a futuro sin restricciones y poder implementar más funcionalidades que vayan en beneficio del usuario final.

1.1. Problema

1.1.1. Antecedentes

Las aplicaciones dedicadas al área de la salud en el Ecuador no proveen una manera amigable de agendar citas médicas; por un lado, el Ministerio de Salud Pública (MSP) del Ecuador habilitó líneas para el agendamiento de citas médicas, pero según reporta el Diario el Telégrafo, "un comunicado emitido el 21 de julio esta cartera de Estado informó que están presentando inconvenientes" (Telégrafo, 2021), razón por la cual se recomendaba hacer uso de la aplicación SaludEC. Sin embargo, al revisar las calificaciones dadas por los usuarios en las respectivas tiendas, se puede observar qué esta cuenta con una calificación de 3.2 estrellas sobre cinco en la tienda de aplicaciones de Google; y en la tienda de aplicaciones para iOS, esta cuenta con una calificación de 2.5 estrellas sobre cinco. En su gran mayoría, las críticas en cuanto a la aplicación hacen referencia a:

- Problemas para mantener sesión (Prichsouth Tecnologías del Sur, 2021).
- No poder actualizar datos personales (Prichsouth Tecnologías del Sur, 2021).
- Dificultad para agendar una cita médica (Prichsouth Tecnologías del Sur, 2021).
- Falta de información en cuanto a la cita médica agendada (Prichsouth Tecnologías del Sur, 2021).

Por el contrario, revisando las alternativas para el sector privado en Ecuador tanto en la tienda de aplicaciones de iOS y Android, se puede encontrar cuatro aplicaciones, las más notorias y con calificaciones por parte de usuarios, de estas cuatro dos se centran en telemedicina antes que la gestión de citas médicas. Siendo la primera de estas Telemédico, cuya descripción dada por los desarrolladores en la Google Play Store afirman que esta es una plataforma en la cual se pueden realizar videoconferencias con profesionales de la salud certificados, a cualquier hora y por un costo accesible, (Telemédico Ecuador, 2021); y la segunda, TL Doctor cuyo funcionamiento y descripción es la misma de la aplicación Telemédico. Estas dos aplicaciones están desarrolladas por la empresa Telemédico Ecuador, y a diferencia de las aplicaciones móviles, esta cuenta con una manera de agendar una cita médica mediante la página web.

Por último, las dos aplicaciones restantes, que se centran en el agendamiento y gestión de citas médicas, son: Doctorisy e iSalud. Estas dos aplicaciones cuentan con una calificación superior a cuatro sobre cinco estrellas en la tienda de aplicaciones de Google. Por un lado, Doctorisy según la descripción dada por sus desarrolladores, garantizan el agendamiento de una cita médica en 3 clics y notificaciones para evitar el olvido de las citas médicas, (Doctorisy, 2021). En el caso de iSalud, esta solo cuenta con aplicación móvil para dispositivos Android; sus desarrolladores afirman, “Nuestro sistema mixto (marketing y administrativo) catalogado el más moderno del país, cuenta con una importante base de datos con múltiple información que nos permite mejorar la experiencia de los usuarios” (iSalud, 2021).

De igual manera, al revisar las alternativas dadas por los principales hospitales privados en la ciudad de Cuenca: Hospital Universitario del Rio, Clínica Santa Ines, Hospital Monte Sinaí; tan solo el Hospital Universitario del Rio cuenta con una aplicación móvil, disponible únicamente para iOS, la cual permite acceder al catálogo de profesionales de la salud y agendar citas médicas, esto mediante correo electrónico o contactando a las secretarias por llamada telefónica, (Primmelabs Tecnologia S.A., 2019), esta aplicación no es más que una agenda telefónica de los diferentes especialistas de la salud que laboran en el hospital, y no permite tener un seguimiento mediante la aplicación móvil de las citas médicas agendadas por el paciente.

1.1.2. Importancia y alcances

Es importante desarrollar tecnologías que vayan a la par con las demandas y crecimiento de los

usuarios de plataformas digitales, sin olvidarse de aquellos usuarios que presentan dificultades al manejar sistemas de agendamiento de citas manuales; razón por la cual, la implementación de un asistente virtual que procese el lenguaje natural para agilizar este proceso es de vital importancia.

El procesamiento del lenguaje natural e inteligencia artificial, al ser una tecnología relativamente nueva, no ha sido utilizada a gran escala en nuestro país, y mucho menos se ha pensado en asistentes virtuales enfocados en al área de la salud. El uso de este tipo de tecnologías tiene una gran demanda para aquellas tareas que requieren una interacción con clientes de manera intensiva. Ecuador está a medio camino en cuanto a la transformación digital, razón por la cual, el implementar un asistente virtual enfocado en agilizar el proceso de agendamiento de citas médicas es de vital importancia. Según un estudio llevado a cabo en la Universidad de Tecnología de Swinburne en cuanto al impacto en el sentimiento de clientes en el área de ventas, el promedio sentimiento del cliente es mayor que al de un agente humano, teniendo un sentimiento negativo mayor el agente humano; sin embargo, el sentimiento se torna negativo con forme la tecnología madura (Tran et al., 2021). Tomando esto en cuenta, podemos aseverar que el uso de este tipo de tecnología tendría un efecto bastante positivo en el usuario final, paciente, y se debe tener un control de calidad conforme la tecnología madura para desarrollarse en al área de especialización.

1.1.3. Delimitación

Los profesionales de la salud, y pacientes que requieran atención dentro de la región geográfica de Cuenca, Azuay – Ecuador.

1.2. Objetivos generales y específicos

En esta sección se explica el objetivo general y los objetivos específicos.

1.2.1. Objetivo general

Desarrollar una aplicación móvil para el agendamiento de citas de consultas médicas utilizando técnicas de procesamiento de lenguaje natural aplicadas a un asistente virtual.

1.2.2. Objetivos específicos

- **OE1.** Estudiar y conocer los fundamentos y técnicas del procesamiento del lenguaje Natural.

- **OE2.** Desarrollar e implementar algoritmos de PLN para el agendamiento de citas médicas.

- **OE3.** Análisis, diseño, desarrollo e implementación de una aplicación móvil para el paciente y profesional de la salud (elicitación, prototipos, modelados,

etc.).

- **OE4.** Implementar pruebas de funcionamiento de la aplicación móvil.

2. Fundamentos teóricos

En este capítulo se describen los principales contenidos teóricos, tecnologías y metodologías fundamentales para el desarrollo de este proyecto técnico.

2.1. Inteligencia Artificial (I.A.)

Según Rouhiainen, "la IA es la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano. Sin embargo, a diferencia de las personas, los dispositivos basados en IA no necesitan descansar y pueden analizar grandes volúmenes de información a la vez" (Rouhiainen, 2018), por esta razón la IA ha tenido varias aplicaciones que buscan automatizar procesos, un derivado de la IA es el Procesamiento Natural del Lenguaje, el cual busca entender computacionalmente el lenguaje de los humanos.

2.1.1. Procesamiento del Lenguaje Natural (PLN)

Según Deng y Liu, el procesamiento del lenguaje natural investiga el uso de computadoras para procesar o entender lenguajes, con la finalidad de realizar tareas útiles. Este es un campo interdisciplinario que combina lingüística computacional, ciencias de la computación, ciencia cognitiva, e inteligencia artificial (Deng & Liu, 2018), este campo está enfocado en facilitar la interacción entre el humano y la máquina, a manera de que esta interacción sea lo más cercana posible a la humano-humano. Es importante recalcar, que el PLN es la primera etapa para la generación de asistentes virtuales, puesto que es necesario comprender el lenguaje natural para finalmente generar lenguaje natural en respuesta a las entradas del sistema.

2.1.2. Comprensión del Lenguaje Natural (CLN)

Esta es una rama de la IA que hace uso de software y procesamiento computacional para entender las entradas al sistema ya sea por texto o voz. Por ende, el CLN permite la interacción humano-maquina, puesto que permite a las computadoras comunicarse con los humanos ya sea: respondiendo a comandos, generando procesos, generando lenguaje natural, entre otros.

Dentro de este campo, existen dos conceptos fundamentales:

- **Intención**

  Este concepto está relacionado con el análisis de sentimientos, puesto que mediante esta técnica se puede determinar el objetivo de la entrada brindada por el usuario. Esta parte es esencial puesto que determinar el significado del texto o voz de entrada al sistema.

- **Reconocimiento de entidades**

  Este es un tipo de CLN que se enfoca en identificar las entidades en un texto de entrada, para luego extraer la información más importante relacionada a las entidades. Por consecuencia, existen dos tipos de entidades: entidades nombradas y entidades numéricas.

  - **Entidades nombradas**

    Este tipo de entidades se caracterizan por estar agrupadas en categorías; por ejemplo, este tipo de entidades podrían ser: personas, ciudades, comandos computacionales, entre otros.

  - **Entidades numéricas**

    En cuanto a las entidades numéricas, están hacen referencia a las entidades que se reconocen como: números, monedas, fechas, porcentajes, y muchos más.

2.1.3. Generación del lenguaje natural

Permite que las computadoras generen textos de lenguaje natural, copiando la manera en la que los humanos se comunican. Sin embargo, este texto generado carece la fluides, emoción y personalidad que caracteriza los contenidos generados por los humanos (TechTarget Contributor, 2021), no obstante, CLN puede hacer uso de PLN para producir textos similares a los producidos por los humanos al identificar el tema central de un escrito, y mediante el uso de PLN establecer la manera más apropiada de escribir una respuesta a la entrada del usuario.

2.1.4. Asistente virtual

Un asistente virtual hace uso inteligencia artificial para automatizar y llevar a cabo tareas, sin la necesidad de que su funcionamiento este supervisado por un ser humano especializado. Las capacidades de estos son bastante extensas, pueden llevar a cabo tareas tan simples como: consultar el clima, verificar agendas virtuales, realizar llamadas, entre otras; y otras más complejas tales como: tomar notas por voz para traducirlas a texto, realizar recomendaciones y

contar chistes.

2.1.5. Chatbots

Esta es una tecnología derivada de la IA que hace uso de técnicas de PLN para llevar a cabo conversaciones por cuenta propia; este tipo de tecnologías es bastante usada en al área de atención al cliente.

2.2. Herramientas para el Procesamiento del Lenguaje Natural

Para que la experiencia conversacional con el usuario sea óptima la información que recibe el sistema debe de ser procesada, razón por la cual se tiene a disposición varias herramientas con las cuales se pueden analizar los datos y darles una estructura que permita al sistema comprender y brindar respuestas coherentes a lo que el usuario solicite.

2.2.1. NLTK

El kit de herramientas de lenguaje natural con sus siglas en inglés (NLTK) es un conjunto de módulos de código abierto en Python, tutoriales y conjuntos de problemas, que traen una gran variedad de material didáctico sobre procesamiento del lenguaje natural tanto simbólico y estadístico, así también varios algoritmos para preprocesamiento y manipulación de datos, los componentes que trae el NLTK pueden ser reemplazados o aumentados según requieran los procesos que se estén ejecutando (Loper et al., 2002)**.**

2.2.2. TextBlob

TextBlob es una biblioteca de Python (2 y 3) con una API sencilla para el procesamiento de texto en los que se refiera al lenguaje natural (PLN), brinda una gran variedad de métodos como la traducción, clasificación con etiquetas, análisis de sentimientos entre otros que ayudan a una mejor comprensión del lenguaje natural dentro de la informática (Steven, 2020).

2.2.3. Gensim

Gensim es una biblioteca de Python creada para el análisis no supervisado de texto aplicando algoritmos rápidos, escalables y eficientes que no necesitan muchos recursos en memoria para su ejecución, de igual manera también se utiliza para la búsqueda de similitudes e indexación de documentos digitales (Řehůřek & Sojka, 2010).

2.2.4. SpaCy

SpaCy es utilizado en gran parte en la industria o en código de producción ya que no fue desarrollado con un enfoque académico como otras herramientas de este tipo, SpaCy se desarrolló para el procesamiento de texto y lingüística computacional, soporta la tokenización para más de 21 idiomas naturales, 6 modelos estadísticos para 5 idiomas, vectores de palabras previamente entrenados, también incluye el etiquetado de partes del habla y un único reconocedor de entidades, esta herramienta debido a que se usa en el ámbito industrial no se componen de funciones innecesarias, además que tiene la capacidad de funcionar con datos del mundo real y con la posibilidad de ser escalable (Bhargav, 2018)**.**

2.2.5. Web Scraping

Web Scraping no es algo nuevo, su implementación se dio desde los inicios del internet, ha adoptado varios nombres a lo largo del tiempo como screen scraping, data mining, web harvesting u otros nombres similares.

Web scraping técnicamente se usa para la recopilación automatizada de datos a través de la web, este realiza peticiones a servidores web los cuales devuelven la información en formato HTML u otros formatos que se manejan en la red, luego realiza el análisis de los datos para extraer información relevante.

Debido a la gran cantidad y variedad de información que se extrae esta herramienta se compone una amplia variedad de técnicas de programación, análisis de datos, análisis sintáctico y seguridad de la información (Ryan, 2018).

2.2.6. RASA

RASA es utilizado ampliamente para desarrollar software conversacional, se divide en Rasa Core y NLU estas son bibliotecas de código abierto desarrolladas en Python, el objetivo principal de esta herramienta es brindar una facilidad en uso y comprensión del lenguaje basado en el aprendizaje automático sin la necesidad de ser un experto en esta área, y debido a esto es utilizado a nivel mundial por miles de desarrolladores para crear sistemas de conversación para diferentes fines como reservas de vuelos, programación de reuniones entre otras aplicaciones, siendo similar a otros sistemas de conversación este se divide en Comprensión del Lenguaje Natural (Rasa NLU) y gestión del dialogo (Rasa Core) estas dos partes trabajan de manera independiente, también se permite la reutilización de diálogos entrenados en diferentes idiomas (Tom et al., 2017).

Debido a la alta demanda que los sistemas conversacionales tienen hoy en día estos deben ser cada vez más capaces de sostener un dialogo fluido como si de dos personas se tratase, los asistentes virtuales más avanzados en este ámbito actualmente son Siri de Apple, Alexa de Amazon, Google Assistant de Google y Cortana de Microsoft, estos tienen una alta capacidad de comprensión de las peticiones de los usuarios ya sea por voz o texto y no necesariamente

tienen que ser muy formales ya que pueden comprender frases informales que no posean muchas palabras o usando ciertos 'atajos' con algunas palabras como por ejemplo, si se le dice al Asistente de Google "Tengo hambre" este automáticamente hace una sugerencia para reservar una mesa en algún restaurante cercano, esto depende del asistente que se use ya que no todos tendrán la misma interpretación y arrojarán diferentes resultados.

- **Keras**

  Es un framework open source escrito en Python para crear redes neuronales y se ejecuta en la plataforma de aprendizaje automático TensorFlow, (Chollet, 2015).

- **TensorFlow**

  Es una biblioteca escrita por Google la cual se liberó en el año 2015, es utilizada para desarrollar modelos de aprendizaje automático (machine learning), sus redes neuronales trabajan con vectores de datos de diferentes dimensiones, llamados "tensores". (Cabero et al., 2020).

2.3. Arquitecturas de Microservicios

Esta arquitectura consiste en dividir a los sistemas en módulos más simples, esta idea parte de la noción que el software debe ser un servicio. Bob Familiar define que, un microservicio provee una capacidad empresarial o de plataforma a través de una API, contrato de datos, y una configuración bien definida. Proporciona esta función y solo esta función. Hace una sola cosa y la hace bien (Familiar, 2015); entonces, esta arquitectura se enfoca en dividir la solución en componentes de microservicios y tratar a estos componentes como entidades separadas que ayudan a completar una tarea o proceso, y estas son entidades especializadas en realizar una sola tarea en específico. Esta arquitectura es bastante flexible puesto que: permite la reusabilidad de servicios, es tolerante a fallos y el tiempo de respuesta en caso de fallos es razonable.

Las principales ventajas de esta arquitectura son:

2.3.1. Abierto

Familiar explica que, los microservicios están diseñados para exponer su funcionalidad a través de estándares de la industria para API y contratos de datos direccionables a la red (Familiar, 2015), en consecuencia, esto permite que la selección de tecnologías y plataformas este sujeta a las necesidades del servicio y el conjunto de habilidades del equipo de desarrollo, sin afectar la interoperabilidad con sistemas externos.

2.3.2. Alta velocidad

Debido a que los microservicios individuales no requieren de grandes capacidades computacionales por ser componentes de un sistema más grande. Familiar expone que, con un equipo responsable del ciclo de vida del desarrollo y de todos los componentes, tecnologías y automatización que los constituyen, la velocidad a la que un microservicio se puede diseñar, desarrollar, implementar y actualizar es significativamente más rápido que la tratar de realizar las mismas operaciones en una solución monolítica (Familiar, 2015).

2.3.3. Reusabilidad

Debido a que los microservicios funcionan como entidades individuales, estos proveen de capacidades empresariales o de plataforma a través de estándares abiertos de internet, como lo es HTTPS. De esta manera, estos pueden ser invocados conforme se los necesiten.

2.3.4. Escalabilidad

Debido a que en esta arquitectura se definen módulos independientes, cada servicio puede ser escalado de manera independiente sin afectar el funcionamiento del sistema.

2.3.5. Versionable y reemplazable

El versionamiento hace referencia a la capacidad de tener múltiples instancias o versiones del mismo microservicio corriendo al mismo tiempo. Familiar expone que, una nueva versión del API del microservicio puede ser expuesta sin impactar a los clientes haciendo uso de versiones anteriores de la API (Familiar, 2015). Debido a esto que los servicios pueden ser completamente reemplazados manteniendo la API actual y las nuevas características se pueden lanzar bajo nuevas versiones.

2.4. Aplicaciones móviles

El auge de las aplicaciones móviles es una realidad, en los últimos años se han registrado millones de descargas en las diferentes plataformas móviles; así también el costo de adquisición de los dispositivos inteligentes es accesible para la gran mayoría de las personas. Estas en la vida diaria se ha vuelto indispensable porque les permiten estar conectados constantemente, compartiendo datos entre los diferentes usuarios que se encuentren en la red esto incluye el uso de redes sociales, videojuegos, plataformas de streaming entre otras aplicaciones.

Para abastecer la demanda creciente de las aplicaciones móviles se crean nuevos frameworks que permiten el desarrollo de aplicaciones más completas en funcionalidades así mismo con un alto grado de calidad que cumplan todos los requerimientos de los usuarios.

2.5. Aplicaciones hibridas

Se definen como aplicaciones hibridas aquellas que son desarrolladas para ser usadas en diferentes plataformas como iOS y Android, estas usan tecnologías multiplataforma como HTML, JavaScript y CSS que son ejecutadas dentro de un contenedor web.

Esta metodología de desarrollo a parte brindar la posibilidad de distribuir las aplicaciones en las diferentes tiendas de las plataformas existentes también permite la reutilización de código y el aprovechamiento de los componentes de hardware del dispositivo.

Una desventaja para destacar en este tipo de aplicaciones es la lenta ejecución a diferencia de una aplicación nativa, de igual manera su interfaz será igual para todas las plataformas (Lisandro et al., n.d.).

2.5.1. Flutter

Flutter es un framework multiplataforma para desarrollar aplicaciones móviles de alto rendimiento que pueden ser ejecutadas en iOS, Android y Fuschia, este fue creado por Google y lanzado en el 2016.

Flutter tiene su propio motor de renderizado para los componentes de la vista por lo que esto permite que las aplicaciones sean de alto rendimiento, en cuanto a la arquitectura el código C/C++ del motor se compila con el NDK de Android y LLVM en iOS respectivamente, y cualquier código Dart es compilado por AOT en código nativo durante la compilación (Wenhao, 2018).

2.5.2. Ionic

Ionic es un framework para desarrollar aplicaciones independientemente de cualquier plataforma, las tecnologías que utiliza son HTML5, CSS y JavaScript. El proceso de desarrollo es sumamente rápido con un acceso directo a las APIs' con Cordova (Waranashiwar & Ukey, 2018).

2.6. Metodología SCRUM

La metodología de desarrollo que seguirá este proyecto será **SCRUM**, esta metodología requiere de un esfuerzo colaborativo para llevar a cabo el desarrollo y permite la adaptación a un entorno cambiante, requerimientos. SCRUM es un modelo de trabajo que empieza en Japón en la década de los ochenta y es introducido por Hirotaka Takeuchi e Ikujiro Nonaka. Es esencial partir desde las necesidades o requerimientos que constan en el Product Backlog para desarrollar la aplicación, se llevarán a cabo sesiones de Sprint Planning Meeting para planificar

como se desarrollará cada fase del proyecto, como resultado de estas sesiones se obtendrá el Sprint Backlog con las tareas que se cumplirán en cierto periodo de tiempo según la complejidad de estas, de esta manera se dará un seguimiento en todo el desarrollo del proyecto hasta cumplir todas las fases previstas. Para esta sección nos basamos en el Scrum Handbook de J. Sutherland (Sutherland, 2010).

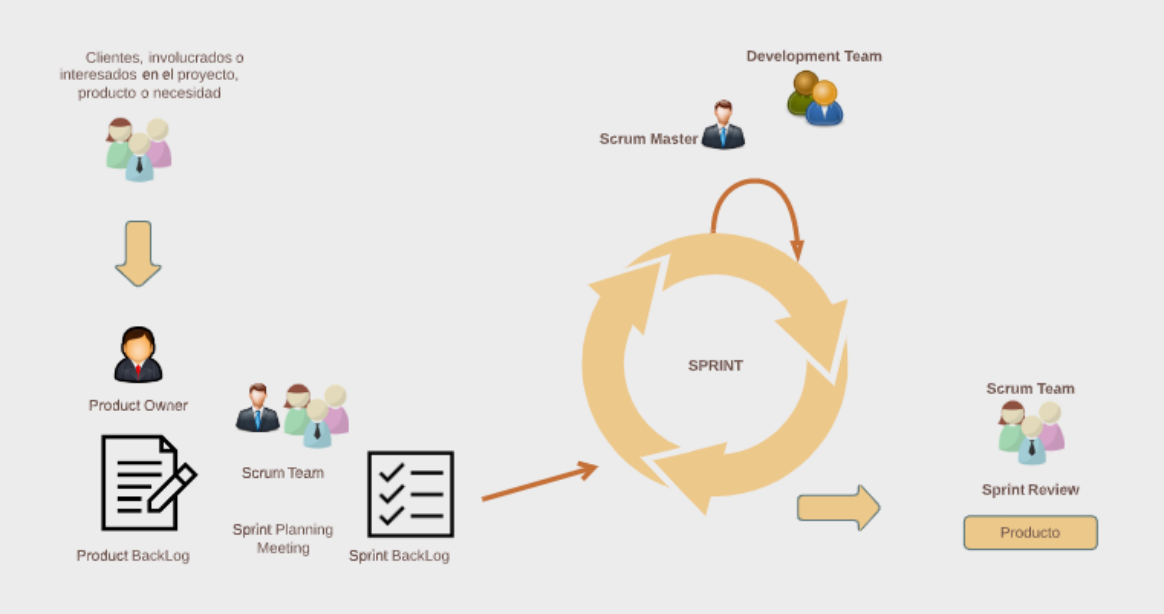


Figura 1. Gráfico Metodología SCRUM

2.6.1. Roles

SCRUM se compone de tres roles principales:

2.6.1.1. Product Owner (Dueño del producto)

Este se encarga de presentar las necesidades de los clientes dentro del equipo de trabajo.

2.6.1.2. SCRUM Máster

Es el encargado de asegurar que todo el ciclo de vida de Scrum se cumpla correctamente dentro del equipo de trabajo asegurándose que se sigan las reglas de Scrum.

2.6.1.3. Development Team

Es el personal capacitado en el desarrollo de software el cual se encargará de construir o solucionar las necesidades de los clientes.

2.6.2. Componentes de SCRUM

Para que SCRUM funcione se debe tener en claro sus componentes y su función.

- **Product Backlog**
  Esta es una lista de requerimientos los cuales se organizan por su relevancia para el cliente o el negocio, estos requerimientos están dados por el **Product Owner**.

- **Sprint Backlog**
  Al momento de iniciar un nuevo Sprint, subconjunto de requerimientos del **Product Backlog** y estos serán los objetivos, requerimientos, a completar por el equipo de desarrollo durante el Sprint.

- **Spring Planning**

  Estas son las reuniones de planeación que se llevan a cabo antes de iniciar cada Sprint, sirven para seleccionar el **Sprint Backlog**, cada requerimiento del Product Backlog se lo desglosa en tareas individuales y estas se las documenta en el **Sprint Backlog.**

- **Daily Scrum Meeting**

  Estas son reuniones que se llevan a diario, no necesariamente, una vez se ha dado inicio al Sprint, sirven para inspeccionar y ponerse al tanto del progreso. Su principal característica es que son rápidas, no más de 15 minutos por encuentro.

