PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

SISTEMAS DE INFORMACIÓN

TEMA:

DESARROLLO DE UN PROTOTIPO DE APLICACIÓN MÓVIL PARA CONTACTAR DUEÑOS DE MASCOTAS CON PASEADORES DE PERROS.

AUTOR:

JORGE SEBASTIAN ANDRADE ABRIL

QUITO DM, JUNIO DE 2024

DEDICATORIA

Dedico este trabajo de titulación principalmente a mis padres siempre son un pilar fundamental en la obtención de mis objetivos profesionales. Gracias a ellos estoy donde estoy ahora, siempre han estado ahí apoyándome en las buenas y en las malas.

AGRADECIMIENTO

Agradezco a Dios que me ha guiado durante todo mi camino universitario, y me ha permitido tomar buenas decisiones a lo largo de la vida.

A mis padres que día a día me impulsan a conseguir mis metas, y luchar por ellas cada día.

A mis amigos que me han apoyado que se han encontrado en los buenos y en los malos momentos.

A mis maestros que me han compartido el conocimiento para poder desarrollarme profesionalmente.

RESUMEN

La tecnología ha avanzado considerablemente a lo largo de los años, centralizando hoy en día toda la información en los teléfonos celulares. Por esta razón, he decidido desarrollar un prototipo de aplicación que permita conectar a los dueños de mascotas con paseadores de perros. Este prototipo se desarrollará mediante la metodología ágil de prototipado evolutivo, que ofrece desde el inicio una visión preliminar del producto final y permite la incorporación continua de mejoras hasta alcanzar el producto completo.

El desarrollo se realizará utilizando Dart, un lenguaje de programación orientado a objetos que facilita la creación de aplicaciones móviles, y el framework Flutter, que permite que el prototipo sea multiplataforma, es decir, compatible tanto con iOS como con Android. Este proyecto surge en respuesta a la gran cantidad de mascotas en el Distrito Metropolitano de Quito, donde muchas personas poseen una mascota, pero carecen del tiempo necesario para atenderlas adecuadamente. De esta necesidad nace la idea de crear una aplicación que facilite el contacto entre los paseadores de perros y los dueños de mascotas.