- **Sprint Review y Retrospectiva**

  Se realiza al final de cada **Sprint,** en estas reuniones el equipo Scrum y los inversionistas inspeccionan los resultados del Sprint y se determina las siguientes actividades para el próximo Sprint.

3. Marco metodológico

El proyecto se compone de tres fases, la **primera** fase consta en estudiar y conocer los fundamentos y técnicas del procesamiento del lenguaje natural (NLP) en formato de texto; debido a que se necesita procesar las entradas en texto del usuario mediante el asistente virtual, con la finalidad de recaudar datos relevantes para el agendamiento de la cita médica. Los datos para recaudar serian: información personal, síntomas, entre otros.

La **segunda** fase se centra en desarrollar e implementar algoritmos de NLP para el agendamiento de citas médicas, esto consiste en proveer una buena experiencia conversacional con el paciente para que aumente la productividad del negocio, el cual en este caso sería el consultorio médico, al facilitar el proceso que implica el agendamiento de citas médicas.

La **tercera** fase es el análisis, diseño, desarrollo e implementación de una aplicación móvil para los pacientes y profesionales de la salud. Por tal motivo, esto implica el uso de metodologías, técnicas, herramientas y documentación de ingeniería de software para crear un sistema confiable y que cumpla con los requisitos tanto del médico como del paciente, así mismo dentro de esta fase se realizarán las pruebas correspondientes del sistema una vez finalizado.

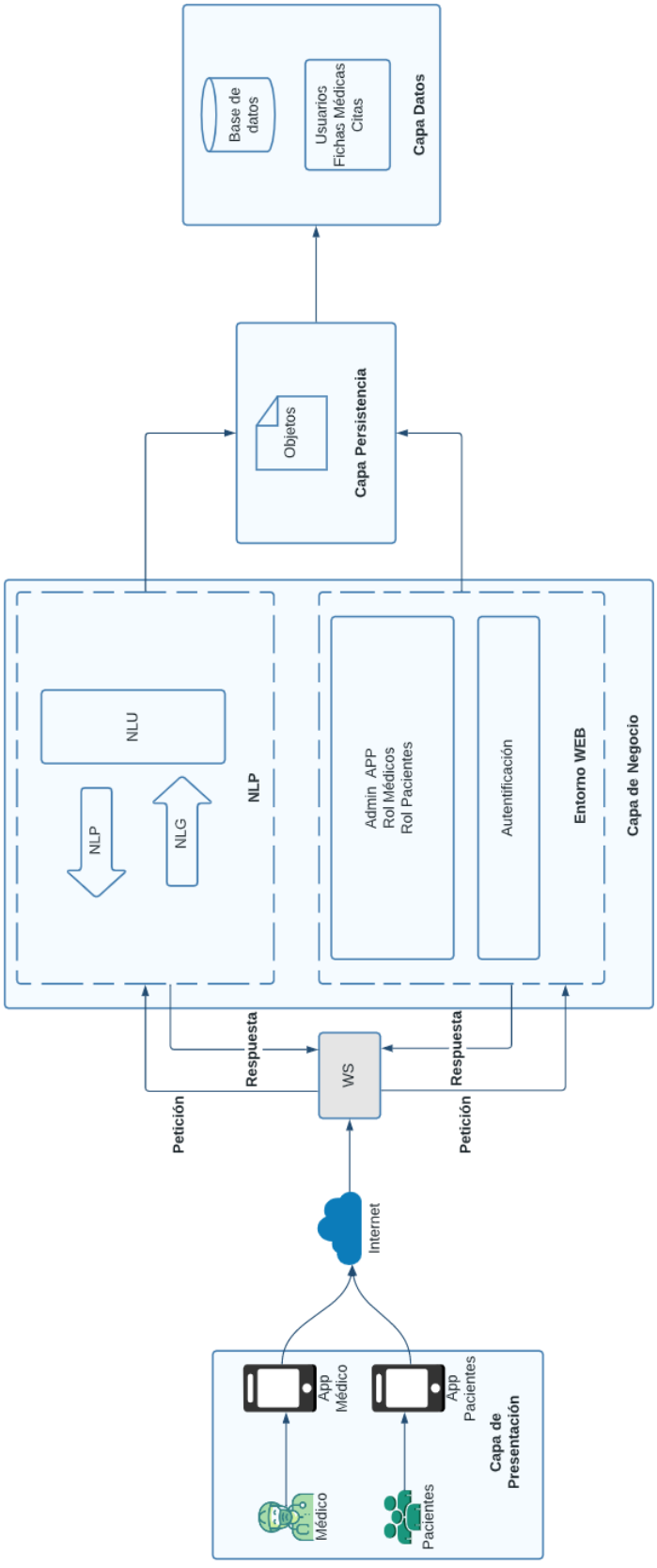


Figura 2. Diagrama de la propuesta de solución

En el diagrama de la Figura *2* se puede observar la relación entre las diferentes capas del sistema, así como la relación con los usuarios.

Entonces, el sistema propuesto se compone de dos aplicaciones, una será utilizada por el médico y otra por los pacientes, cada paciente debe registrarse en el sistema para tener acceso a las diferentes funcionalidades como: consultar doctores, consultar las especialidades, horarios de atención y el agendamiento de citas médicas. Por otro lado, los médicos deben registrarse en el sistema ingresando datos como: especialidad, datos de contacto, horario de atención, entre otros.

Cuando el paciente requiera agendar una cita con algún médico este debe de iniciar sesión en la aplicación correspondiente, una vez en el sistema se contará con la ayuda de un asistente virtual el cual facilitará el proceso de agendamiento. El proceso del agendamiento de citas médicas tendrá como primer paso la recopilación de los síntomas del paciente posterior a esto se le presentará una lista de días disponibles de cualquier especialidad, una vez elegido el día, el asistente le muestra al paciente una lista de especialidades disponibles para ese día, una vez elegido la especialidad, el asistente lista los médicos disponibles con la especialidad seleccionada; por último, se presentan los horarios disponibles para agendar una cita médica. Una vez terminados cada uno de los pasos el asistente agendará la cita médica y notificará al paciente y al médico la fecha seleccionada y clínica por e-mail.

Asimismo, el médico podrá revisar la lista de citas que tiene agendadas en donde se mostrará la fecha, el paciente y los síntomas de este, los estados de las citas serán Pendiente y Cancelada.

3.1. SCRUM

La metodología que se va a emplear para el desarrollo del proyecto, como ya se mencionó, es SCRUM que tiene como su enfoque inicial en el desarrollo de programas de software.

3.1.1. Roles

Dentro de SCRUM existen tres roles principales los cuales en este proyecto serán asignados de la siguiente manera:

La Dra. Cecilia Ochoa tomará el rol del **Product Owner,** ya que es la encargada de expresar las necesidades de los médicos al momento de llevar las citas con los pacientes.

El PhD. Gabriel León tomará el rol de **ScrumMaster**, y sé que se encargará de dirigir el desarrollo del proyecto asegurándose que se cumplan todas las reglas, conceptos y técnicas de Scrum.

Los estudiantes Alex Reinoso y Christian Zhirzhan serán el equipo de desarrollo **(Development Team**) los cuales se encargarán de solucionar las necesidades de los clientes, en este caso los

médicos – sobre todo en el sector privado.

3.1.2. Proceso

La Dra. Cecilia Ochoa definirá el **Product Backlog,** este contendrá todos los requerimientos o necesidades de los médicos para llevar las citas médicas con sus pacientes.

Luego de definir los requerimientos se realizará el **Sprint Planning Meeting** para conocer cada una de las necesidades o requisitos de los médicos y a su vez definir el **Sprint BackLog**, este que contendrá un subconjunto de los requerimientos que se tendrá que cumplir en un determinado periodo de tipo según su complejidad.

Durante las dos semanas de desarrollo del **Sprint** se realizarán reuniones de seguimiento que duren 20 minutos al menos dos veces por semana entre el equipo de desarrollo y el Scrum Master, para verificar que las tareas se estén realizando correctamente y además cumplir con el plazo de entrega; durante estas reuniones se trataran preguntas especificas referentes a las actividades que realizan los desarrolladores, así también los errores que se presenten durante el desarrollo.

Después de haber concluido el periodo de desarrollo de cada Sprint se realizará una reunión general para la revisión de las actividades cumplidas y analizar el rendimiento del equipo para tomar las medidas necesarias para el próximo Sprint, de esta manera se irán completando cada una de las fases previstas para el proyecto.

3.1.2.1. Lista de actividades

OE1. Estudiar y conocer los fundamentos y técnicas del procesamiento del lenguaje Natural.

Tabla 1: Actividades del Objetivo Específico 1

No.	**Actividad**
1	Estudiar los modelos para el procesamiento del lenguaje natural.

2	Estudiar los chatbots basados en reglas y autodidactas.
3	Documentación del conocimiento relevante adquirido para el informe final

OE2. Desarrollar e implementar algoritmos de PLN para el agendamiento de citas médicas.

Tabla 2: Actividades del Objetivo Específico 2

No.	**Actividad**
1	Determinar las tecnologías a utilizar para el PLN.
2	Determinar el conjunto de datos para el entrenamiento.
3	Implementar y entrenar el modelo de PLN para el agendamiento de citas médicas.
4	Comprobar y validar la eficacia del modelo entrenado.
5	Documentar información relevante al tema en el informe final.

OE3. Análisis, diseño, desarrollo e implementación de una aplicación móvil para el paciente y profesional de la salud. (elicitación, prototipos, modelados, etc.).

Tabla 3: Actividades del Objetivo Específico 3

No.	**Actividad**
1	Especificación y análisis de los requerimientos.
2	Diseño y prototipado de la aplicación.

3	Diseño de la base de datos (modelo entidad relación).
4	Diseño y desarrollo de las interfaces de usuario para las dos aplicaciones (Frontend).
5	Diseño y desarrollo del Backend o capa de servicios.
6	Documentar información relevante para el informe final

OE4. Implementar pruebas de funcionamiento de la aplicación móvil.

Tabla 4: Actividades del Objetivo Específico 4

No.	**Actividad**
1	Diseño de un plan de pruebas de funcionamiento.
2	Ejecución de pruebas.
3	Documentar datos relevantes a las pruebas.
4	Desarrollo del informe final.

3.2. Ingeniería de Software

En este capítulo se detallan los procesos de ingeniería de software previos al desarrollo del proyecto técnico.

3.2.1. Introducción

La especificación de los requerimientos de software del desarrollo de una aplicación móvil para el agendamiento de citas de consultas médicas utilizando técnicas de procesamiento de lenguaje natural aplicadas a un asistente virtual.

3.2.2. Propósito

Presentar de manera organizada los requisitos específicos, tanto funcionales como no funcionales, que el sistema de software deberá satisfacer a manera de facilitar el agendamiento

de citas de consultas médicas en el área geográfica de Cuenca – Ecuador.

3.2.3. Ámbito del sistema

Con este proyecto técnico se espera desarrollar una aplicación móvil que permita el agendamiento de citas médicas de una manera que sea amigable con el usuario, y de igual manera reduciendo la necesidad de supervisar el proceso de agendamiento de citas médicas por parte del profesional de la salud.

A través del sistema el profesional de la salud podrá gestionar las citas médicas agendadas, así también como el agendamiento de citas médicas; y, contar con un sistema de notificaciones al momento que se ha agendado una cita médica y citas médicas pendientes.

Por otra parte, el paciente contara con un asistente virtual el cual facilitara el agendamiento de una consulta médica, así también como su corrección y cancelación de cita médica.

3.2.4. Definiciones, Acrónimos y Abreviaturas

RF: Requerimiento Funcional

PLN: Procesamiento del Lenguaje Natural

Frontend: Refiere a la parte visual del sistema, en este caso las dos aplicaciones móviles, paciente y profesional de la salud.

Backend: Refiere a la implementación de la lógica del sistema, en este caso la lógica del sistema está estructurada a manera de servicios, tales como: autenticación, registro, consulta de bases de datos, consumo del servicio de PLN.

3.2.5. Análisis de requerimiento de software

En esta sección se especifican los requerimientos funcionales y no funcionales del sistema.

3.2.6. Descripción general

Para constituir la aplicación se tiene planeado usar tecnologías emergentes y libres para simplificar el proceso de desarrollo y además darle un rendimiento optimo independientemente del dispositivo o sistema operativo en el que se vaya a ejecutar.

3.2.7. Características de usuarios

En esta sección se describen los 3 tipos de usuarios que tiene la aplicación: Administrador, Médico y Paciente.

- **Administrador**

Tipo de usuario	Administrador
Nivel Educación	Educación superior
Experiencia	Gestión de sistema de información
Actividades	Administrador de usuarios

- **Medico**

Tipo de usuario	Médico
Nivel Educación	Educación superior / Cualquier rama de la medicina
Experiencia	Manejo básico de sistemas de información
Actividades	Atender las citas medicas

- **Paciente**

Tipo de usuario	Paciente
Nivel Educación	Ninguno
Experiencia	Manejo básico de sistemas de información
Actividades	Realizar citas médicas

3.2.8. Restricciones

- Para usar la aplicación se necesita conexión a internet.
- Interfaces graficas intuitivas en ambas aplicaciones.
- Se utilizará Ionic como framework de desarrollo de las interfaces.

3.2.9. Requerimientos funcionales

Código Requerimiento	RF01
Nombre	Registro paciente
Propósito	Registrar un usuario de tipo paciente que sea capaz de agendar citas médicas
Descripción	Una vez el usuario acceda a la aplicación móvil, este será presentado con una pantalla de inicio de sesión con la opción de registrarse en el caso no contar con una cuenta existente.
Entrada	Formulario de registro de usuario.
Salida	Mensaje de bienvenida y redirección a la vista de inicio de sesión.
Prioridad	Alta

Código Requerimiento	RF02
Nombre	Registro profesional de la salud
Propósito	Registro de usuario de tipo Profesional de la Salud el cual brinda el servicio de consultas médicas.
Descripción	El registro del profesional de la salud se realiza en una aplicación móvil diferente a la del usuario de tipo paciente. Una vez el usuario entre a la aplicación móvil en la vista de inicio de sesión podrá acceder al formulario de registro.
Entrada	Formulario de registro de usuario profesional de la salud
Salida	Mensaje de bienvenida y redirección a la vista de inicio de sesión.
Prioridad	Alta

Código Requerimiento	RF03
Nombre	Inicio de sesión
Propósito	Inicio de sesión de los usuarios
Descripción	Una vez el usuario se haya registrado, podrá ingresar al sistema y este mantendrá el inicio de sesión del usuario en el caso de haberlo realizado previamente
Entrada	Formulario de inicio de sesión
Salida	Redireccionamiento al tablero principal de la aplicación si los datos de ingreso son correctos, de lo contrario redirigir al inicio de sesión
Prioridad	Alta

Código Requerimiento	RF04
Nombre	Notificaciones agendamiento cita médica
Propósito	Notificar el agendamiento de una nueva cita médica
Descripción	Cuando el paciente haya agendado una cita médica con un profesional de la salud, la aplicación móvil para el profesional de la salud, la aplicación notificará de este hecho mediante correo electrónico
Entrada	Formulario de agendamiento de cita
Salida	Mensaje de confirmación y redirigir a vista previa.
Prioridad	Alta

Código Requerimiento	RF05
Nombre	Historial paciente
Propósito	Mantener un registro de las citas médicas realizadas

Descripción	Registrar la información relevante de las citas médicas realizadas y se pueda interactuar con ellas de manera: se pueda obtener más información, buscar cita médica y ordenar por cita médica más reciente
Entrada	Texto, selección del menú
Salida	Listado de las citas médicas realizadas
Prioridad	Media

Código Requerimiento	**RF06**
Nombre	Historial de citas médicas
Propósito	Visualizar de mejor manera las citas médicas agendadas
Descripción	Una sección que contiene las citas médicas agendadas, a manera de historial, con información relevante de la cita.
Entrada	Selección
Salida	Lista con las citas médicas agendadas
Prioridad	Media

Código Requerimiento	**RF07**
Nombre	Agendar cita médica
Propósito	Reservar una cita médica con un profesional de la salud
Descripción	El usuario registrado podrá reservar una cita médica del listado de citas médicas disponibles con el profesional de la salud de su preferencia. Una vez se haya agendado la cita médica, esta pasa de estado "disponible" a "ocupado"
Entrada	Selección
Salida	Mensaje de confirmación
Prioridad	Alta

Código Requerimiento	**RF08**
Nombre	Editar cita médica
Propósito	Permitir cancelar la cita médica.
Descripción	Permite que el usuario cancele una cita médica y en el caso de necesitarse, agendar una cita médica para otro día con disponibilidad
Entrada	Selección
Salida	Calendario de citas médicas e información relevante de esta
Prioridad	Media

Código Requerimiento	RF09
Nombre	Editar información usuario, paciente
Propósito	Actualizar información relevante
Descripción	Editar información del usuario relevante para el uso de la aplicación e ingreso a la misma
Entrada	Formulario de registro de usuario
Salida	Mensaje de confirmación
Prioridad	Alta

Código Requerimiento	RF10
Nombre	Editar información usuario, médico
Propósito	Actualizar información relevante
Descripción	Editar información del usuario relevante para el uso de la aplicación e ingreso a la misma
Entrada	Formulario de registro de usuario
Salida	Mensaje de confirmación
Prioridad	Alta

3.2.10. Casos de uso

En esta sección se utilizan diagramas UML (Unified Modeling Language) para visualizar secuencias de acciones en funcionalidades del sistema.

La Figura *3* describe el proceso de agendamiento de una cita médica mediante el asistente virtual.

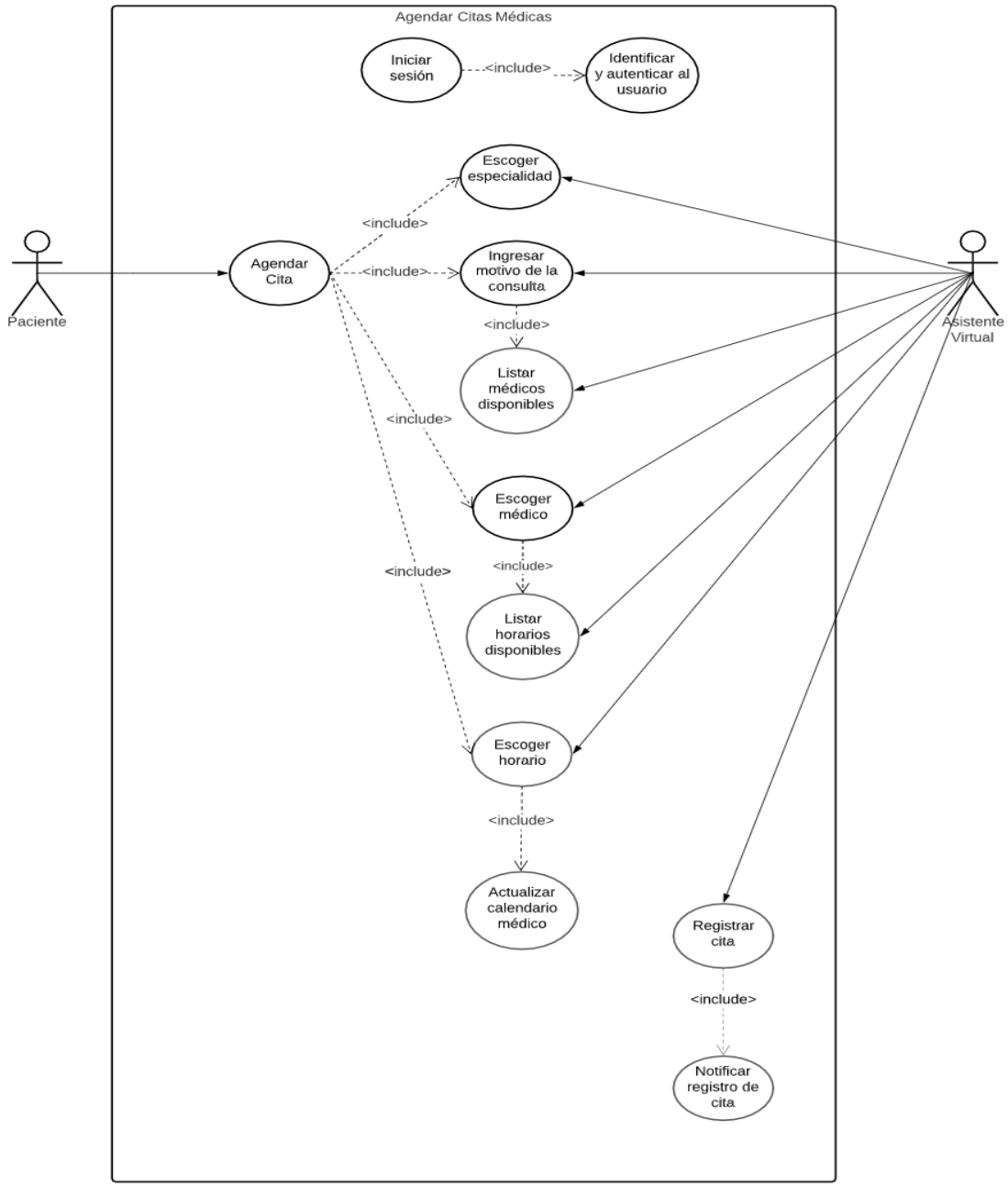


Figura 3. Diagrama agendamiento de citas médicas

En la Figura *4* se explica el proceso para registrar un nuevo usuario, independientemente si este

es un profesional de la salud o un paciente.

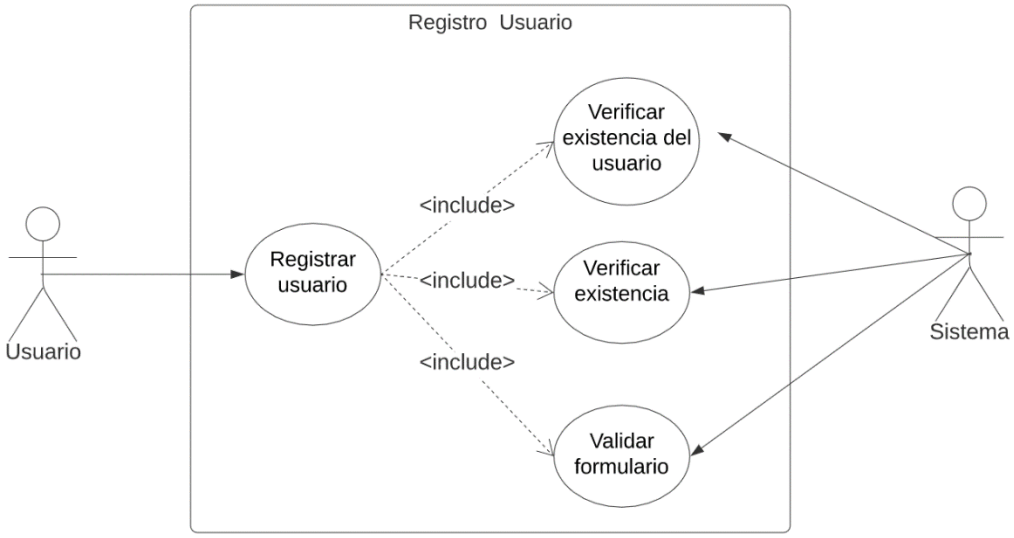


Figura 4. Diagrama registro de usuario

La Figura *5* ejemplifica el proceso de inicio de sesión del sistema.

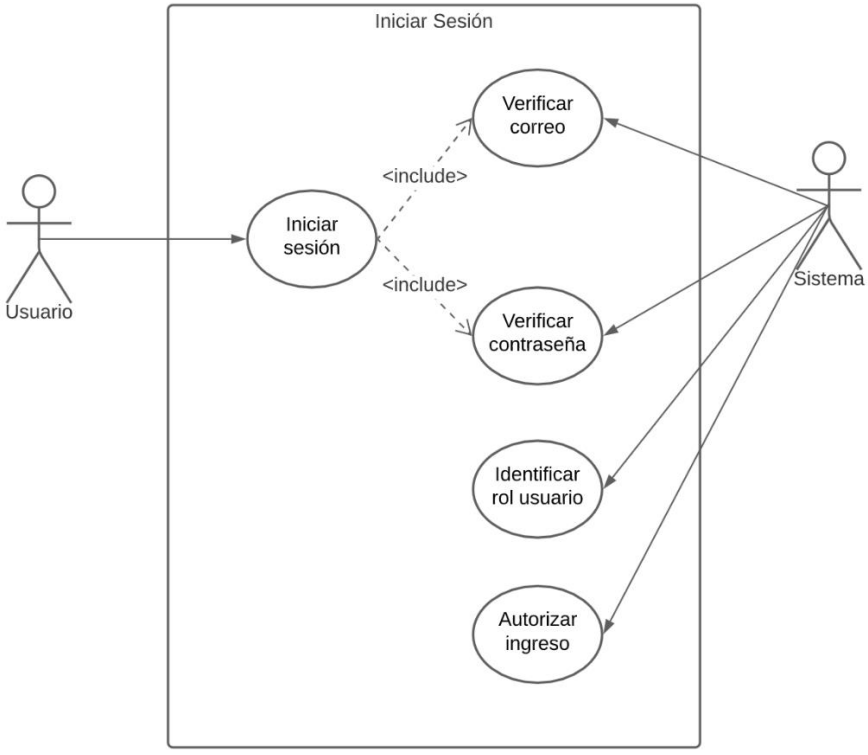


Figura 5. Diagrama iniciar sesión

La Figura *6* describe el proceso de ingreso de horarios de atención para una clínica registrada

por el profesional de la salud.

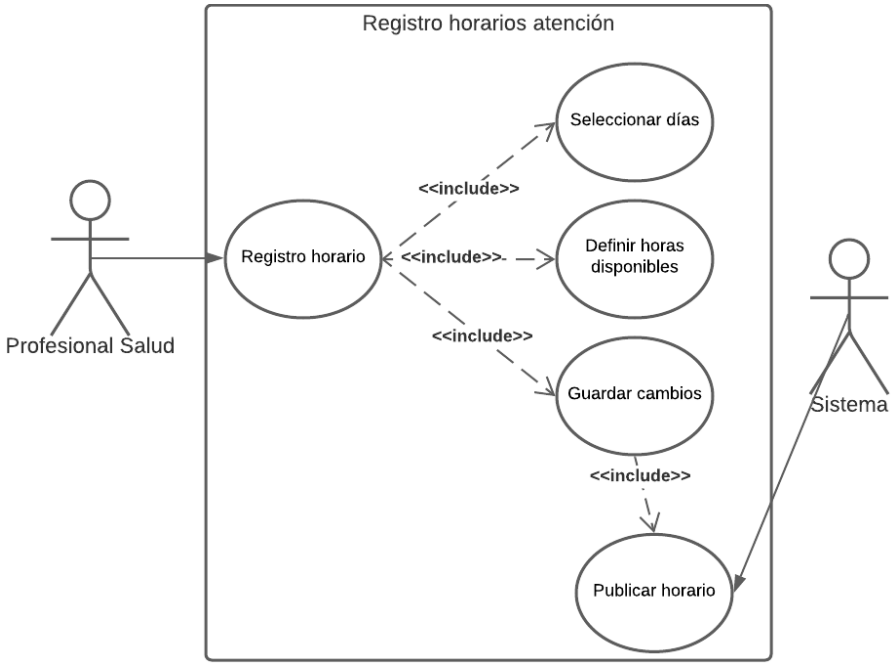


Figura 6. Diagrama registro horarios de atención

En la Figura *7* se describe el proceso de gestión de las citas, permitiendo ver las citas siguientes y agregar las citas atendidas y canceladas a un historial de citas médicas.

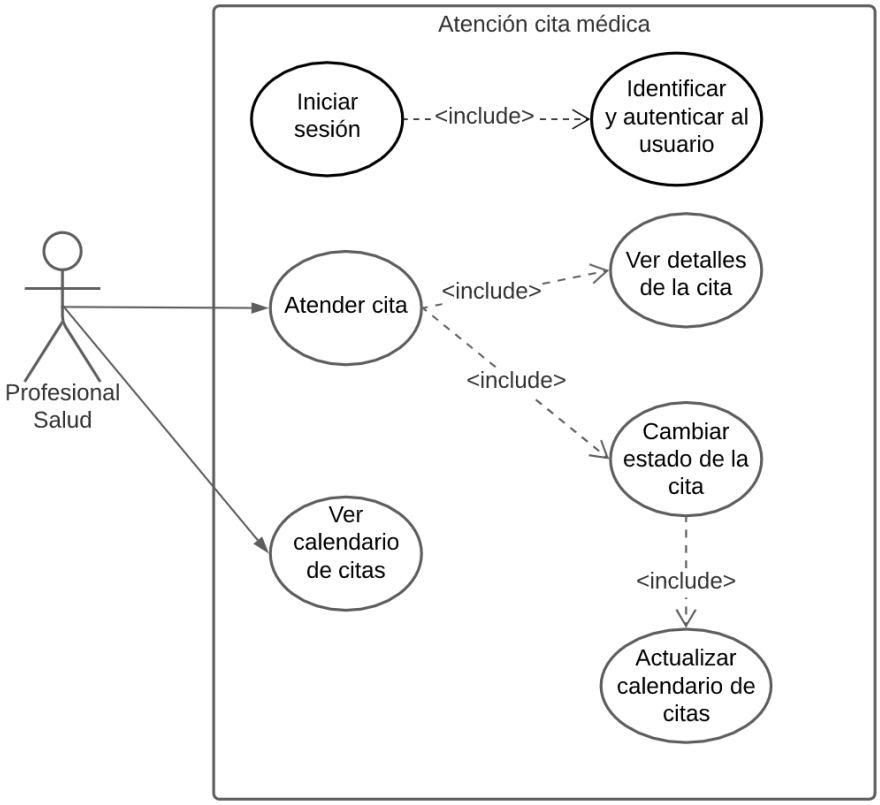


Figura 7. Diagrama atención cita médica

La Figura *8* explica el proceso de gestión de los usuarios registrados en el sistema, desde el registro, modificación y consulta de estos.

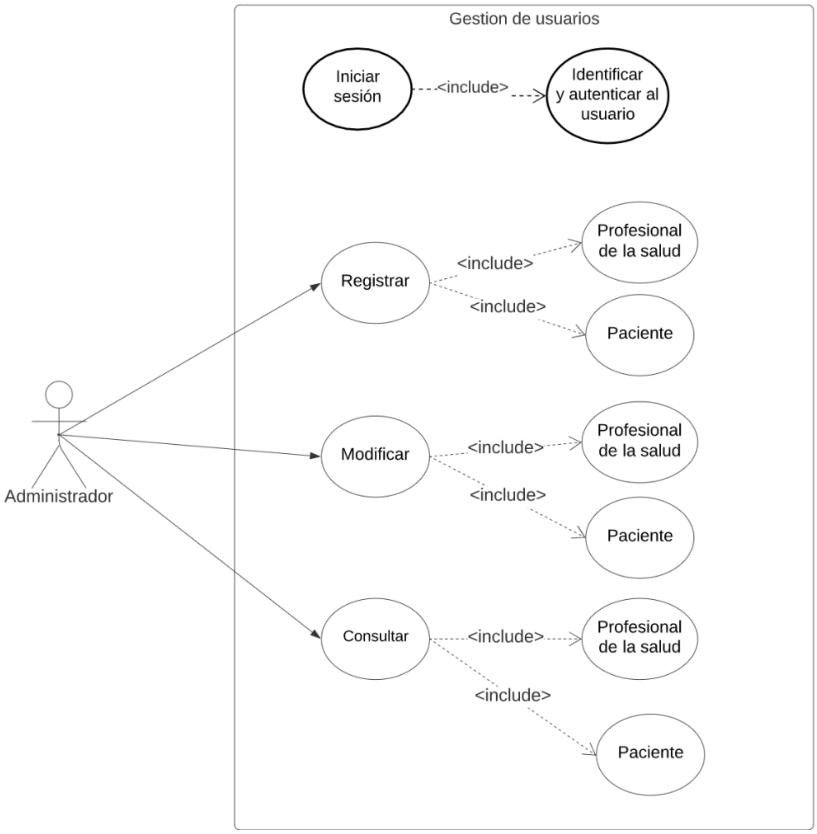


Figura 8. Diagrama gestión de usuarios

3.3. Diseño y arquitectura del Backend

En esta sección se describe la implementación de la capa de negocio y la capa de datos, iniciando con el diseño de la base de datos y la arquitectura de del Backend, este se fue desarrollado con Spring y el Java OpenJDK 11.

3.3.1. Capa de datos

Para la implementación de este proyecto se utilizó PostgreSQL en conjunto con Spring Data JPA.

3.3.2. Diagrama entidad relación

En esta sección se expone el diagrama entidad relación utilizado en este proyecto.

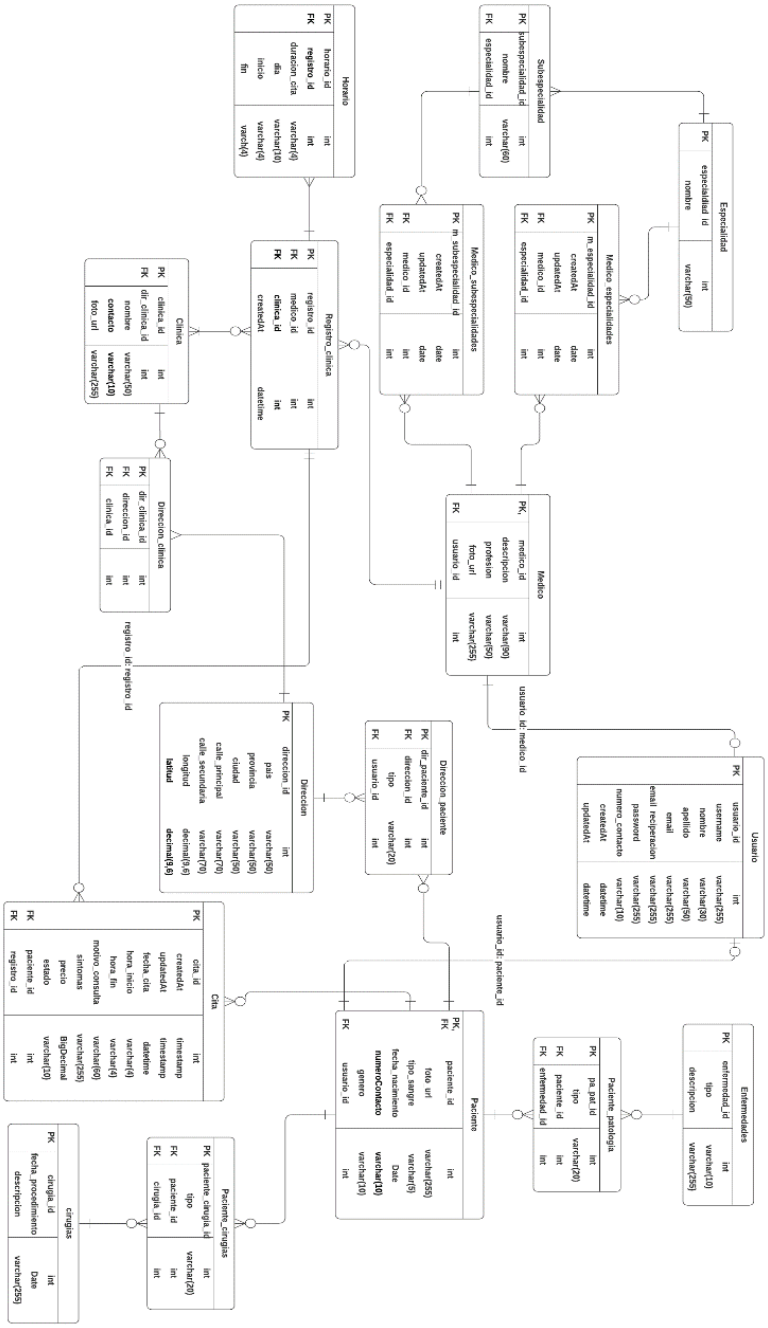


Figura 9. Diagrama entidad relación

Para la capa de datos se utilizó una base de datos PostgreSQL alojada en Amazon Web Services (AWS) haciendo el uso de Amazon Relational Database Service (RDS).

3.3.2. Capa de negocio

En esta sección se aborda la arquitectura de la capa de servicios y la interacción de la capa de

negocios con la capa de datos.

3.3.3. Arquitectura del Backend

La Figura 10 describe la división de los microservicios en privados y públicos, así también como la autenticación de los usuarios por tokens para acceder a servicios de escritura y lectura privados. En cuanto al registro y consulta de datos públicos, estos microservicios se encuentran en la parte pública.

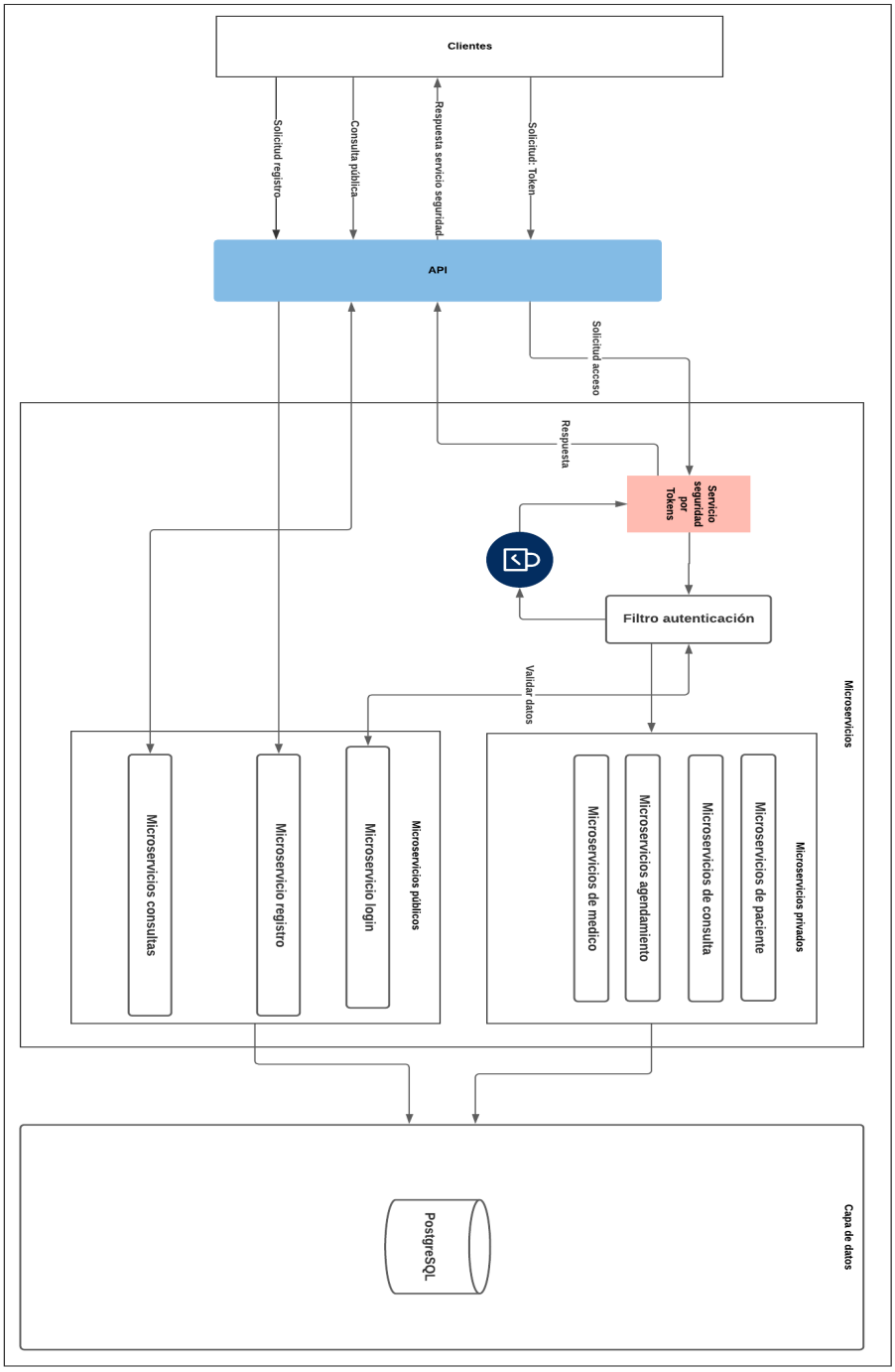


Figura 10. Arquitectura del Backend

En la Figura 10 se puede observar la interacción de con los clientes y la capa de negocios y datos. La capa de negocios contiene dos secciones: microservicios privados y microservicios públicos.

En el caso de los microservicios privados, dedicados a servicios de escritura y consulta de datos personales y necesarios para el agendamiento de citas médicas, estos pueden ser consultados solo con un token de autenticación proviste por filtro de seguridad por tokens.

Por último, la sección publica está dedicada al registro, inicio de sesión y consultas generales como: clínicas registradas, horarios de atención, fechas disponibles y médicos registrados; para esta última sección no es necesario solicitar un token de acceso.

Una vez se ha pasado por el filtro de seguridad, en el caso de los microservicios privados, se puede acceder a la información almacenada en la base de datos.

3.4. Desarrollo del asistente virtual

3.4.1. Descripción de las funcionalidades

Se creó una base de datos cumpliendo con los objetivos establecidos, se estudió la sintaxis y nomenclatura de Rasa para establecer las preguntas requeridas que hará el bot.

A parte de las preguntas requeridas el bot también implementa funciones para responder a saludos y despedirse, en caso de que el bot no comprenda el mensaje este responderá con una respuesta por defecto.

Las funcionalidades que se implementaron en el bot son las siguientes:

- El bot tiene la capacidad de responder a saludos y despedidas por parte del usuario.
- Es capaz de entender las preguntas o peticiones por parte del usuario al momento de solicitar el agendamiento de una cita médica.
- El bot muestra las repuestas a elegir con opciones de selección múltiple para mayor facilidad al momento de interactuar con este.
- Es capaz de repetir las preguntas en caso de que el usuario no haya ingresado la respuesta correcta, además de eso presenta un mensaje indicando el error que se ha cometido.
- Es capaz de comprender las respuestas que tengan faltas ortográficas como las tildes en ciertos términos.
- Al momento de terminar la recolección de los datos necesarios para el agendamiento de la cita el bot tiene la capacidad de preguntar al usuario si está seguro de agendar o no la cita.
- Cuando se ha agendado la cita médica el bot tiene la capacidad de mostrar todos los datos recolectados como retroalimentación para el usuario.

La base de datos utilizada contiene la información requerida para el agendamiento de citas como los médicos, clínicas y horarios disponibles, estos datos se muestran según las condiciones que desee el usuario.

3.4.2. Arquitectura Rasa

En la Figura 11. Arquitectura RasaFigura 11. Arquitectura Rasa se describe la arquitectura de Rasa, en ella se pueden observar los diferentes módulos del asistente virtual así también los componentes que conforman dichos módulos.

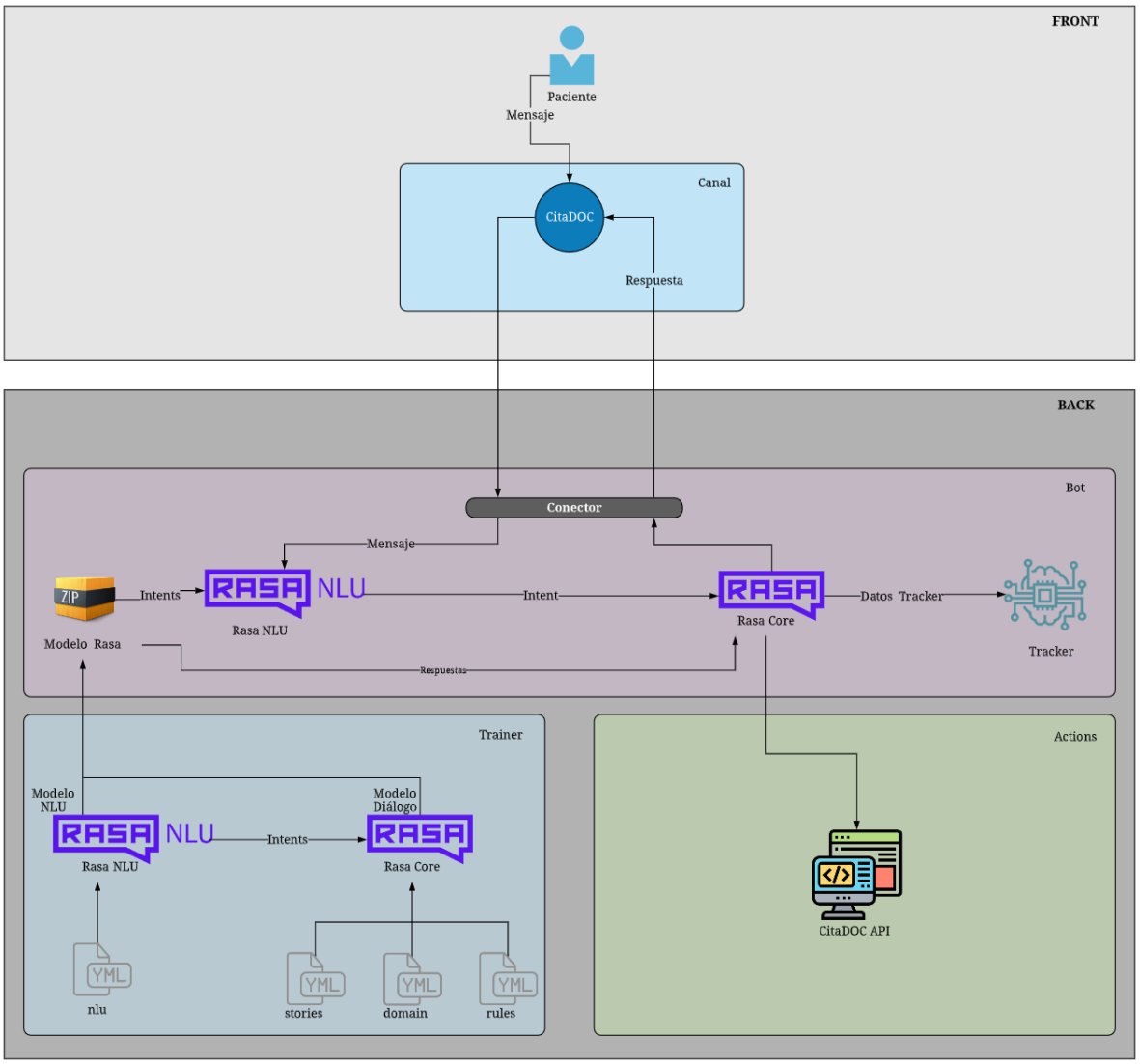


Figura 11. Arquitectura Rasa

Los módulos que componen el asistente son:

- Canal: es el medio por el cual el paciente interactúa con el bot, en este caso el canal es la aplicación desarrollada específicamente para el paciente.

- Bot: en este módulo se encuentra Rasa CLN (comprensión del lenguaje natural) con sus siglas en inglés NLU y Core los cuales se encargan de la gestión de la conversación lo cual explicaremos en la sección **Gestión de mensajes**, así también se interactúa con el *tracker* que no es más que la memoria del bot en donde están almacenados los datos que se hayan recolectado a lo largo de la conversación.

- Trainer: en este módulo se encuentran los modelos de conversación y CLN para la predicción de acciones y clasificación de intenciones que realizará el bot.

- Actions: en este módulo se definen las diferentes acciones personalizadas con las cuales se consumirán los servicios de la API CitadDOC para agendar las citas médicas.

3.4.3. Paquetes usados

En la Figura 12. Diagrama de paquetes se pueden observar los paquetes y librerías que se utilizaron para el desarrollo de nuestro asiste, así también se presentan las dependencias entre estos.

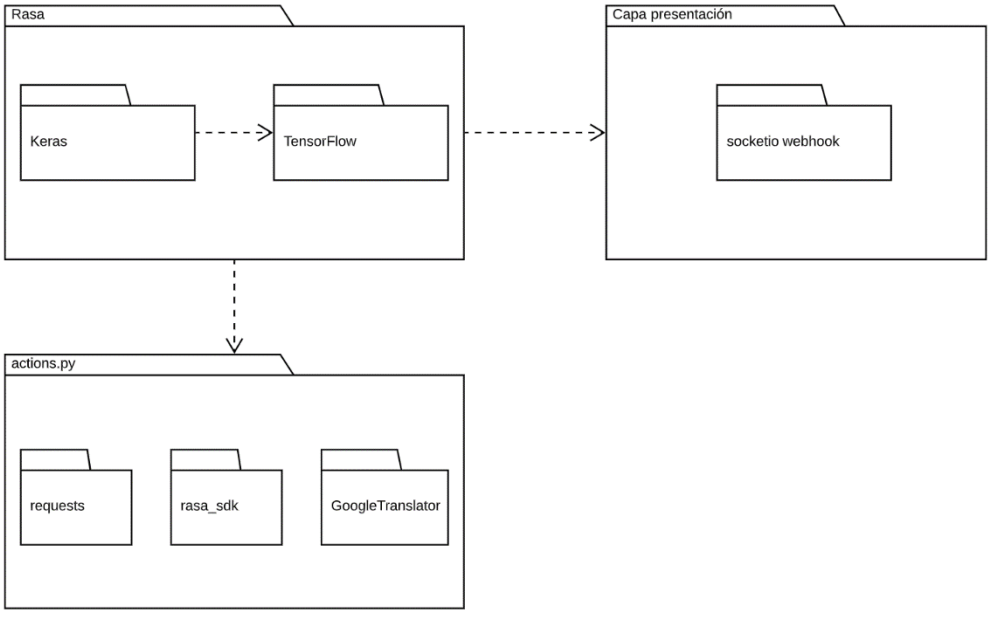


Figura 12. Diagrama de paquetes

La implementación del bot con RASA usa una red neuronal recurrente LSTM (Long Short-Term Memory), creada por defecto por Keras, dicha red se utiliza para seleccionar la acción correspondiente a ejecutar. Por otro lado, RASA usa TensorFlow para el entrenamiento del

asistente virtual. En cuanto a rasa_sdk, esta es una librería de Rasa que permite crear acciones personalizadas para servicios mediante el módulo Request, que permite realizar peticiones HTTP usando Python, brinda un formato el cual facilita el uso de parámetros en las peticiones para la interacción con sistemas externos, como lo es la aplicación del paciente; por último, GoogleTranslator para detectar y traducir textos que serán mostrados al usuario.

3.4.4. NLU datos de entrenamiento

El componente NLU (Natural Language Understanding) ayuda al asistente entender lo que solicita el usuario, cada mensaje que este envía se analiza y se clasifica con la intención correspondiente, en caso de que existan entidades estas se extraen del texto en forma de datos estructurados.

- **Entidades e intenciones**

  El usuario puede expresar sus intenciones con diferentes palabras a estas se las llaman utterances, por ejemplo, se puede expresar la intención de despedida de muchas maneras como: adiós, hasta luego, hasta pronto, etc. Algunos mensajes pueden contener entidades las cuales son necesarias de extraer para personalizar las respuestas, por ejemplo, si se define una intención para consultar los ingredientes de un tipo de pizza en particular la utterance podría ser "¿Qué ingredientes tiene la pizza ***{tipo_pizza}***?", el *tipo_pizza* es una entidad que se utilizará para buscar los ingredientes de ese tipo de pizza mediante alguna API externa.

- **Historias de entrenamiento**

  Las historias representan la conversación entre el usuario y el asistente por lo que forman parte del entrenamiento de los modelos para la gestión del dialogo.

  Estas historias se estructuran bajo un formato especifico en donde las entradas de los usuarios están representadas con las intenciones (*intent*) y entidades (*entity*) en caso de que existan, las respuestas del bot se representan como acciones (*action*).

  Las historias se definen en el archivo *stories.yml* dentro de la carpeta *data*, la historia *agendar cita médica con opciones múltiples* la cual representa la conversación entre el paciente y el bot para agendar una cita médica, está estructurada de la siguiente manera:

  - *story:* contiene el nombre de la historia.
  - *steps:* se definen los pasos o el proceso que tomara la conversación.
  - *intent:* corresponde a la intención que tiene el usuario, su estructura es la

siguiente *intent: nombre_intencion* en donde *nombre_intencion* corresponde a la intención que podría tener el usuario, por ejemplo, *intent: saludo.*

- *action:* representa la respuesta que dará el bot su estructura es la siguiente *action: utter_accion* en donde *utter_accion* corresponde a la acción que realizara el bot, por ejemplo, *action: utter_saludo* con esto el bot estaría saludando al usuario, también se pueden llamar diferentes acciones en este caso se activa un formulario con el cual se recolectan los datos necesarios para agendar una cita médica.
- *active_loop:* se utiliza para iniciar y finalizar un formulario, para iniciar o activar un formulario se lo realiza de la siguiente manera *active_loop: cita_form* en donde *cita_form* es el formulario ya definido en el dominio.
- *slot_was_set:* dentro de este se define la entidad que deberá ser ingresada en el *solt* como así también se definen las posibles entradas de dicha entidad, por ejemplo, para la entidad *síntoma* se pueden realizar las siguientes entradas "*mareos, dolor de estómago*".
- *requested_slot:* indica que dicho slot debe ser ingresado de manera obligatoria por lo que tiene validación de formulario lo que indica que en caso de que se ingrese un valor erróneo no permitirá continuar con este hasta que se ingrese el valor correcto de la entidad, dichas validaciones se definen en las acciones personalizadas.

Cuando ya se hayan ingresado todos los slots del formulario este se desactiva y se realiza el agendamiento de la cita médica con los datos obtenidos, finalmente el bot llama a la acción *action_restart* para reiniciar la conversación con esto todos los valores de los slots se eliminan para poder iniciar una conversación nueva.

En la Figura 13: Representación visual de la historia se puede observar la representación visual de la historia antes descrita.

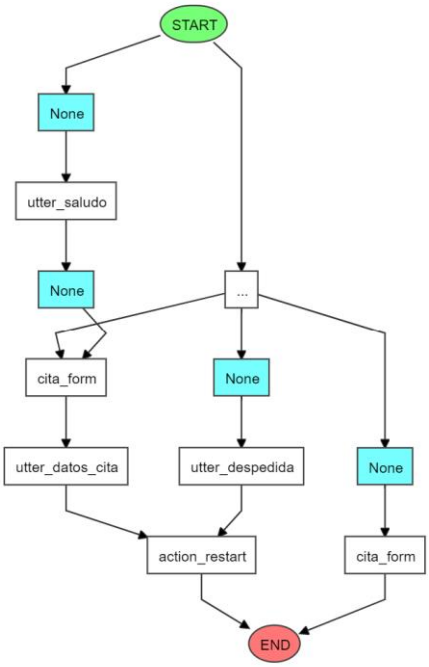


Figura 13: Representación visual de la historia

- **Acciones**

  Las acciones se definen en el archivo *actions.py*, estas son funciones escritas en Python las cuales ayudan a implementar una mayor personalización ya que se pueden conectar a otros servicios permitiendo aumentar las funcionalidades del bot.

  La estructura de una acción personalizada es la siguiente.

  1. Se define una clase del tipo (*Action*).
  2. La función *name* retorna el nombre de la acción siguiendo la nomenclatura de Rasa, la cual es *action_nombre_accion*, que en nuestro caso el bot solo implementará acciones de preguntas por lo que la nomenclatura se define de la siguiente manera *action_ask.*
  3. En la función *run* va la lógica necesaria para la personalización del bot.
  4. Para que el bot envie un mensaje al usuario lo realiza a través de *dispatcher* usando la función *utter_message.*
  5. Finalmente se retorna un conjunto vacío.

  Lista de acciones definidas:

  - *action_ask_sintoma:* pregunta los síntomas al paciente.
  - *action_ask_dia:* devuelve los días en los que existen horarios disponibles.
  - *action_ask_especialidad:* devuelve las especialidades disponibles según el día elegido.

- *action_ask_medico:* devuelve los médicos disponibles según la especialidad elegida.
- *action_ask_clinica:* devuelve las clínicas disponibles por el medico elegido.
- *action_ask_fecha:* devuelve los horarios disponibles según el día, el médico y clínica elegidos.
- *action_ask_confirmacion:* pregunta al usuario si está seguro de agendar la cita médica.

También se define una clase del tipo (*FormValidationAction*) la cual se encargará de validar los valores de cada uno de los slots del formulario.

- **Dominio**

  Todo el ambiente de rasa se define en el archivo *domain.yml* el cual se encuentra en la carpeta raíz del proyecto, en este se especifican las intenciones, entidades, slots, acciones y formularios, este archivo se estructura de la siguiente manera:

1. *Intents:* se especifican las intenciones estas deben coincidir con el archivo *nlu.ym*

2. *Entities:* se especifican las entidades que se hayan definido en el archivo *nlu.yml*.
3. *Actions*: se especifican las acciones definidas en el archivo *actions.py*, si estas no se especifican en el archivo del dominio no podrán ser usadas para el entrenamiento del modelo.
4. *Responses:* se definen las respuestas del bot, estas son opcionales ya que se puede responder usando las acciones personalizadas. Algo a tener en cuenta es la nomenclatura de Rasa, cada respuesta debe comenzar con *utter* o en caso de que el bot deba realizar una pregunta *utter_ask.*

5. *Slots*: estos son la memoria del bot y se utilizan como variables para almacenar los datos de la conversación, en la **¡Error! No se encuentra el origen de la referencia.** se pueden observar algunos de los slots usados para almacenar los datos para el agendamiento de citas médicas.

6. *Forms*: finalmente se definen los formularios que se utilizarán en la conversación para el asistente solo se usó uno, primero se debe definir el nombre del formulario en nuestro caso es *cita_form*, luego se especifica la lista de slots que serán ingresados de manera obligatoria bajo la declaración *required_slots*, cada slot del formulario tomará el valor de la entidad correspondiente, cierto valor será extraído de la intención del usuario.

- **Reglas**

  Las reglas se definen en el archivo *rules.yml*, el cual se encuentra dentro de la carpeta *data*, estos datos se utilizan para el entrenamiento del modelo de gestión de dialogo del bot, estas reglas describen pequeñas conversaciones que siempre siguen los mismos pasos, antes de la implementación de las reglas se debe agregar la política de reglas (*RulePolicy*) en el archivo *config.yml.*

  Su estructura es similar a la de las historias, se tiene que definir el nombre de la regla y los pasos que se deben de seguir para que esta se cumpla, cada regla se comienza con una intención que inicia una conversación y luego se agregan las acciones pertinentes que debe realizar el bot.

  En nuestro caso se definieron tres reglas:

  1. El bot se despide cuando detecta la intención de *despedida* por parte del usuario y reinicia la conversación.
  2. El bot activa el formulario para recolectar los datos necesarios para agendar la cita médica cuando este detecta la intención de *agendar*.
  3. El bot envía el formulario una vez que se hayan ingresado correctamente todos los slots.

3.4.5. Configuración

La configuración general del asistente se realiza en el archivo *config.yml* el cual se divide en dos partes: las políticas para Rasa Core las cuales se explicarán más adelante y los compontes para la canalización (pipeline) de Rasa NLU. También se especifica el código de idioma *ISO* de dos letras que interpretará el bot en nuestro caso español (*es*).

La configuración o componentes de canalización para Rasa NLU usados son los siguientes:

- *WhitespaceTokenizer:* se encarga de extraer las entidades del texto mediante las funciones que este crea, procesa idiomas en los que sus palabras están separada por un espacio en blanco.

- *RegexFeaturizer:* ayuda a identificar intenciones y entidades para crear un conjunto de datos en formato de Rasa para el entrenamiento.
- *LexicalSyntacticFeaturizer: "*se encarga de extraer y codificar los rasgos sintácticos y léxicos". (Rasa Technologies GmbH, 2021).
- *CountVectorsFeaturizer:* crea una secuencia de conteo de tokens basados en CountVectorizer de sklearn, se usa para entrenar solo los datos que nosotros

ingresemos, funciona para los idiomas en los que las palabras estén separadas por espacios.

- *analyzer: char_wb: "*crea n-grmas (conjunto de n elementos en un texto) de caracteres en base a un texto dado dentro de un intervalo de palabras" (Rasa), también se utiliza para crear *Subword Semantic Hashing* que es muy útil para la clasificación de intenciones en un conjunto de datos. (Shridhar et al., 2019)
- *min_ngram y max_ngram:* se definen los límites de los n-gramas.
- *DIETClassifier:* Transformador de doble intención y entidad, se utiliza para clasificar intenciones e identificar entidades de manera conjunta gracias a su arquitectura de transformador. (Rasa Technologies GmbH, 2021)
- *epochs:* se indica el número de veces que se ejecutara el algoritmo *DIETClassifier* para el entrenamiento de los datos.
- *model_confidence:* se define como se calcularán los valores de confianza durante la inferencia. (Rasa Technologies GmbH, 2021)
- *linear_norm:* indica el rango de valores de confianza, en este caso están entre [0, 1].

3.4.6. Políticas

Rasa Core utiliza políticas para la decidir qué acciones se deben tomar a lo largo de la conversación, existen políticas de aprendizaje de maquina y de reglas que el bot debe usar, estas políticas se definen en el archivo *config.yml* de nuestro proyecto.

Las políticas para Rasa Core usadas son las siguientes:

- *MemoizationPolicy:* esta política memoriza todas las conversaciones en los datos de entrenamiento, luego comprueba si la conversación actual tiene alguna similitud con las historias del set de datos, dado que exista alguna similitud, el bot podrá anticipar cual es la siguiente acción que deberá tomar según la historia de sus datos de entrenamiento con un valor de confianza del 1.0, caso contrario si no existe ninguna historia similar se predice Ninguna con un valor de confianza de 0.0.

- *TEDPolicy:* La política del diálogo de integración de transformadores, el bot utiliza esta política para escoger la siguiente acción a seguir, TED ayuda a decidir qué acciones se deben ignorar o seleccionar mediante su arquitectura transformadora.

- *max_history:* define el número de historias que se analizan para predecir la siguiente accion a tomar.

- *epochs:* indica el número de veces que el algoritmo examinará los datos de

entrenamiento.

- *constrain_similarities: "*al tomar el valor *true* aplica la perdida de entropía cruzada sigmoidea" (Rasa), mantiene la similitud entre valores pequeños y etiquetas negativas con el objetivo de generalizar el modelo a todos los datos de prueba del mundo real.

- *RulePolicy:* la política de reglas permite definir condiciones, activar o desactivar formularios y especificar acciones que se deben elegir cuando se detecta alguna intención en particular, por ejemplo, el bot siempre saludará cuando se detecte la intención de *saludar* por parte del usuario.

3.4.7. Entrenamiento

Nuestro bot usará una red neuronal del tipo LTSM creada con Keras y entrenada con TensorFlow para predecir la acción con mayor porcentaje de confianza a realizar.

Rasa brinda la posibilidad de entrenar los modelos de NLU y Core por separado con los comandos *rasa train nlu* y *rasa train core*, en nuestro caso se usó *rasa train* el cual entrena tanto NLU y Core. Una vez terminado el entrenamiento del modelo este se guarda dentro de la capeta *models*, si no se define un nombre para el modelo este se guarda como *"<timestamps>.tar.gz.*

3.4.8. Conectores

Los conectores ayudan a que bot se conecte con los canales de entrada y salida de datos, en Rasa existen cuatro canales predefinidos:

- Facebook messenger
- Slack
- Mattermost
- Web
- Telegram
- Google Hangouts Chat
- Microsoft Bot Framework
- Cisco Webex Teams
- RocketChat
- Mattermost
- Conectores personalizados

Para usar cada uno de los conectores se debe agregar las credenciales correspondientes en el archivo *credentials.yml*, cabe recalcar que no todos los conectores tendrán los mismos campos para las credenciales ya que le número de tokens de acceso podrían variar.

El canal que usamos es *socketio*, este usa websockets y puede enviar y recibir mensajes en tiempo real, este canal se usa para conectar con nuestra propia página web en nuestro caso la aplicación móvil para el paciente en donde desarrollamos nuestro propio chat para que el usuario se comunique con el bot.

Los valores *user_message_evt* y *bot_message_evt* definen los nombres de eventos utilizados por Rasa Open Source al enviar o recibir mensajes a través de socket.io (Rasa).

3.4.9. Gestión de mensajes

En la Figura 14. Gestión de mensajes se puede observar cómo se realiza la gestión de mensajes mediante NLU y Core.

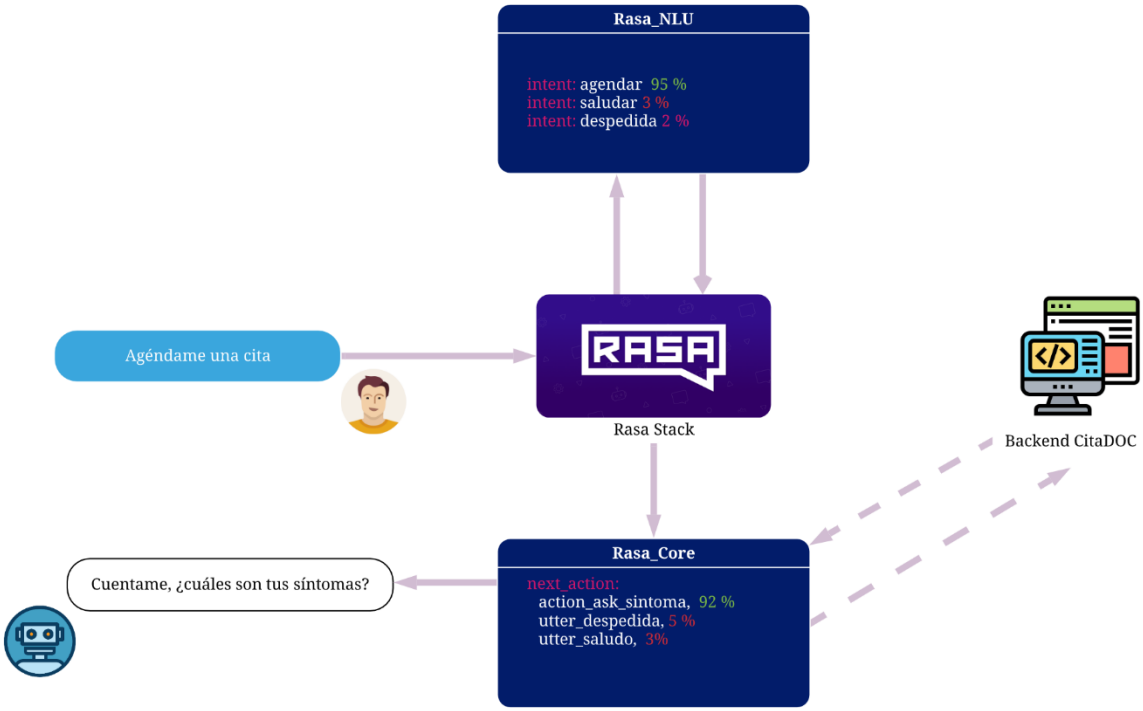


Figura 14. Gestión de mensajes

1. Los mensajes del usuario son enviados a *Rasa Stack (NLU+Core)*, en este caso el mensaje "*Agéndame una cita*", el stack se encarga de procesar dicho mensaje y gestiona la comunicación de las diferentes capas.

2. Rasa NLU se encarga de descifrar el mensaje para extraer las entidades en forma de

datos estructurados y recuperar las respuestas para posteriormente clasificar estos datos en la intención correspondiente, como se puede observar cada intención tiene un porcentaje de aceptación, el % mínimo para que una intención sea confiable es del 40% (dicho porcentaje puede ser modificado) caso contrario el bot solicitará nuevamente al usuario que ingrese el valor correcto presentando un mensaje de error.

3. Rasa Core toma la intención (intent) clasificada por Rasa NLU y la analiza para posteriormente realizar las acciones pertinentes según las historias y reglas definidas para el entrenamiento de este. Se toma la opción que mayor porcentaje de confianza tenga y en base a eso se realiza la acción pertinente como activar formularios o consultar APIs externas según la lógica que se haya definido, en nuestro caso se activa un formulario y se conecta a los servicios de CitaDOC para extraer los datos necesarios para el agendamiento de las citas médicas.

4. Para finalizar el proceso, el bot retorna el mensaje de respuesta al usuario usando el canal pertinente.

3.4.10. API Rasa

Para conectar la aplicación del paciente y el asistente se utilizó la API HTTP de rasa, la cual permite enviar y recibir mensajes al bot realizando llamadas del tipo POST con los parámetros correspondientes, de esta manera se realiza la recolección de los datos necesarios del paciente para el agendamiento de la cita médica.

4. Resultados

Aquí se presentan los resultados obtenidos del desarrollo del chatbot y su implementación con el sistema para el agendamiento de citas médicas.

4.1. Interfaces aplicación del médico

La aplicación del médico contiene siete funcionalidades importantes: Login, registro, citas, historial, clínicas, datos personales y académicos.

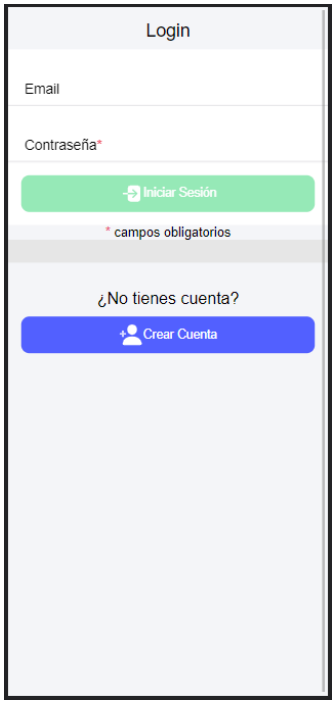


Figura 15. Login de médico

La Figura 15. Login de médico muestra la vista de inicio de sesión del profesional de la salud.

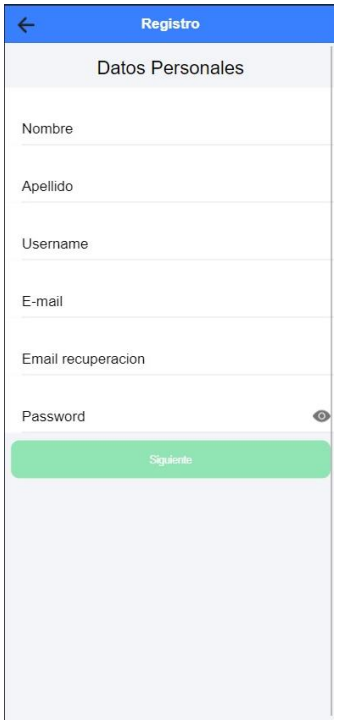


Figura 16. Primer formulario de registro de usuario

La Figura 16Figura 16. Primer formulario de registro de usuario es la primera parte del formulario de registro del profesional de la salud, este formulario deshabilita el botón

de avanzar con la parte final del registro hasta que los campos sean validados y completados.

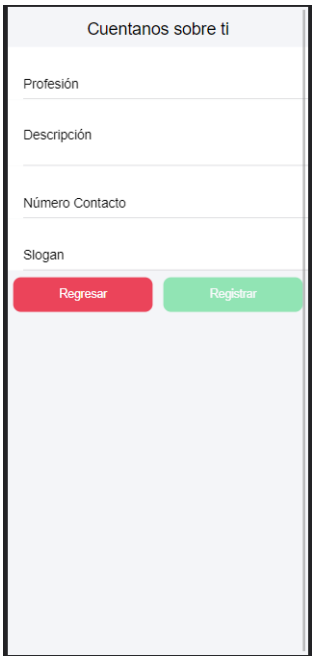


Figura 17. Segunda parte formulario de registro

La segunda parte, Figura 17 del formulario de registro está centrada en los datos descriptivos y académicos del profesional de la salud. Una vez todos los campos de los dos registros hayan sido validados se procede con el registro del nuevo profesional de la salud.

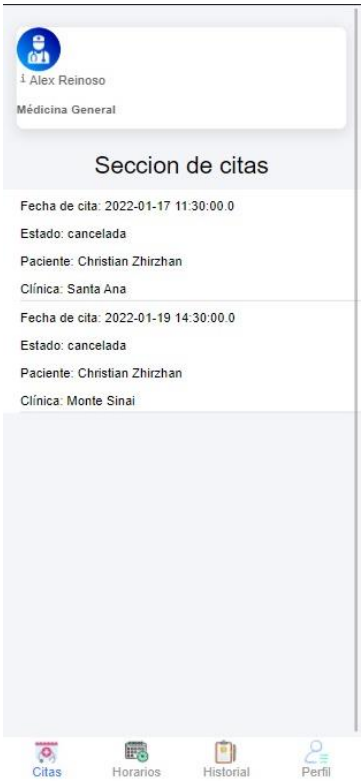


Figura 18. Interfaz principal del profesional de la salud

La Figura 18 muestra la interfaz de la sección de citas, en esta interfaz se muestran las citas agendadas a partir del día de consulta, así también como información relevante sobre estas.

Figura 19. Historial de citas agendadas

La Figura 19 contiene el historial de todas las citas ordenadas de manera descendente por fecha de agendamiento de la cita médica.

La sección de clínicas se encuentra divida en 2 secciones, en la primera se presentan las clínicas registradas por el profesional de la salud y contiene una seccion apartada para el registro de una nueva clínica.

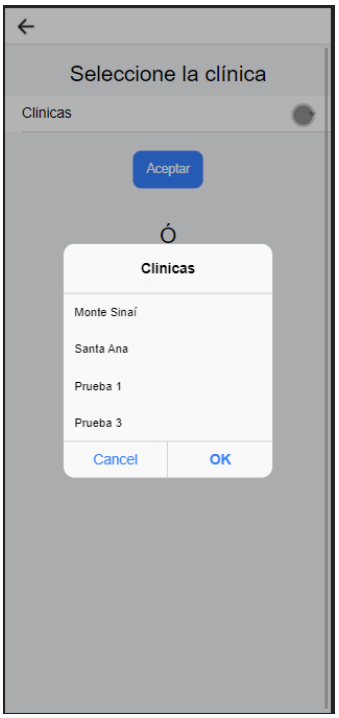


Figura 20. Selección de clínicas registradas

En la Figura 20 se muestran las clínicas disponibles, todavía no registradas por el profesional de la salud, para agregar a la lista de cínicas en donde labora el profesional de la salud.

En el caso de que la clínica que se desee registrar no se encuentre en la lista de clínicas disponibles, se permite al profesional de la salud registrar una nueva clínica. Este formulario se encuentra dividido en dos secciones: datos de clínica y dirección de esta.

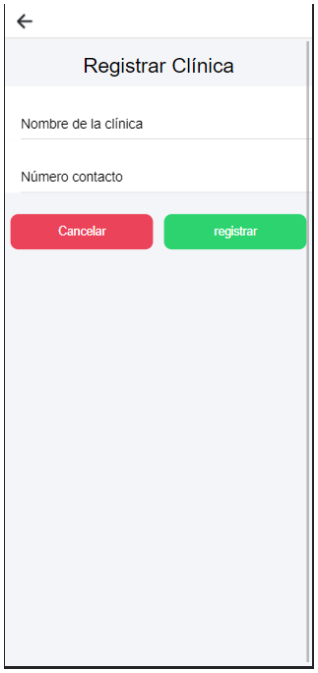


Figura 21. Formulario registro nueva clínica

En la Figura 21. Formulario registro nueva clínica se muestra la interfaz de registro de la clínica, en esta se puede agregar el nombre y número de contacto de esta. Esta es la primera parte del registro de una nueva clínica.

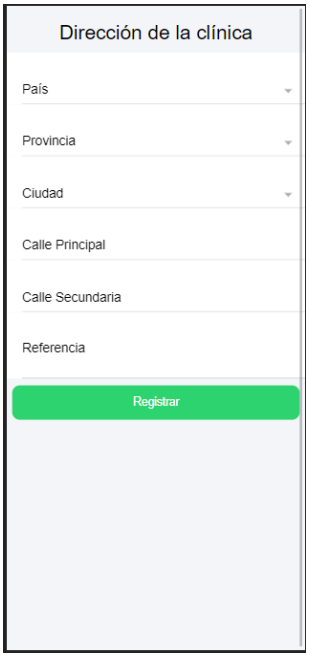


Figura 22. Registro dirección de clínica

La Figura 22 contiene la última parte del registro de una nueva clínica, en este formulario se ingresa la información de la dirección de la clínica a registrar.

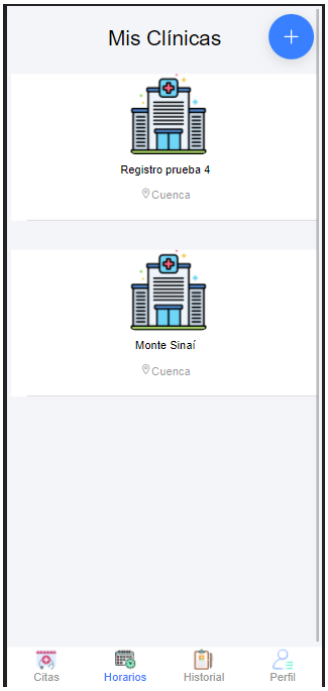


Figura 23. Interfaz de clínicas registradas

En la Figura 23. Interfaz de clínicas registradas se puede observar las clínicas registradas por el profesional de la salud. Al seleccionar una de estas podemos acceder a la interfaz para agregar el horario de atención de cualquiera de estas.

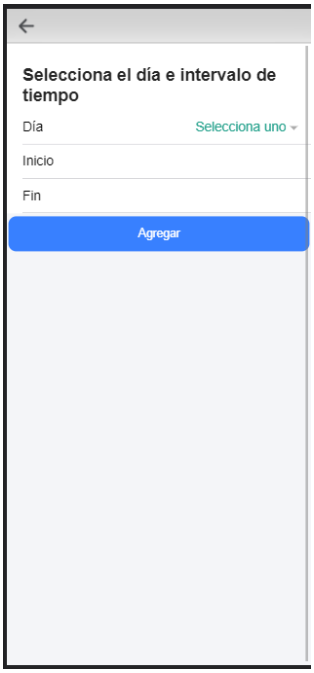


Figura 24. Interfaz agregar horario de atención

En la Figura 24. Interfaz agregar horario de atención se puede observar la manera de ingresar un horario de atención para una clínica seleccionada, se elige el día, la hora de inicio y la hora de fin de atención de citas médicas.

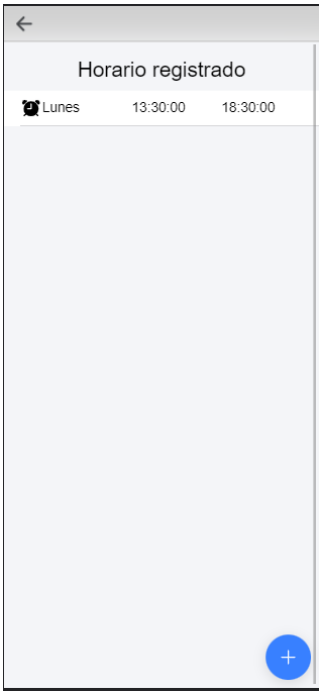


Figura 25. Listado de horarios registrados

En la **¡Error! No se encuentra el origen de la referencia.** Figura 25**¡Error! No se encuentra el origen de la referencia.** se muestra la interfaz de horarios registrados, de igual manera contiene el botón de acceso a la interfaz de agregar un nuevo horario de atención de citas médicas. En el caso de desear eliminar un horario registrado, se debe deslizar de derecha a izquierda sobre el horario a eliminar.

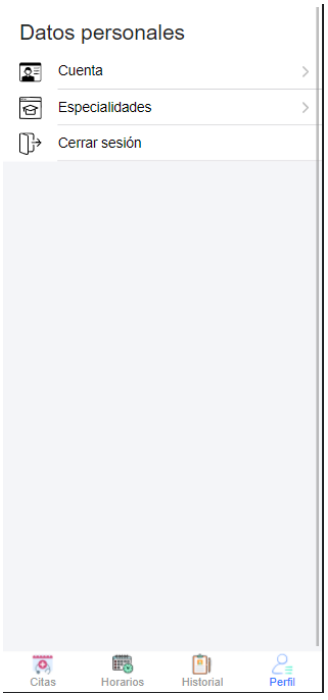


Figura 26. Datos académicos y presentación del médico

En la Figura 26 se muestra el acceso a las vistas para agregar datos académicos como: especialidades y subespecialidades; y, modificar datos de la cuenta como: slogan, descripción y numero de contacto.

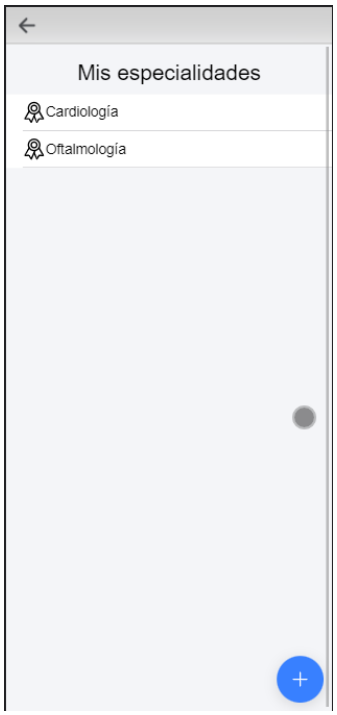


Figura 27. Interfaz de especialidades médicas

En el caso de la Figura 27 para el agregado de especialidades este funciona de igual manera que el agregado de clínicas, una vez se ha registrado una especialidad de la lista de especialidades registradas, está ya no se muestra como disponible al menos que se elimine esta. Por otro lado, en el caso de desear agregar una subespecialidad, se seleccionar la especialidad y se muestran las subespecialidades todavía no registradas por el profesional de la salud.

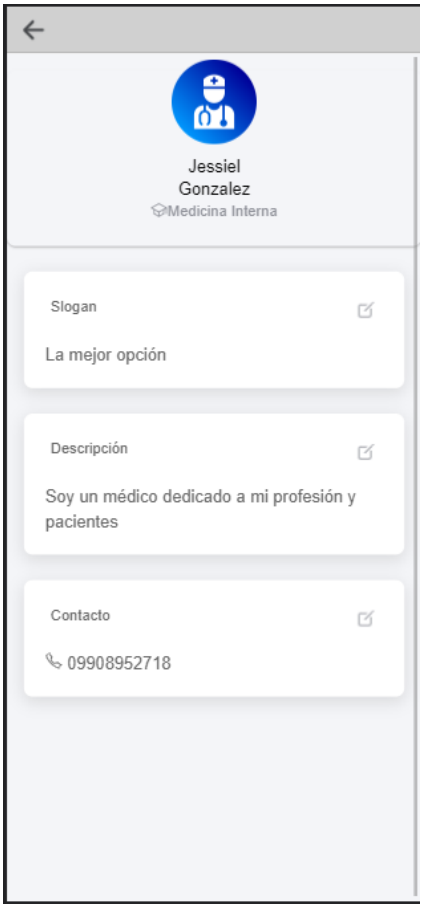


Figura 28. Editar datos de presentación

En la Figura 28**¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.** se pude ver la vista para editar el slogan, descripción y número de contacto del profesional de la salud.

4.2. Interfaces aplicación del paciente

Como se había indicado en los objetivos, se debe desarrollar una aplicación móvil para que un paciente agende las citas médicas mediante el uso de un chatbot, a continuación, se presentan las diferentes figuras en donde se pueden observar las pantallas de la aplicación desarrollada.

Figura 29. Pantalla inicio de sesión paciente

En la Figura 29. Pantalla inicio de sesión paciente se muestra la pantalla de inicio de sesión del paciente, en donde deberá ingresar su correo electrónico y contraseña para ingresar al sistema.

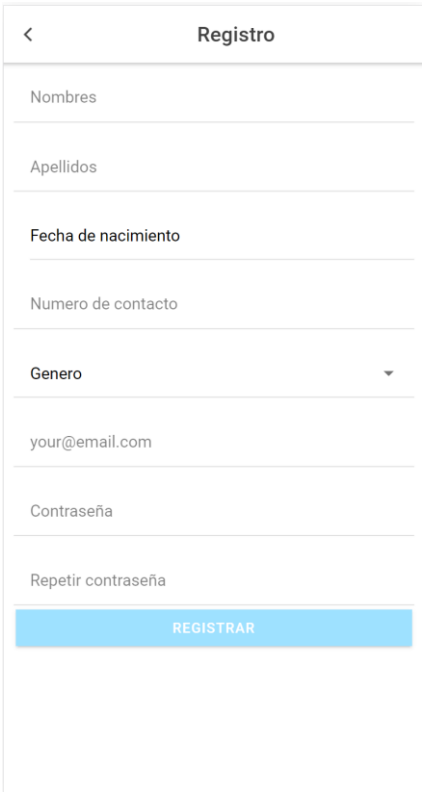


Figura 30. Pantalla registro de usuario

En la Figura 30. Pantalla registro de usuario se muestra la pantalla para el registro del paciente, en esta se observan los campos que deberá ingresar el usuario para registrarse.

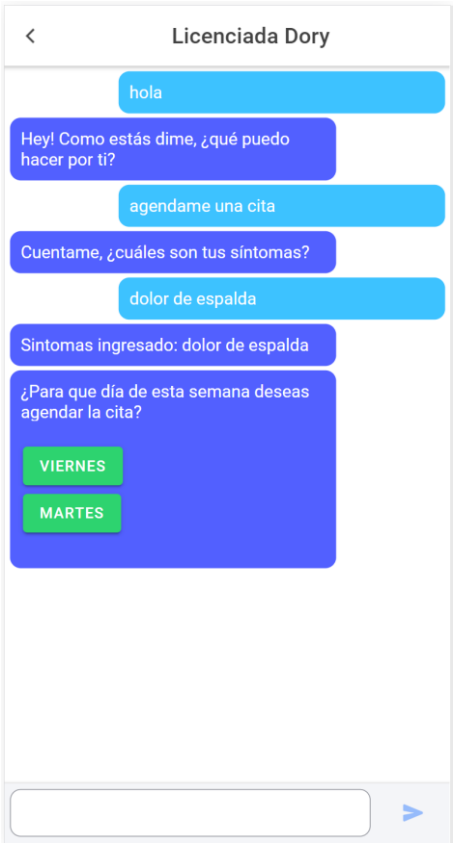


Figura 31. Pantalla Chatbot

En la Figura 31. Pantalla Chatbot se muestra la pantalla del chat con el bot Dory, en donde el usuario podrá interactuar con el asistente para agendar las citas médicas.

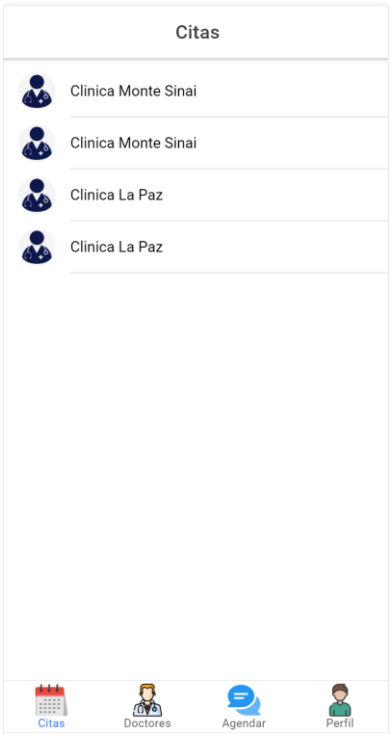


Figura 32. Pantalla lista de citas agendadas

En la Figura 32. Pantalla lista de citas agendadas se muestra la pantalla en donde se listan todas las citas agendadas del paciente, estas se muestran en un principio con el nombre de la clínica en donde se agendó la cita.

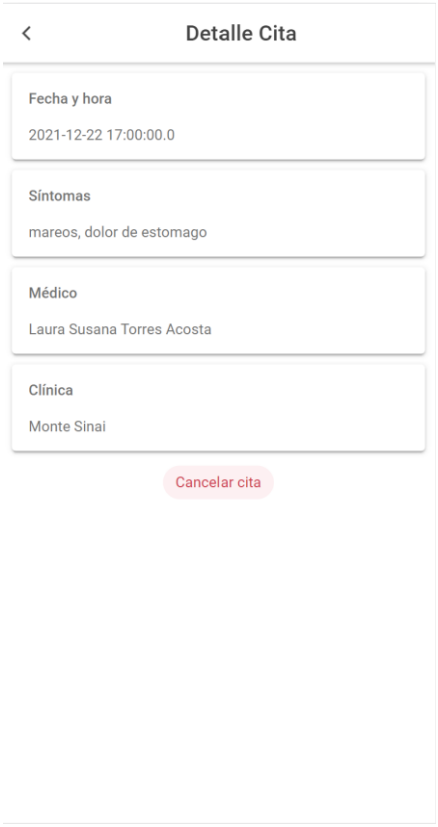


Figura 33. Pantalla datos de la cita médica

En la Figura 33. Pantalla datos de la cita médica se muestra la pantalla se muestran los datos de la cita agendada, así como también la opción de cancelar la misma.

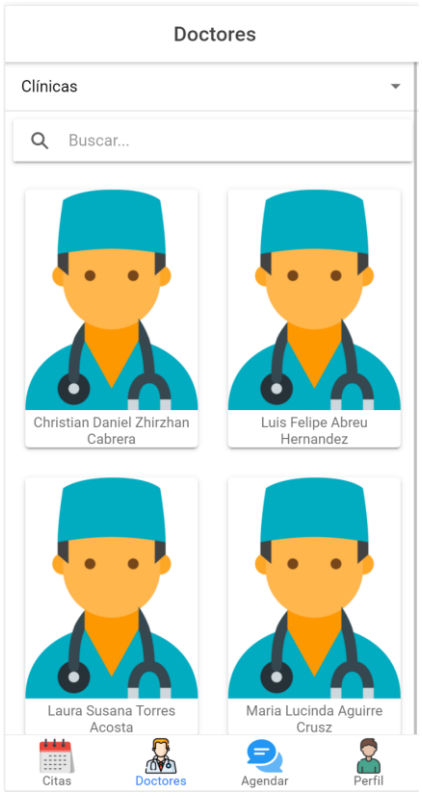


Figura 34. Pantalla listado de médicos

En la Figura 34. Pantalla listado de médicos se puede observar la pantalla en donde se muestran los médicos disponibles, estos podrán ser filtrados mediante las clínicas en las que trabajan y también pueden ser consultados mediante la barra de búsqueda implementada.

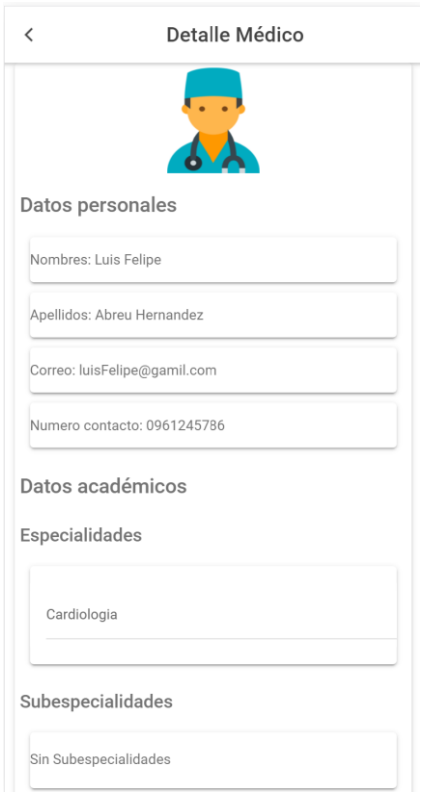


Figura 35. Pantalla datos del médico

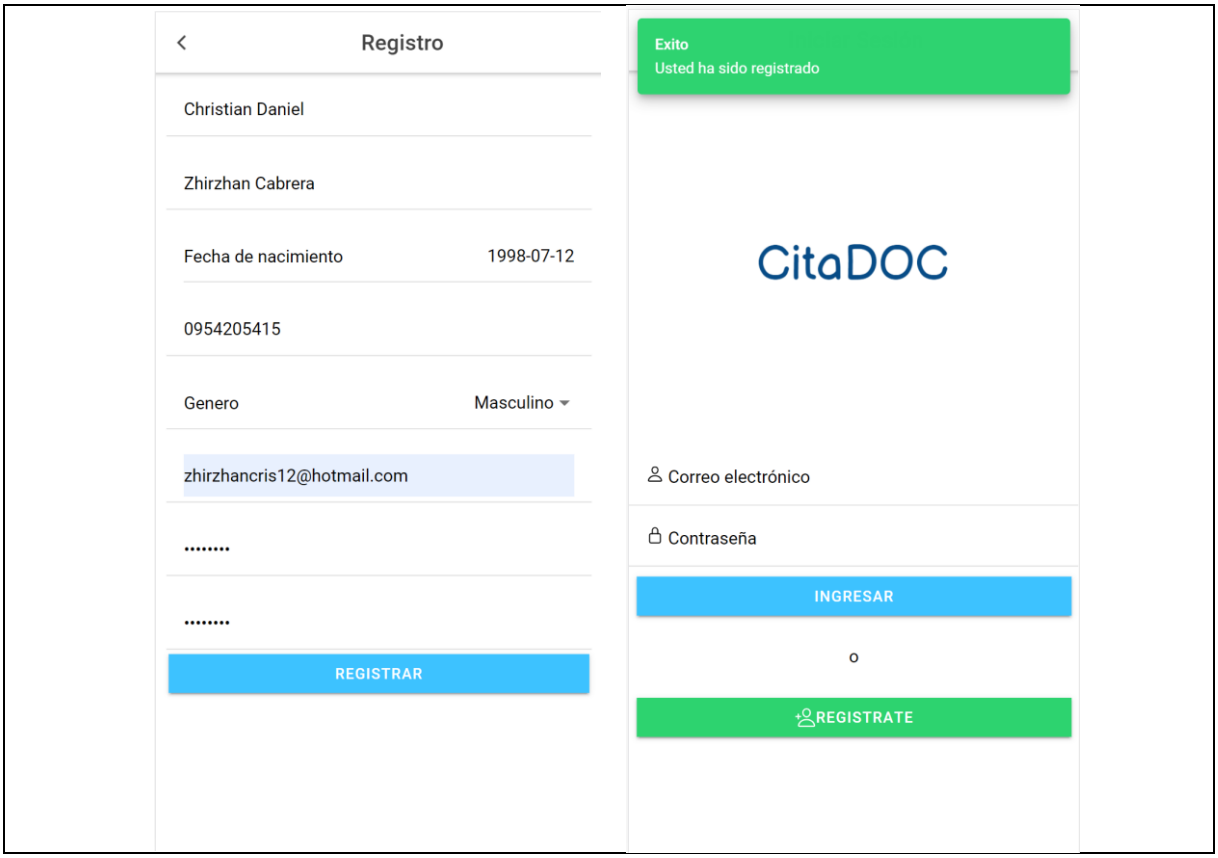
En la Figura 35. Pantalla datos del médico se puede observar la pantalla que muestra los datos personales y académicos de un médico seleccionado.

Figura 36. Pantalla perfil del usuario

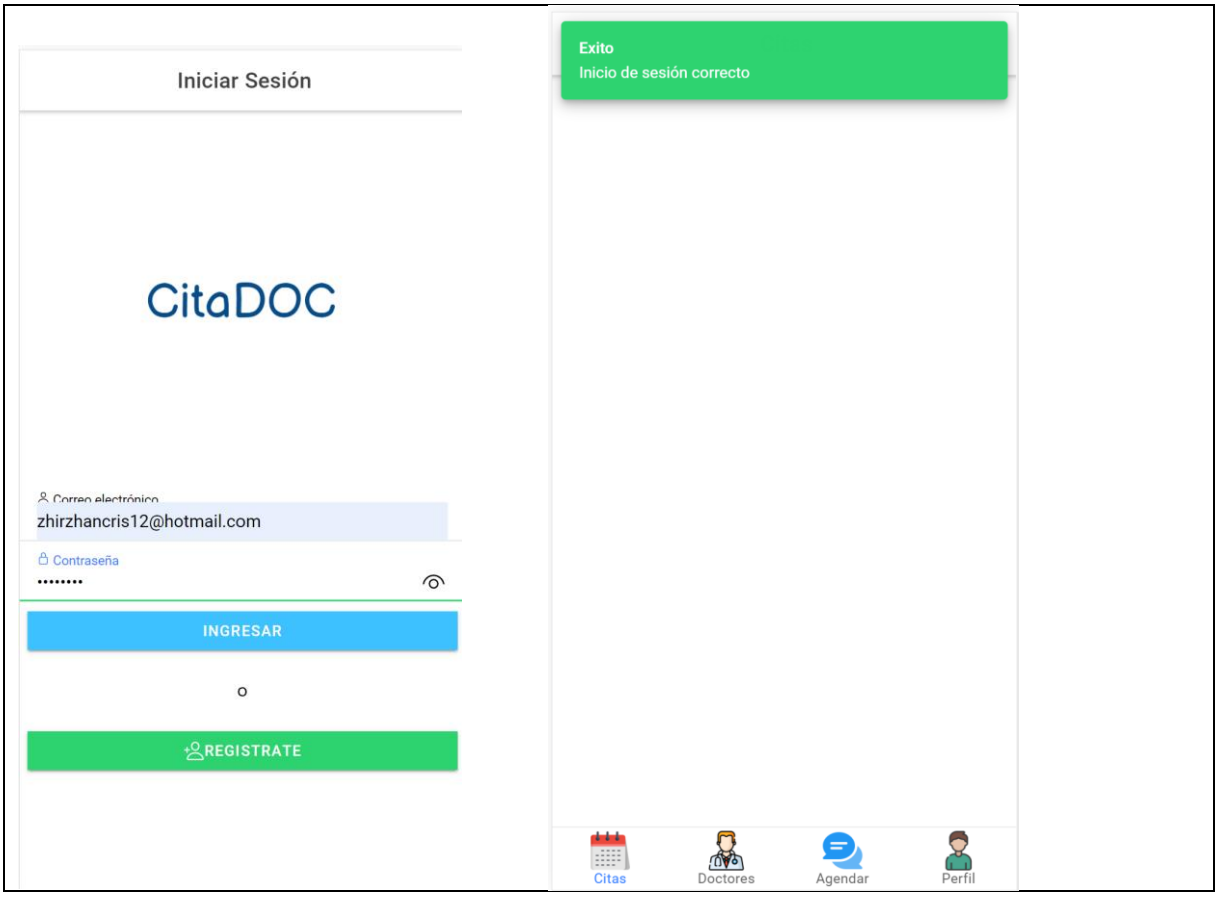
En la Figura 36. Pantalla perfil del usuario se puede observar la pantalla en donde se muestran los datos personales y médicos del usuario, mediante esta también se podrán editar dichos datos.

4.3. Pruebas de requerimientos aplicación paciente

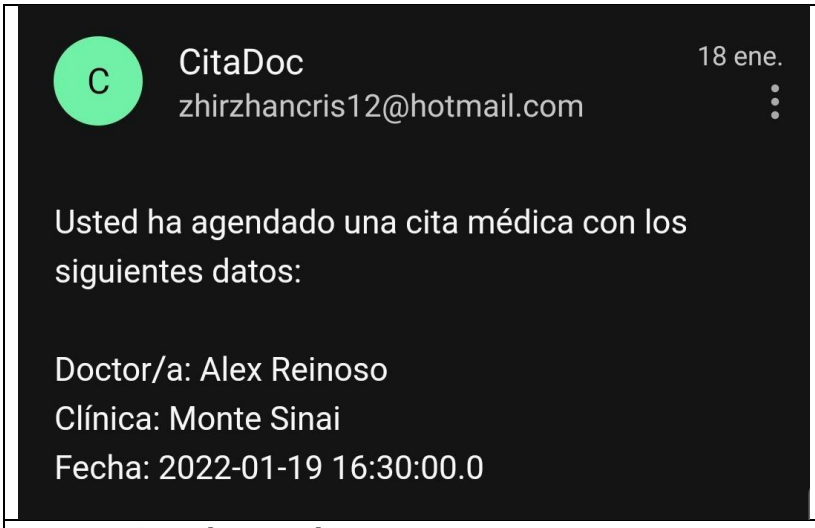
Caso prueba RAP-01	
Responsable	Christian Zhirzhan
Fecha ejecución	23/01/2022
Requerimiento	Registro paciente
¿Se aprobó la prueba? (SI/NO)	SI
Resultados esperados	
Una vez el usuario acceda a la aplicación móvil, este será presentado con una pantalla de inicio de sesión con la opción de registrarse en el caso no contar con una cuenta existente.	
Resultados obtenidos	



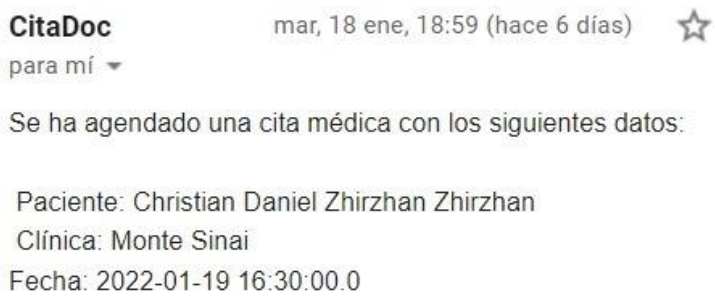
Caso prueba RAP-02	
Responsable	Christian Zhirzhan
Fecha ejecución	23/01/2022
Requerimiento	Inicio de sesión de los usuarios
¿Se aprobó la prueba? (SI/NO)	SI
Resultados esperados	
Una vez el usuario se haya registrado, podrá ingresar al sistema y este mantendrá el inicio de sesión del usuario en el caso de haberlo realizado previamente	
Resultados obtenidos	



Caso prueba RAP-03	
Responsable	Christian Zhirzhan
Fecha ejecución	23/01/2022
Requerimiento	Notificaciones agendamiento cita médica
¿Se aprobó la prueba? (SI/NO)	Si
Resultados esperados	
Cuando el paciente haya agendado una cita médica con un profesional de la salud, la aplicación móvil para el profesional de la salud, la aplicación notificará de este hecho mediante correo electrónico	
Resultados obtenidos	
Correo electrónico paciente	



Correo electrónico médico



Caso prueba RAP-04	
Responsable	Christian Zhirzhan
Fecha ejecución	23/01/2022
Requerimiento	Agendar cita médica
¿Se aprobó la prueba? (SI/NO)	SI

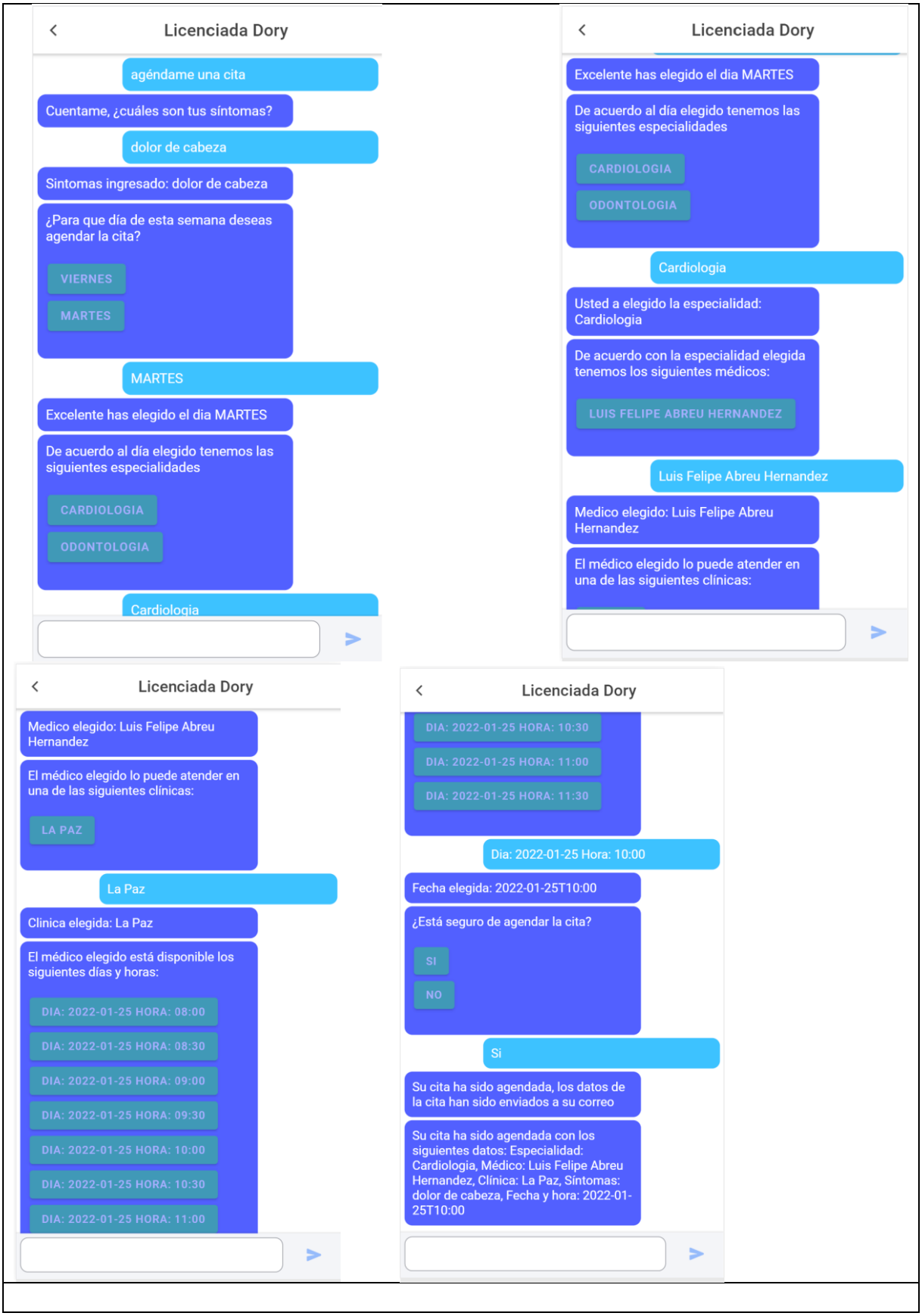
Resultados esperados

El usuario registrado podrá agendar una cita médica mediante el chatbot ingresando "agéndame una cita".

Flujo esperado de la conversación:

1. Solicitar agendar una cita médica al bot.
2. Solicitar al usuario los síntomas.
3. Enviar los síntomas.
4. Mostrar los síntomas ingresados por el usuario.
5. Mostrar los días de la semana que tengan horarios disponibles.
6. Seleccionar día.
7. Mostrar día seleccionado por el usuario
8. Mostrar las especialidades disponibles en el día seleccionado.
9. Seleccionar especialidad.

10. Mostrar especialidad seleccionada por el usuario. 11. Mostrar los médicos disponibles en la especialidad seleccionada. 12. Seleccionar médico. 13. Mostrar médico seleccionado por el usuario. 14. Mostrar las clínicas disponibles en las cuales atiende el médico seleccionado. 15. Seleccionar clínica. 16. Mostrar la clínica seleccionada por el usuario. 17. Mostrar los horarios disponibles en base a los datos antes seleccionados. 18. Seleccionar horario. 19. Mostrar el horario seleccionado por el usuario. 20. Preguntar al usuario si está seguro de agendar la cita. 21. El usuario selecciona una de las opciones disponibles (SI/NO). 22. Mostrar datos recolectados. 23. Informar al usuario sobre el estado del agendamiento de la cita médica.
Resultados obtenidos

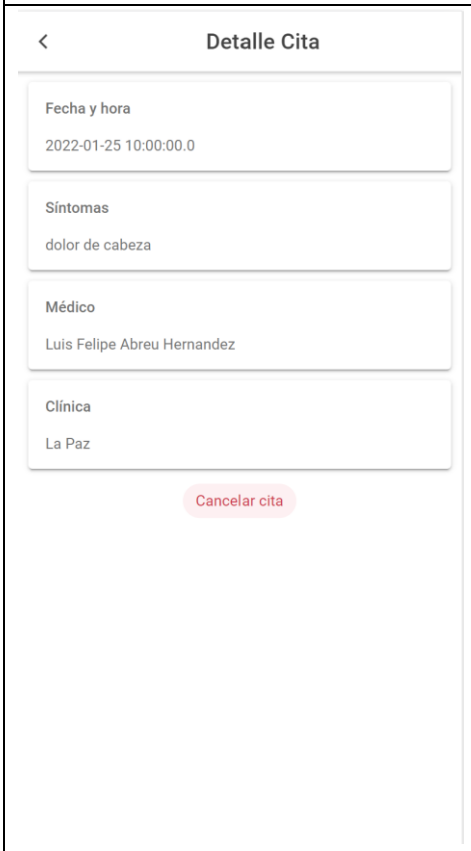

Licenciada Dory
agéndame una cita
Cuentame, ¿cuáles son tus síntomas?
dolor de cabeza
Sintomas ingresado: dolor de cabeza
¿Para que día de esta semana deseas agendar la cita?
VIERNES
MARTES
MARTES
Excelente has elegido el dia MARTES
De acuerdo al día elegido tenemos las siguientes especialidades
CARDIOLOGIA
ODONTOLOGIA
Cardiologia
Licenciada Dory
Excelente has elegido el dia MARTES
De acuerdo al día elegido tenemos las siguientes especialidades
CARDIOLOGIA
ODONTOLOGIA
Cardiologia
Usted a elegido la especialidad: Cardiologia
De acuerdo con la especialidad elegida tenemos los siguientes médicos:
LUIS FELIPE ABREU HERNANDEZ
Luis Felipe Abreu Hernandez
Medico elegido: Luis Felipe Abreu Hernandez
El médico elegido lo puede atender en una de las siguientes clínicas:
Licenciada Dory
Medico elegido: Luis Felipe Abreu Hernandez
El médico elegido lo puede atender en una de las siguientes clínicas:
LA PAZ
La Paz
Clinica elegida: La Paz
El médico elegido está disponible los siguientes días y horas:
DIA: 2022-01-25 HORA: 08:00
DIA: 2022-01-25 HORA: 08:30
DIA: 2022-01-25 HORA: 09:00
DIA: 2022-01-25 HORA: 09:30
DIA: 2022-01-25 HORA: 10:00
DIA: 2022-01-25 HORA: 10:30
DIA: 2022-01-25 HORA: 11:00
Licenciada Dory
DIA: 2022-01-25 HORA: 10:30
DIA: 2022-01-25 HORA: 11:00
DIA: 2022-01-25 HORA: 11:30
Dia: 2022-01-25 Hora: 10:00
Fecha elegida: 2022-01-25T10:00
¿Está seguro de agendar la cita?
SI
NO
Si
Su cita ha sido agendada, los datos de la cita han sido enviados a su correo
Su cita ha sido agendada con los siguientes datos: Especialidad: Cardiologia, Médico: Luis Felipe Abreu Hernandez, Clínica: La Paz, Síntomas: dolor de cabeza, Fecha y hora: 2022-01-25T10:00

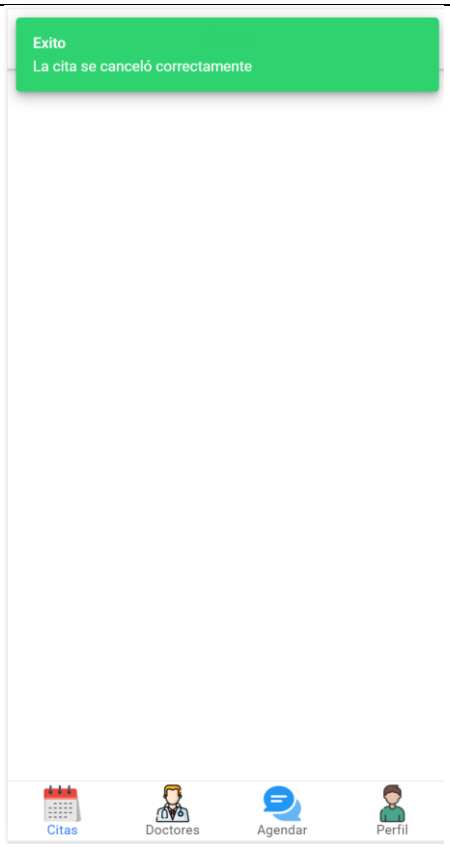
Caso prueba RAP-05	
Responsable	Christian Zhirzhan
Fecha ejecución	23/01/2022
Requerimiento	Editar cita médica.
¿Se aprobó la prueba? (SI/NO)	SI

Resultados esperados

Permite que el usuario cancele una cita médica y a su vez se notifique al médico mediante correo electrónico.

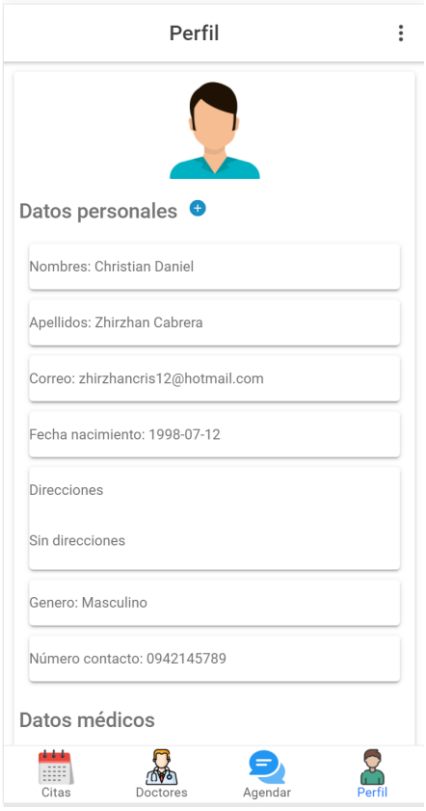
Resultados obtenidos

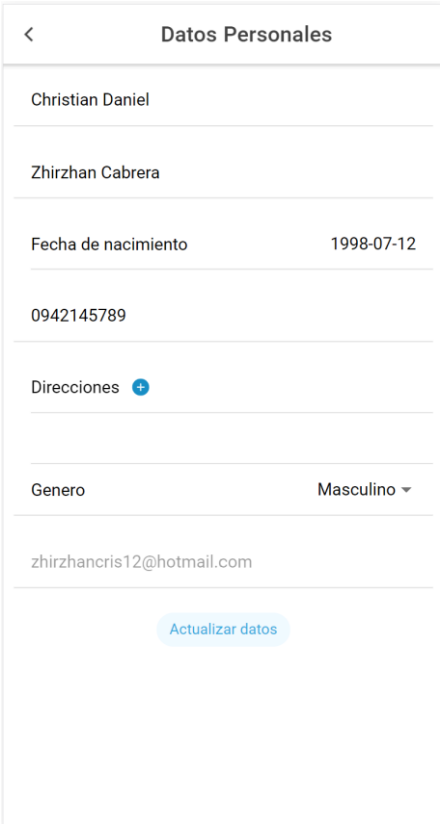


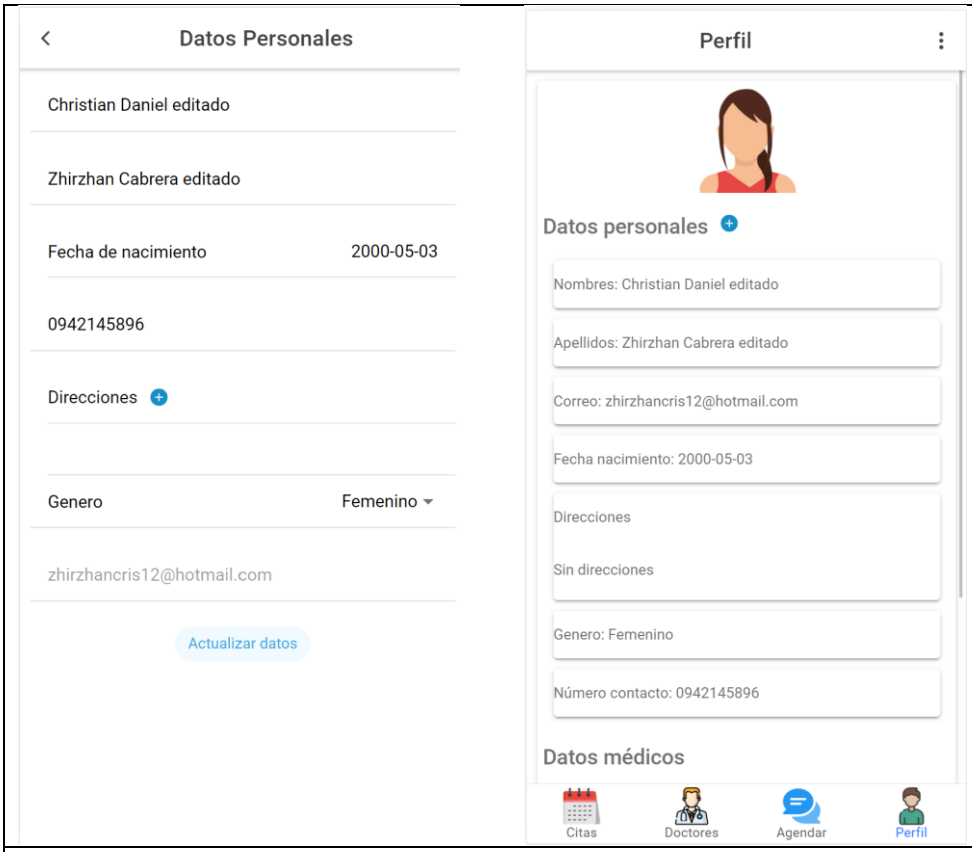


Caso prueba RAP-06	
Responsable	Christian Zhirzhan
Fecha ejecución	23/01/2022
Requerimiento	Editar información usuario, paciente
¿Se aprobó la prueba? (SI/NO)	SI
Resultados esperados	
Permite editar los datos personales del paciente y agregar patologías, cirugías y direcciones, como así también editarlos y eliminarlos.	
Resultados obtenidos	

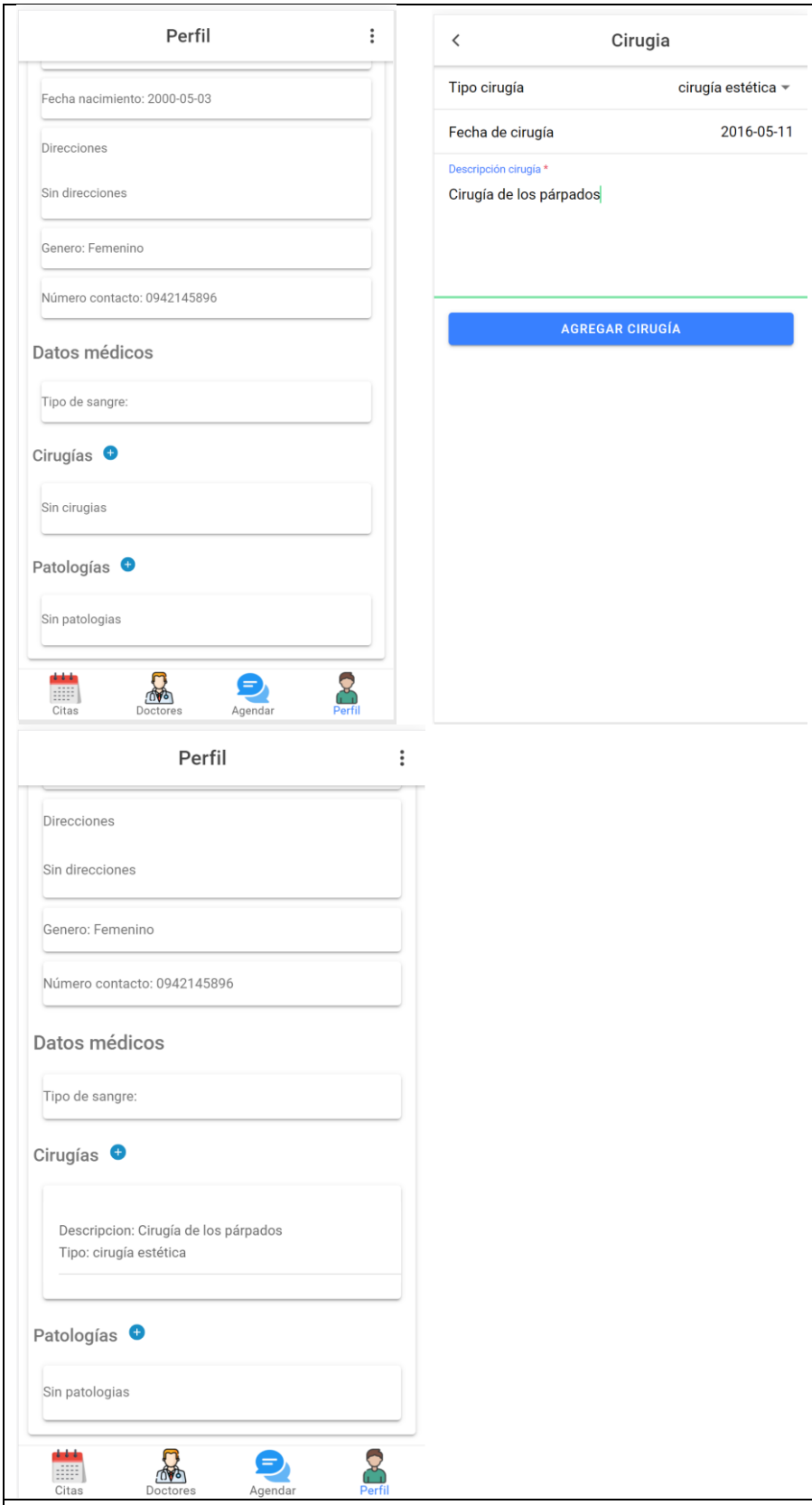
Edición de datos personales



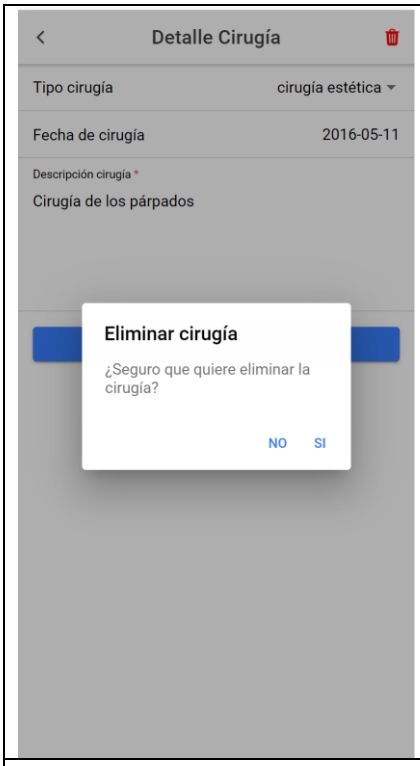


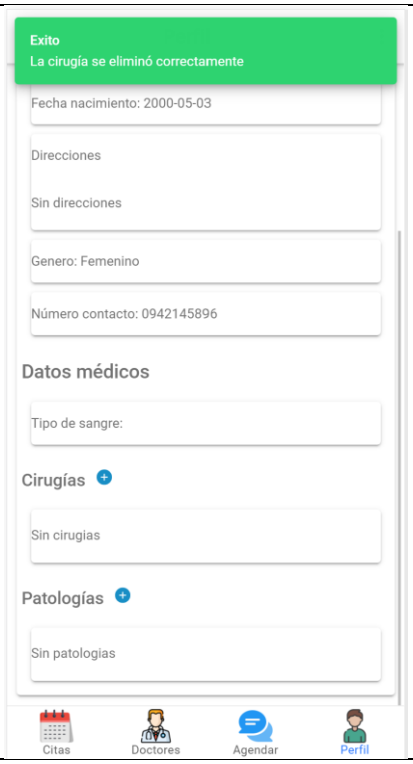


Agregar cirugías

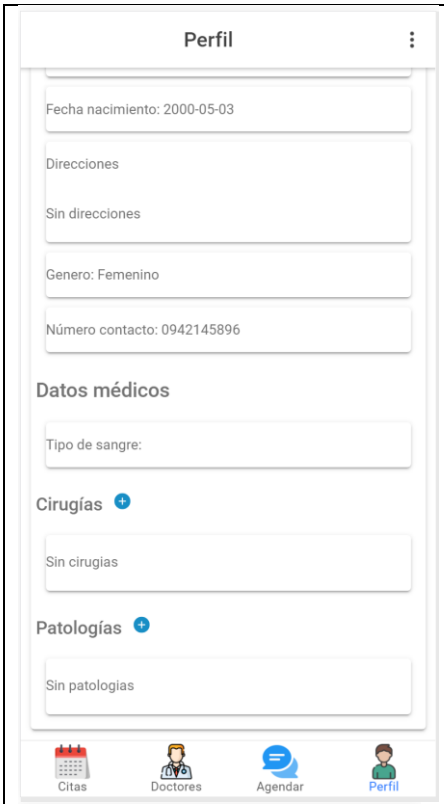


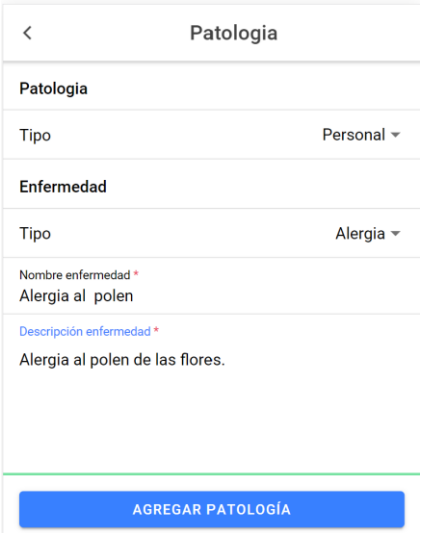
Eliminar cirugía

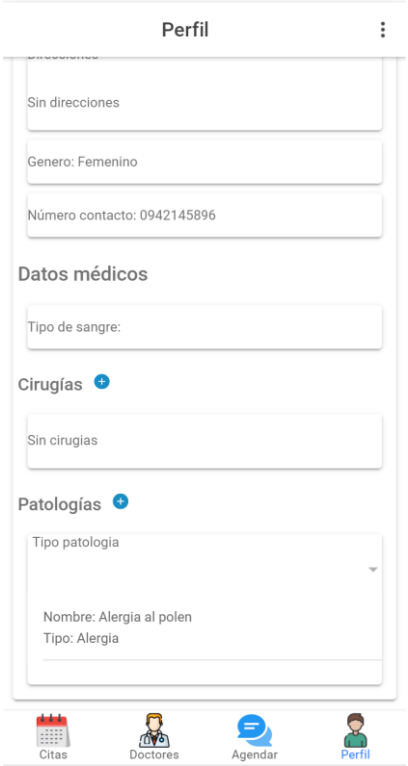




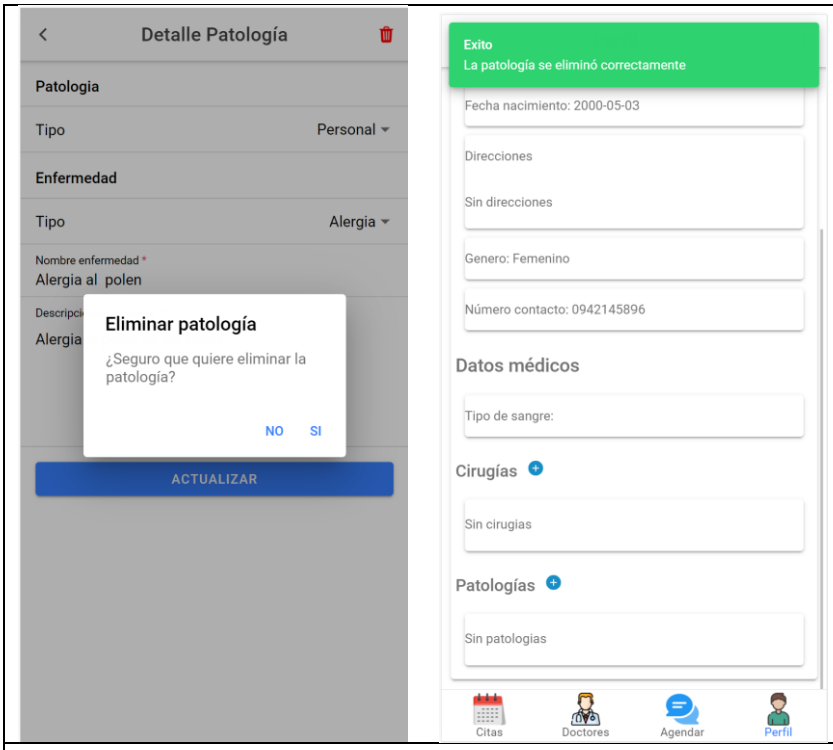
Agregar patologías



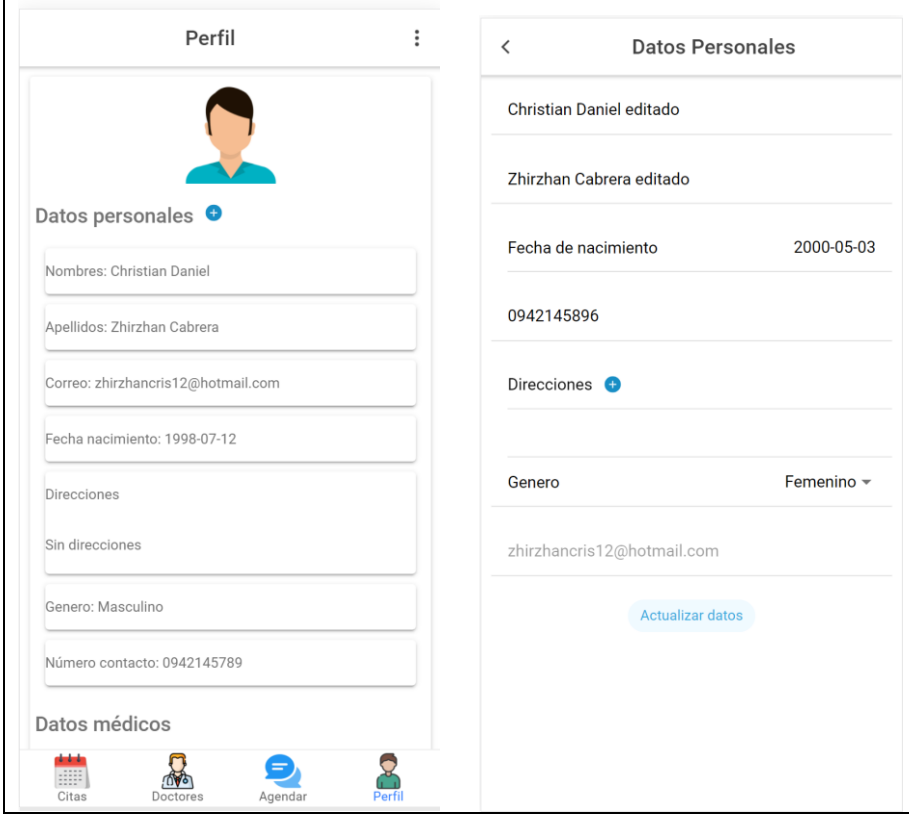


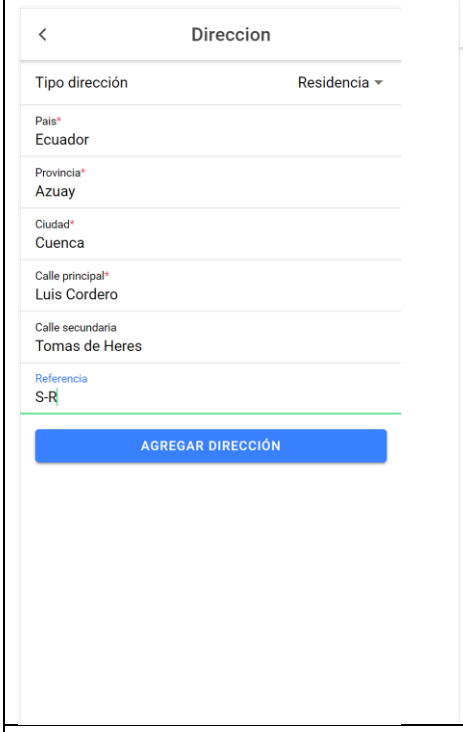


Eliminar patología

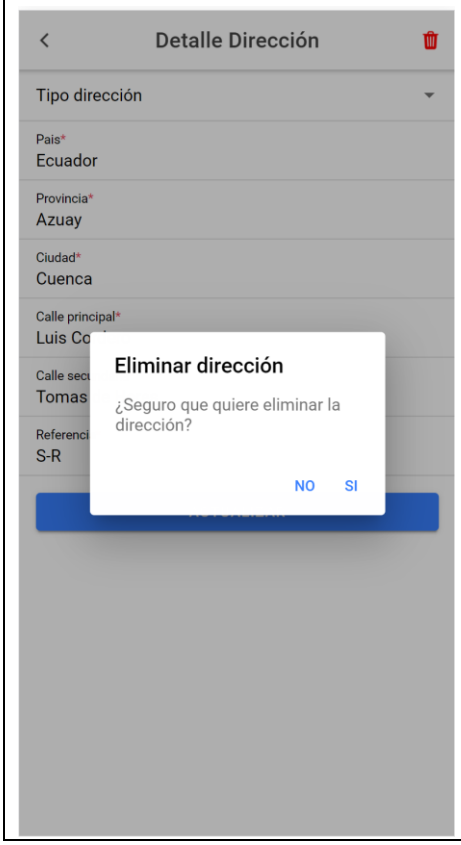


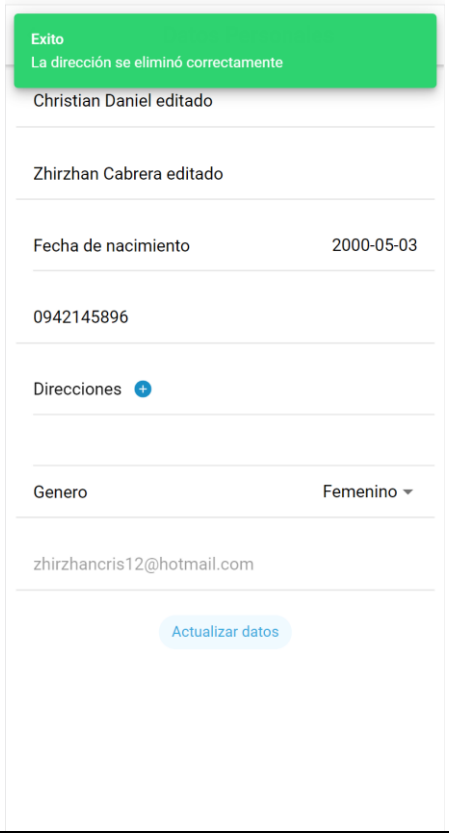
Agregar dirección





Eliminar dirección

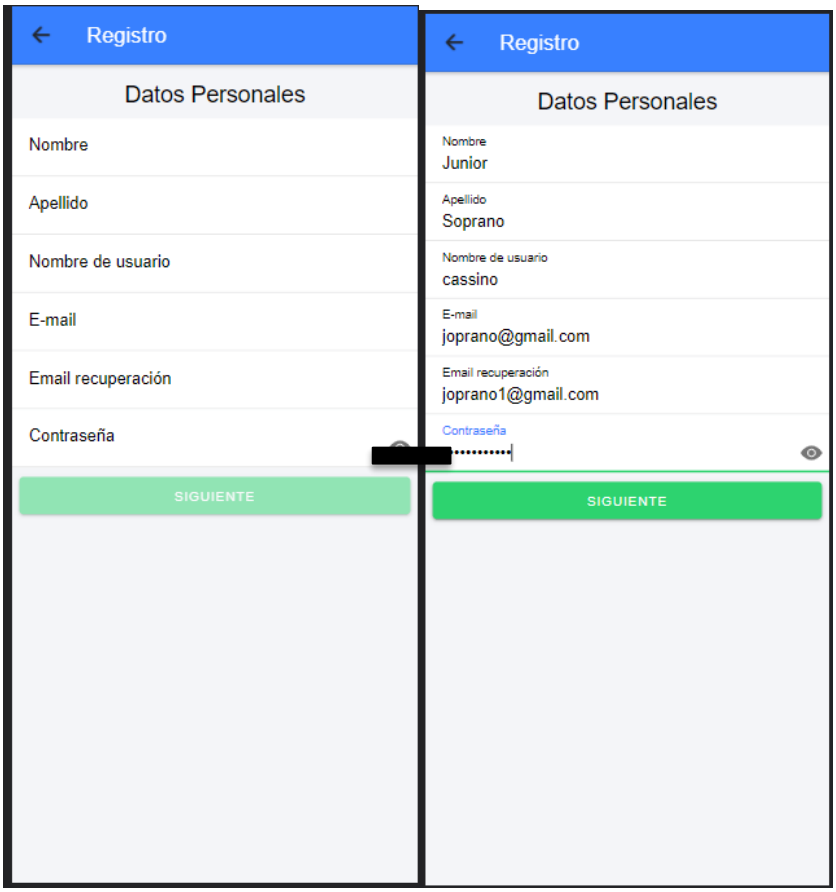




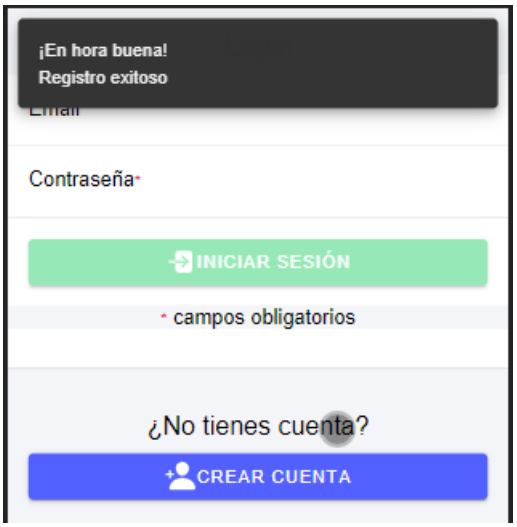
4.4. Pruebas de requerimientos aplicación médico

Abreviatura: Prueba Aplicación Médico (PAM)

Caso prueba PAM-01	
Responsable	Alex Reinoso
Fecha ejecución	23/01/2022
Requerimiento	Registro profesional de la salud
¿Se aprobó la prueba? (SI/NO)	Si
Resultados esperados	
Cuando el profesional de la salud se registra, se le presentan dos formularios, no se permite el registro sin que estos dos formularios sean validados; una vez se validen, se procede al registro, se le muestra un mensaje de confirmación exitosa de registro y se redirige a la pantalla de Login.	
Resultados obtenidos	
Registro sin validar no permite avanzar a la sección final del registro, registro validado permite avanzar con el registro.	



Mensaje de validación y redirección a página de login



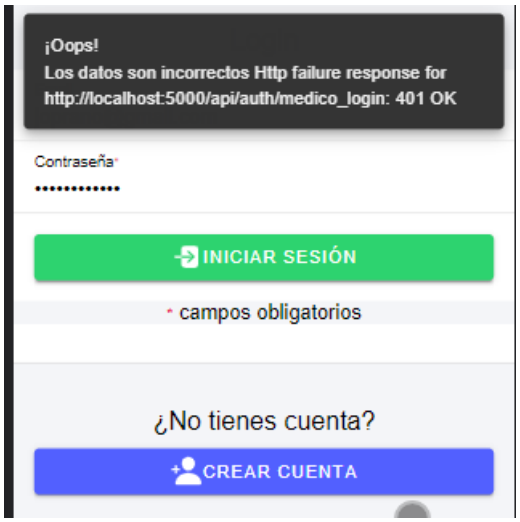
Caso prueba PAM-02	
Responsable	Alex Reinoso
Fecha ejecución	23/01/2022
Requerimiento	Inicio de sesión
¿Se aprobó la prueba? (SI/NO)	Si

Resultados esperados

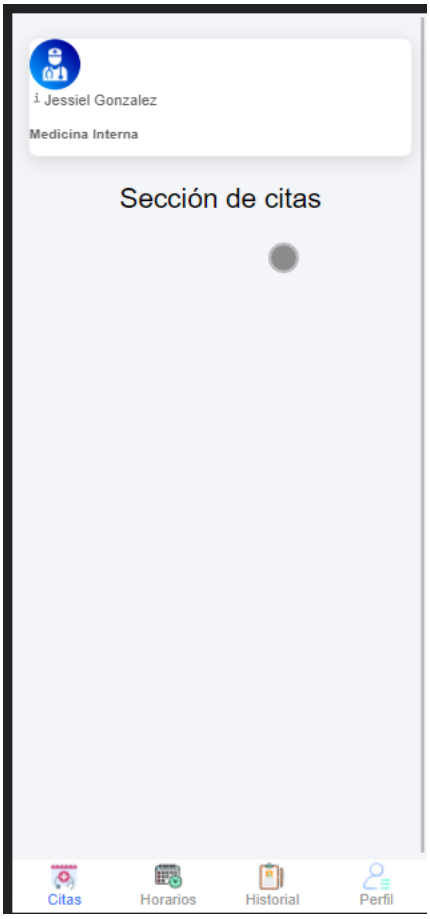
Una vez los usuarios se hayan registrado, estos pueden iniciar sesión y se le redirige a la pantalla principal de cada aplicación. En el caso de que los datos sean incorrectos, se muestra un mensaje de error y se solicita el ingreso de credenciales correctas.

Resultados obtenidos

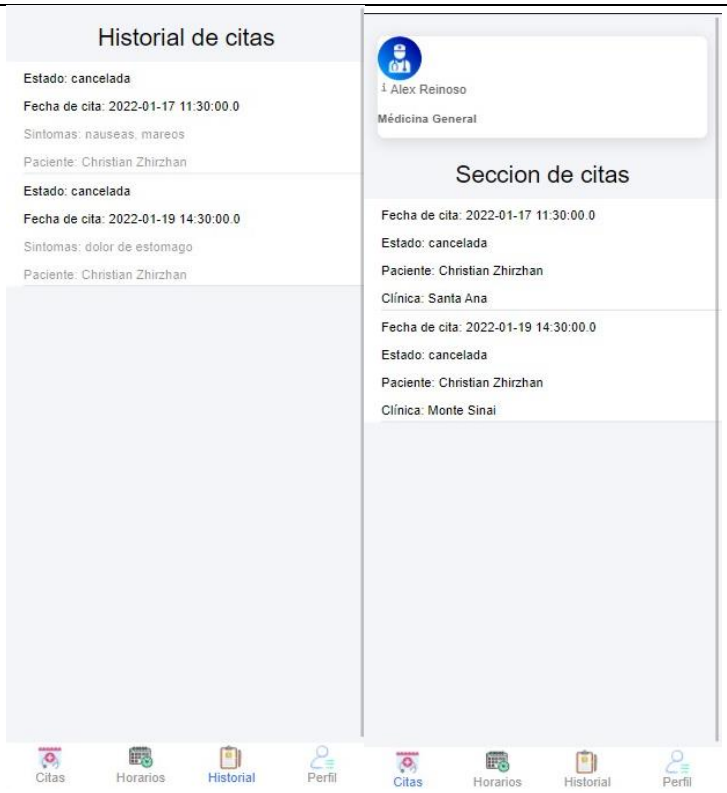
Ingreso de sesión no exitoso



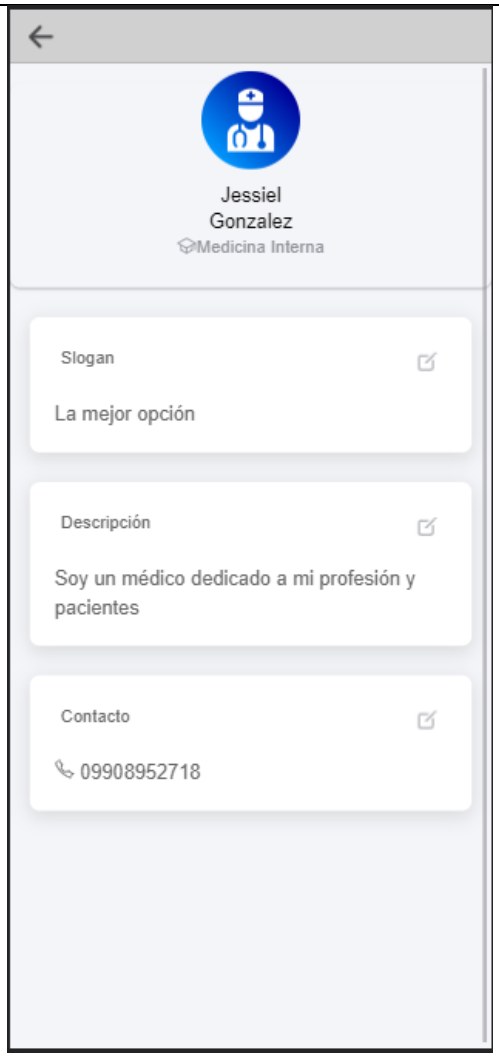
Ingreso de sesión exitosos



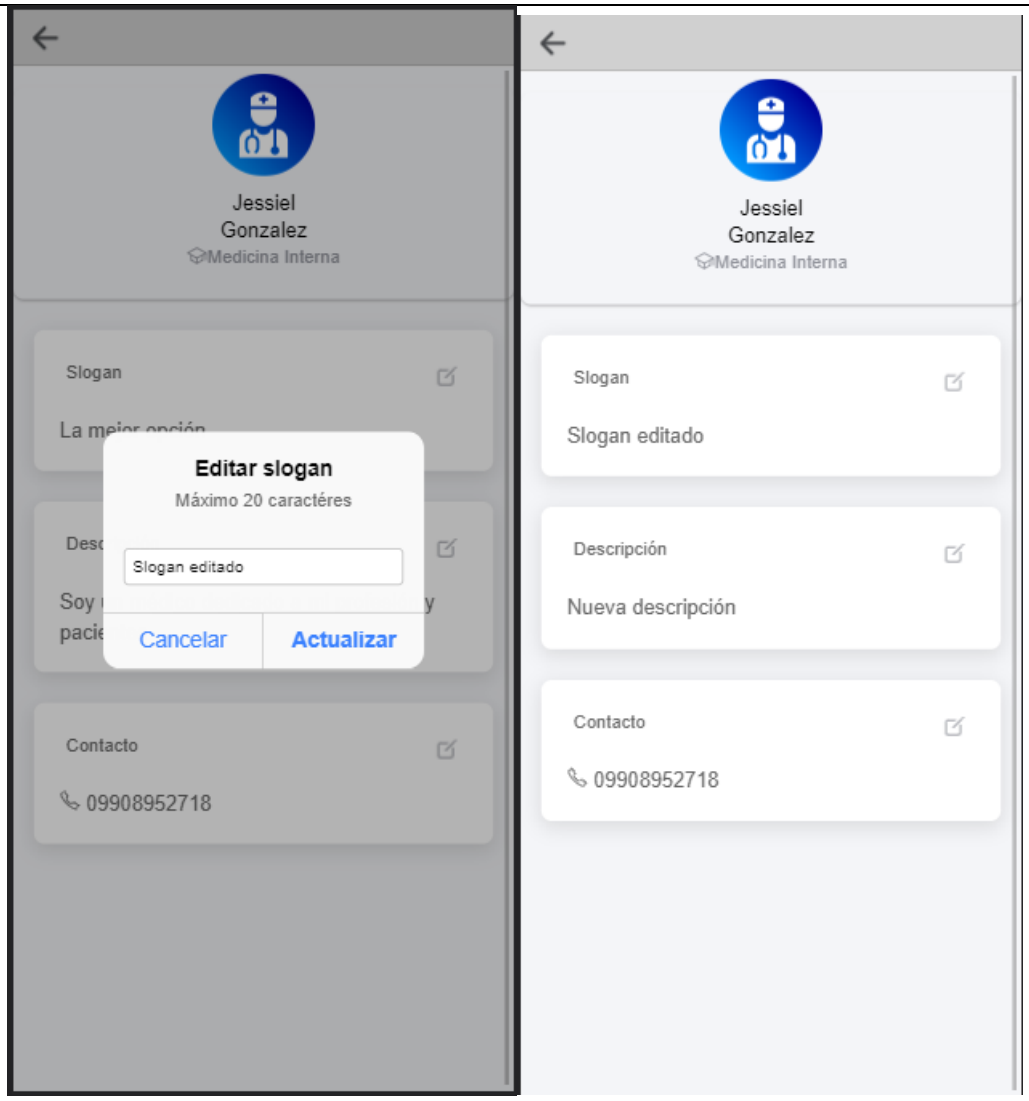
Caso prueba PAM-03	
Responsable	Alex Reinoso
Fecha ejecución	23/01/2022
Requerimiento	Historial de citas médicas
¿Se aprobó la prueba? (SI/NO)	Si
Resultados esperados	
Pantalla que contiene el historial de las citas médicas agendadas y otro con las citas médicas agendadas a partir del día de consulta.	
Resultados obtenidos	
Sección de historial de citas médicas y citas médicas a partir del día de consulta	



Caso prueba PAM-04	
Responsable	Alex Reinoso
Fecha ejecución	23/01/2022
Requerimiento	Editar información de usuario, médico
¿Se aprobó la prueba? (SI/NO)	Si
Resultados esperados	
Se permite editar información del médico	
Resultados obtenidos	
Vista de información general	



Ingreso de sesión exitosos



4.5. Encuestas

Se presentó el funcionamiento de las aplicaciones a algunos profesionales de la salud a los cuales se les realizó una encuesta sobre la usabilidad y el desempeño de ambas aplicaciones, a continuación, se muestran los resultados de las diferentes preguntas realizadas:

- ¿Cómo calificaría la facilidad de registro en la aplicación?

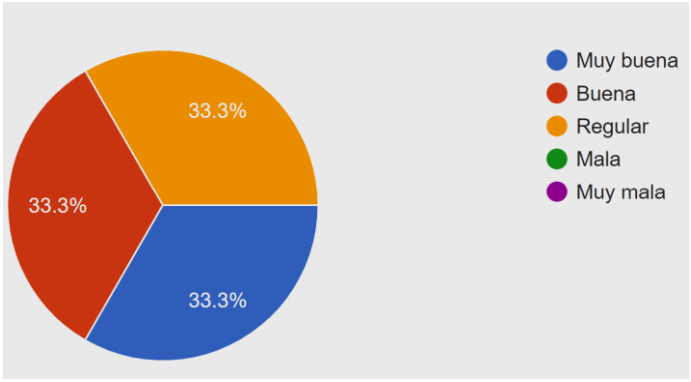


Figura 37. Grafica respuestas pregunta 1

- ¿Cómo calificaría la facilidad para agendar una cita médica usando el Chatbot?

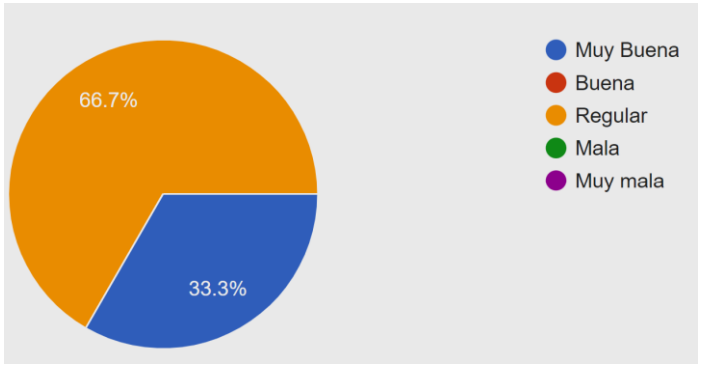


Figura 38. Gráfica respuestas pregunta 2

- ¿Qué le parece la gestión de los datos personales y médicos del paciente?

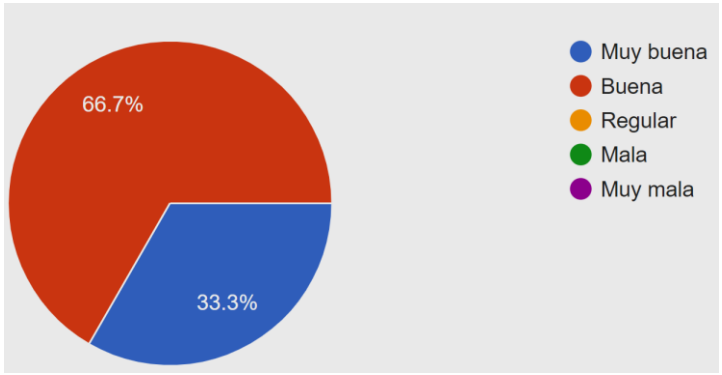
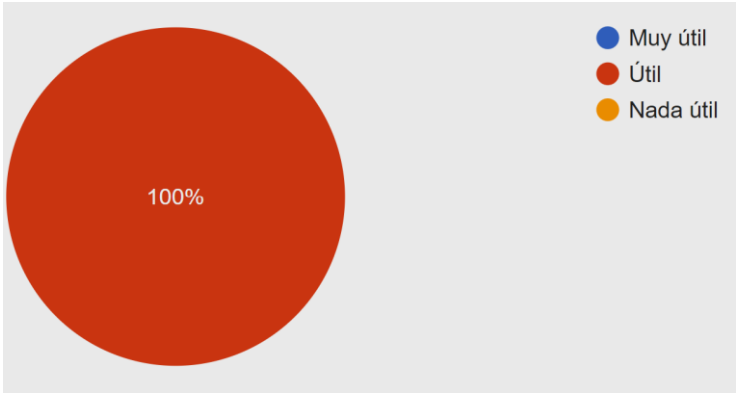


Figura 39. Gráfica respuestas pregunta 3

- ¿Qué tan útil es el sistema de notificaciones por correo?



- *Figura 40. Gráfica respuestas pregunta 4*

- En el caso de que esta aplicación llegase a estar disponible para el público, ¿Qué tan probable es que usted la use para agendar sus citas médicas?

  Respuesta 1: Listado de médicos por especialidad para el agendamiento, omitir síntomas para agendar la cita, agregar opciones de pago en línea.

  Respuesta 2: Mejorar el sistema de notificación de citas, es muy limitado que sea únicamente por mail.

  Respuesta 3: Agendamiento por búsqueda de médicos y no basado en síntomas.

Los médicos encuestados concuerdan en que se deben implementar notificaciones propias de la aplicación a parte de las del correo electrónico, así también se mejoró el proceso de agendamiento con algunas sugerencias dadas por los mismos.

5. Cronograma y presupuesto

Este capítulo contiene el cronograma de actividades del proyecto dividido en Sprints y el presupuesto del proyecto.

5.1. Cronograma

Actividad	Duración (horas)	Inicio	Finalización	Encargado
Proyecto	**649.3**	Lun 11/10/2021	Mie 19/01/22	
Sprint 1		Lun 11/10/2021	Vie 22/10/21	
Sprint Planning	0.15	Lun 11/10/2021	Lun 11/10/2021	Gabriel León
OE. 1. ACT. 1. Estudiar los modelos del procesamiento del lenguaje natural.	20	Lun 11/10/2021	Jue 14/10/21	Alex Reinoso, Christian Zhirzhan
OE. 1. ACT. 2. Estudio de los chatbots basados en reglas y autodidactas.	20	Vie 15/10/21	Mie 20/10/21	Alex Reinoso, Christian Zhirzhan
OE.1. ACT. 3. Documentación del conocimiento relevante para el informe final.	5	Mie 20/10/21	Jue 21/10/21	Alex Reinoso, Christian Zhirzhan
Sprint Review	1	Vie 22/10/21	Vie 22/10/21	Gabriel León
REVISION/CORRECION	4	Sab 23/10/21	Sab 23/10/21	Alex Reinoso, Christian Zhirzhan
Documentación	5	Dom 24/10/21	Dom 24/10/21	Alex Reinoso, Christian Zhirzhan
Sprint 2		Lun 25/10/21	Vie 05/11/21	
Sprint Planning	0.5	Lun 25/10/21	Lun 25/10/21	Gabriel León

OE.3. - ACT. 1. Especificación y análisis de los requerimientos.	20	Lun 25/10/21	Mar 26/10/21	Alex Reinoso, Christian Zhirzhan
OE.3. ACT. 2. Diseño y prototipado de la aplicación.	20	Mie 27/10/21	Jue 28/10/21	Alex Reinoso, Christian Zhirzhan
Sprint Meeting	1	Vie 29/10/21	Vie 29/10/21	Gabriel León, Cecilia Ochoa
OE. 2. ACT. 2. Determinar el conjunto de datos para el entrenamiento.	40	Vie 29/10/21	Lun 01/11/21	Alex Reinoso, Christian Zhirzhan
Sprint Meeting	1	Lun 01/11/21	Lun 01/11/21	Gabriel León
Sprint Review	1	Mar 02/11/21	Mar 02/11/21	Gabriel León, Cecilia Ochoa, Alex Reinoso, Christian Zhirzhan
REVISION/CORRECION	3	Mar 02/11/21	Mar 02/11/21	Alex Reinoso, Christian Zhirzhan
OE.1 ACT. 3. Documentación del conocimiento relevante para el informe final.	5	Jue 04/11/21	Jue 04/11/21	Alex Reinoso, Christian Zhirzhan
Sprint 3		Vie 05/11/21	Vie 05/11/21	
Sprint Planning	1	Vie 05/11/21	Vie 05/11/21	Gabriel León

OE. 2. ACT. 1. Determinar las tecnologías a utilizar para el PLN.	30	Vie 05/11/21	Mar 09/11/21	Alex Reinoso, Christian Zhirzhan
Sprint Meeting	0.15	Mar 09/11/21	Mar 09/11/21	Gabriel León
OE.2. ACT. 3. Implementar y entrenar el modelo de PLN para el agendamiento de citas médicas.	80	Mier 10/11/21	Jue 18/11/21	Alex Reinoso, Christian Zhirzhan
Sprint Meeting	0.2	Vie 19/11/21	Vie 19/11/21	Gabriel León
OE. 2. ACT. 4. Comprobar y validar la eficacia del modelo entrenado.	20	Sab 20/11/21	Dom 21/11/21	Alex Reinoso, Christian Zhirzhan
Sprint Review	1	Lun 22/11/21	Lun 22/11/21	Gabriel León, Cecilia Ochoa, Alex Reinoso, Christian Zhirzhan
REVISION/CORRECION	14	Lun 22/11/21	Mar 23/11/21	Alex Reinoso, Christian Zhirzhan
OE. 2. ACT. 5. Documentar información relevante en el informe final.	15	Mie 24/11/21	Jue 25/11/21	
Sprint 4		Vie 26/11/21	Vie 10/12/21	
Sprint Planning	1	Vie 26/11/21	Vie 26/11/21	Gabriel León
OE. 3. ACT. 3. Diseño de la base de datos (modelo entidad relación).	10	Vie 26/11/21	Vie 26/11/21	Alex Reinoso, Christian Zhirzhan

OE. 3. ACT. 5. Diseño y desarrollo del Backend o capa de servicios. Primera Fase.	50	Lun 29/11/21	Jue 02/12/21	Alex Reinoso, Christian Zhirzhan
Sprint Meeting	0.15	Jue 02/12/21	Jue 02/12/21	Gabriel León
OE. 3. ACT. 5. Diseño y desarrollo del Backend o capa de servicios. Segunda Fase.	50	Vie 03/11/21	Jue 09/12/21	Alex Reinoso, Christian Zhirzhan
Sprint Review	1	Vie 10/12/21	Vie 10/12/21	Gabriel León, Cecilia Ochoa, Alex Reinoso, Christian Zhirzhan
REVISION/CORRECION	5	Vie 10/12/21	Vie 10/12/21	Alex Reinoso, Christian Zhirzhan
OE. 3. ACT. 6. Documentar información relevante para el informe final. Primera Fase.	10	Sab 11/12/21	Sab 11/12/21	Alex Reinoso, Christian Zhirzhan
Sprint 5		Lun 13/12/21	Vie 17/12/21	
Sprint Planning	1	Lun 13/12/21	Lun 13/12/21	Gabriel León
OE. 3. ACT. 4. Diseño y desarrollo de las interfaces de usuario para las dos aplicaciones. Paciente.	50	Lun 13/12/21	Mie 22/12/21	Alex Reinoso
Sprint Meeting	0.15	Jue 23/12/21	Jue 23/12/21	Gabriel León
OE. 3. ACT. 4. Diseño y desarrollo de las interfaces de usuario para las dos aplicaciones. Profesional Salud.	50	Jue 23/12/21	Lun 03/01/22	Christian Zhirzhan

Sprint Review	1	Mar 04/01/22	Mar 04/01/22	Gabriel León, Cecilia Ochoa, Alex Reinoso, Christian Zhirzhan
REVISION/CORRECION	20	Mar 04/01/22	Mie 05/01/22	Alex Reinoso, Christian Zhirzhan
Documentación	5	Jue 06/01/22	Jue 06/01/22	Alex Reinoso, Christian Zhirzhan
Sprint 6		Vie 07/01/22	Vie 07/01/22	
Sprint Planning	1	Vie 07/01/22	Vie 07/01/22	Gabriel León
OE. 4. ACT. 1. Diseño de un plan de pruebas de funcionamiento.	35	Vie 07/01/22	Lun 10/01/22	Alex Reinoso, Christian Zhirzhan
OE. 4. ACT. 2. Ejecución de pruebas.	20	Mar 11/01/22	Jue 13/01/22	Alex Reinoso, Christian Zhirzhan
OE. 4. Documentar datos relevantes de las pruebas.	10	Vie 14/01/22	Vie 14/01/22	Alex Reinoso, Christian Zhirzhan
OE. 4. Desarrollo informe final.	10	Sab 15/01/22	Sab 15/01/22	Alex Reinoso, Christian Zhirzhan

Sprint Review	1	Lun 17/01/22	Lun 17/01/22	Gabriel León, Cecilia Ochoa, Alex Reinoso, Christian Zhirzhan
REVISION/CORRECCION	10	Lun 17/01/22	Mie 19/01/22	Alex Reinoso, Christian Zhirzhan

5.2. Presupuesto

Denominación	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
	unidades	dólares	dólares
1. Bienes			
Impresiones	50	$0.05	$2.50
Copias	50	$0.01	$0.50
Empastado	2	$9.00	$18.00
2. Tecnológico			
Laptop	2	$1,300.00	$2,600.00
Celular Inteligente	2	$400.00	$800.00
Servidores	2	$304.96	$609.92
Plantillas Front-end	2	$75.00	$150.00
Internet	4	$34.00	$136.00
3. Servicios			
Transporte publico	32	$0.30	$9.60
Transporte intercantonal	16	$1.45	$23.20
Taxis	20	$2.00	$40.00
4. Personal			
Horas asesoría especializada	14	$18.00	$252.00
Horas desarrollo	650	$4.16	$2,704.00
5. Misceláneo			
Imprevistos	2	$50.00	$100.00
Total	848	$2,148.93	$7,345.72

Conclusiones

Se logró implementar el asistente virtual para el agendamiento de citas médicas en una aplicación multiplataforma desarrollada con Ionic mediante el uso de Rasa server para el procesamiento del lenguaje natural.

El sistema a manera de interfaces de usuarios se conforma de dos aplicaciones móviles, una destinada para el agendamiento de citas médicas de parte de los pacientes y otra para profesionales de la salud dentro del área geográfica de Cuenca – Ecuador, en esta aplicación se puede tener un registro de las citas que se han agendado, de igual manera como ingresar datos académicos relevantes para el agendamiento de una cita médica: especialidades, clínicas en las que atiende y sus horarios respectivos de atención. Mediante el ingreso de los horarios de

atención se generan fechas disponibles filtrando las citas previamente agendadas, que mantienen un estado diferente a “cancelado” y cuya fecha y hora sea mayor a la hora y fecha que consulta del paciente mediante el asistente. De igual manera, se implementó notificaciones de agendamiento y cancelación de citas mediante correo electrónico.

En cuanto a la parte del Backend, se cuenta con un servidor desarrollado con Spring y el OpenJDK 11 de Java para manejar las transacciones entre la capa de negocios con la capa de datos, que para este proyecto se seleccionó una base de datos PostgreSQL versión 12.8.

Por otro lado, se logró implementar la seguridad en las aplicaciones mediante tokens generados a partir correo y contraseña ingresados en el formulario de registro. Esto con la finalidad, de garantizar la seguridad y prevenir el acceso de agentes no deseados a las zonas privadas del sistema. Cabe recalcar que las contraseñas se encriptan previamente a ser guardadas en la base de datos para evitar que estas sean vulneradas.

En conclusión, se logró implementar el agendamiento de citas médicas mediante un asistente virtual utilizando técnicas de procesamiento de lenguaje natural, así también como cumplir con los objetivos planteados; sin embargo, es importante mencionar que todavía existe trabajo a futuro para mejorar el sistema, en cuanto a la parte del asistente: implementar algoritmos de aprendizaje de máquina para el análisis de síntomas y recomendar especialidades basado en estos y comentarios dejados por los pacientes. En la lógica, implementar mayores funcionalidades en la capa de negocio para incluir pagos en línea, conexión directa con el calendario de Google del profesional de la salud, implementaciones para crear una cuenta mediante Google, Facebook, etc.

Recomendaciones

Algo para tener en cuenta es que este es un proyecto algo extenso, debido a que se han implementado los servicios de PLN desde cero con una tecnología Open-source que permite el desarrollo futuro de sus funcionalidades sin restricciones. De igual manera, el Backend, capa de negocios, al haber implementado servicios personalizados, en especial el filtro de seguridad por tokens representa un esfuerzo bastante grande para dos desarrolladores.

Razón por la cual recomendamos hacer uso de tecnologías serverless que permitan enfocar los esfuerzos en funcionalidades más avanzadas para el asistente virtual, funcionalidades como: recomendación de doctores basado en síntomas, reconocimiento de comandos por voz y la capacidad de ser autodidacta; o, en el caso de desear hacer uso de asistentes listos para ser entrenados y desplegados como IBM Watson que permite enfocarse en el entrenamiento y ofrece la capacidad de que el asistente sea autodidacta y facilidad de despliegue a través de un protocolo seguro de internet.

De igual manera, en el caso del Backend, a pesar de la flexibilidad que permite el desarrollar funciones personalizadas, software a la medida, para satisfacer las necesidades de este proyecto todavía quedan muchas más funciones por implementar, funciones que en el caso de hacer uso de tecnologías serverless ya están preparadas y refinadas para hacer uso completo de las ultimas funcionalidades para aplicaciones móviles.

Referencias bibliográficas

Bhargav, S.-D. (2018). *Natural Language Processing and Computational Linguistics* (Packt Publ).

Cabero, Á. C., d'Informàtica de Barcelona, F., & de Catalunya. Departament d'Enginyeria de Serveis i Sistemes d'Informació, U. P. (2020). *Rasa framework - Análisis e implementación de un chatbot*. Universitat Politècnica de Catalunya. Facultat d'Informàtica de Barcelona. https://books.google.com.ec/books?id=S18azgEACAAJ

Chollet, F. (2015). *About Keras*. Keras Special Interest Group. https://keras.io/about/%0Ahttps://keras.io/about/%0Ahttps://keras.io/about/#keras-amp-tensorflow-2%0Ahttps://keras.io/about/

Deng, L., & Liu, Y. (2018). *Deep learning in natural language processing*. Springer.

Doctorisy. (2021). *Doctorisy (Versión 1.8.1)[Aplicación móvil]*. Google Play Store. https://play.google.com/store/apps/details?id=com.bayteq.doctorisy.paciente

Familiar, B. (2015). Microservice Architecture. In *Microservices, IoT, and Azure: Leveraging DevOps and Microservice Architecture to Deliver SaaS Solutions* (pp. 21–31). Apress. https://doi.org/10.1007/978-1-4842-1275-2_3

iSalud. (2021). *iSalud (Versión 1.0.9)[Aplicación móvil]*. Google Play Store. https://play.google.com/store/apps/details?id=ec.com.dirisalud.app

Lisandro, D., Nicolás, G., Pablo, T., & Patricia, P. (n.d.). *Un Análisis Experimental de Tipo de Aplicaciones para Dispositivos Móviles*.

Loper, E., Loper, S. B., & Steven, B. (2002). *NLTK: The Natural Language Toolkit*.

Prichsouth Tecnologías del Sur. (2021). *SaludEC (Versión 1.19)[Aplicación móvil]*. Google Play Store. https://play.google.com/store/apps/details?id=com.phuyusalud.movil

Primmelabs Tecnologia S.A. (2019). *Hospital Universitario del Rio (versión 1.1.0) [Aplicación móvil]*. App Store. https://apps.apple.com/ec/app/hospital-universitario-del-rio/id1289502994

Rasa Technologies GmbH. (2021). *Introduction to Rasa Action Servers*. https://rasa.com/docs/rasa/

Řehůřek, R., & Sojka, P. (2010). Fast and Faster: A Comparison of Two Streamed Matrix Decomposition Algorithms. *Lecture Notes in Computer Science*, 25–28.

Rouhiainen, L. (2018). Inteligencia artificial. *Madrid: Alienta Editorial*. https://static0planetadelibroscom.cdnstatics.com/libros_contenido_extra/40/39308_Inteligencia_artificial.pdf

Ryan, M. (2018). *Web scraping with Python: Collecting more data from the modern web* (O'Reilly M).

Shridhar, K., Dash, A., Sahu, A., Pihlgren, G. G., Alonso, P., Pondenkandath, V., Kovacs, G., Simistira, F., & Liwicki, M. (2019). Subword Semantic Hashing for Intent Classification on Small Datasets. *Proceedings of the International Joint Conference on Neural Networks, 2019-July*. https://doi.org/10.1109/IJCNN.2019.8852420

Steven, L. (2020). *TextBlob: Simplified Text Processing*.

Sutherland, J. (2010). Scrum handbook. *Scrum Training Institute, May,* 464.

TechTarget Contributor. (2021). *natural language understanding (NLU).* https://searchenterpriseai.techtarget.com/definition/natural-language-understanding-NLU

Telégrafo, E. (2021). *Ministerio de Salud reportó inconvenientes en línea 171. ¿Cómo acceder a una cita médica?* https://www.eltelegrafo.com.ec/noticias/salud/1/ministerio-de-salud-reporto-inconvenientes-en-linea-171-como-acceder-a-una-cita-medica

Telemédico Ecuador. (2021). *Telemédico (Versión 1.0.1)[Aplicación móvil]*. Google Play

Store. https://play.google.com/store/apps/details?id=com.telemedico.paciente&hl=es_EC&gl=US

Tom, B., Joey, F., Nick, P., & Alan, N. (2017). *Rasa: Open Source Language Understanding and*.

Tran, A. D., Pallant, J. I., & Johnson, L. W. (2021). Exploring the impact of chatbots on consumer sentiment and expectations in retail. *Journal of Retailing and Consumer Services*, *63*, 102718. https://doi.org/10.1016/j.jretconser.2021.102718

Waranashiwar, J., & Ukey, M. (2018). Ionic Framework with Angular for Hybrid App Development. *International Journal of New Technology and Research (IJNTR)*.

Wenhao, W. (2018). *React Native vs Flutter, cross-platform mobile*.