ÍNDICE

CONTENIDO

1. INTRODUCCIÓN 12

1.1. Justificación 12

1.2. Planteamiento del problema 13

1.3. Objetivo General 13

1.4. Objetivos Específicos 13

1.5. Antecedentes 14

1.6. Alcance 15

2. MARCO CONCEPTUAL 16

2.1. Metodologías de desarrollo de software 16

2.1.1. Metodologías de desarrollo Tradicional 17

2.1.2. Metodologías Ágiles 18

2.2. Desarrollo Móvil 22

2.3. Back-End 22

2.3.1. NestJS 22

2.3.2. Django 23

2.3.3. ASP.NET 23

2.3.4. Comparación entre Frameworks de desarrollo para el Back-End 23

2.4. FrontEnd 24

2.4.1. IONIC 25

2.4.2. Flutter 25

2.4.3. React Native 25

2.4.4. Comparación entre Frameworks Front-End para el desarrollo móvil 26

2.5. API 27

2.5.1. API REST 27

2.6. Base de Datos 27

2.6.1. Tipos de Base de Datos 28

2.6.2. Comparación entre SQL y NoSQL 29

2.6.3. PostgreSQL 30

2.6.4. Microsoft SQL SERVER 30

2.6.5. MYSQL 31

2.6.6. Comparación entre motores de base de datos relacionales 31

3. Metodología de desarrollo del aplicativo 33

3.1. Prototipado Evolutivo 33

3.1.1. Análisis de Requerimientos 34

3.1.2. Diseño y desarrollo del prototipo 34

3.1.3. Pruebas del prototipo 35

3.1.4. Retroalimentación del prototipo 35

3.1.5. Refinamiento del prototipo 35

4. DESARROLLO 36

4.1.1. Configuración de la base de datos 36

4.1.2. Configuración de la API Google Maps 38

4.1.3. Configuración conexión desde el Fron-End hacia el Back-End 39

4.2. Análisis de Requerimientos 40

4.2.1. Levantamiento de requerimientos 47

4.3. Desarrollo de Prototipos 50

4.3.1. Prototipo de Autenticación 50

4.3.2. Prototipo de Gestión de Mascota 55

4.3.2.1. Desarrollo Prototipo Gestión Mascota 56

4.3.2.2. Pruebas 56

4.3.2.3. Feedback del Usuario 57

4.3.2.4. Refinamiento 57

4.3.3. Prototipo de Gestión de Paseo 57

4.3.3.1. Desarrollo Prototipo Gestión de Paseo 58

4.3.3.2. Pruebas 58

4.3.3.3. Feedback del Usuario 60

4.3.3.4. Refinamiento 60

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES 71

5. Conclusiones y Recomendaciones. 71

5.1. Conclusiones 71

5.2. Recomendaciones 72

BIBLIOGRAFÍA 73

Bibliografía 73

ÍNDICE DE FIGURAS Y TABLAS

TABLAS

Tabla 1 Ventajas y desventajas entre la metodología tradicional y ágil 19

Tabla 2 Ventajas y desventajas de la metodología prototipos 21

Tabla 3 Comparación entre NestJS, Django y ASP.NET 24

Tabla 4 Comparación entre Frameworks de Front-End para desarrollo móvil 26

Tabla 5 Comparación entre SQL y NoSQL 30

Tabla 6 Comparación entre motores de base de datos relacionales 31

Tabla 7 Listado de historias de usuario. 47

Tabla 8 Historia de usuario: Autenticación en el sistema. 48

Tabla 10 Historia de usuario: Administración de mascota. 48

Tabla 11 Historia de usuario: Solicitud de paseo de mascota 49

Tabla 12 Historia de usuario: Ubicación en tiempo real para el dueño y la mascota 50

Tabla 15 Historia de usuario: Calificación de Paseo. 50

FIGURAS

Figura 1 Metodologías Tradicionales de Desarrollo 17

Figura 2 Proceso de SCRUM. 20

Figura 3 Modelo de Prototipado Evolutivo 33

Figura 4 Conexión a Base de Datos 36

Figura 5 Configuración de la ruta de la conexión Back-End 37

Figura 6 Tablas de la Base de Datos 37

Figura 7 Consola API Google 38

Figura 8 Conexión desde el Front-End con el Back-End 40
Figura 9 Pregunta 1 ¿Vives en casa o departamento?.....41
Figura 10 Pregunta 2 ¿Dispone de un espacio físico grande para su mascota? 42
Figura 11 Pregunta 3 ¿Con qué frecuencia saca a pasear a su mascota?.....42
Figura 12 Pregunta 4 ¿Cuánto tiempo a la semana dedica para que su mascota haga actividades físicas o de recreación?.....43
Figura 13 Pregunta 5 ¿Mientras usted no está en casa donde queda la mascota? ..44
Figura 14 Pregunta 6 ¿Contrata algún tipo de servicio para que su mascota realice actividad física? 45
Figura 15 Pregunta 7 ¿Usted confiaría en encargar la mascota a un paseador de perros? 45
Figura 16 Pregunta 8 ¿Estaría dispuesto a pagar a un paseador para que le lleve a su perro a pasear?.....46
Figura 17 Método Post del Módulo de Registrarse 51
Figura 18 Método Post del Módulo de Autenticación de Usuario.....53
Figura 19 Método Post del Ingreso de Roles 54
Figura 20 Feedback por parte del usuario prototipo1 55
Figura 21 Método Post de Registro Mascota 56
Figura 22 Método Post Solicitud de Paseo 58
Figura 23 Método Get para la obtención del paseador a través de las coordenadas 59
Figura 24 Método Get para la obtención de la posición del paseador a través del ID 60
Figura 25 Feedback por parte del usuario prototipo3 60
Figura 26 Pantalla autenticación de usuarios.....61
Figura 27 Pantalla autenticación de usuarios con validación de campos 61
Figura 28 Pantalla registro de usuario 62

Figura 29 Pantalla registro de usuario con validación de campos 62
Figura 30 Pantalla perfil usuario 63
Figura 31 Pantalla actualizar perfil usuario 63
Figura 33 Pantalla menú paseador 64
Figura 32 Pantalla menú dueño 64
Figura 35 Mapa localización dueño 65
Figura 34 Mapa localización paseador 65
Figura 36 Solicitudes de paseo del dueño hacia el paseador 66
Figura 37 Pantalla contraoferta del paseador para el dueño 67
Figura 38 Pantalla oferta enviada por el paseador al dueño 67
Figura 39 Pantalla paseo paseador 68
Figura 40 Pantalla paseo dueño 68
Figura 41 Pantalla calificación paseo paseador 69
Figura 42 Pantalla calificación paseo dueño 69
Figura 43 Pantalla historial de paseos 70

CAPÍTULO I: INTRODUCCIÓN

1. INTRODUCCIÓN

1.1. Justificación

El aumento de las mascotas como es el caso de los perros en el Ecuador crea conciencia sobre la importancia de los animales en la vida de las personas. Esta tendencia se debe a un cambio en la mentalidad de la sociedad, donde las mascotas son consideradas parte de la familia. Debido al entorno laboral, y el escaso tiempo que las personas pueden dedicar al cuidado de sus mascotas, se ha creado una necesidad evidente: encontrar personas dedicadas a la labor de pasear mascotas. Para abordar esta necesidad y facilitar la vida de sus dueños, surge la idea de desarrollar una aplicación que permita encontrar paseadores de confianza de manera efectiva y conveniente.

El contar con una aplicación con este fin presenta múltiples beneficios, constituye una solución eficiente para conectar dueños de mascotas con paseadores. Además, brinda acceso a personas interesadas en trabajar como paseadores o buscando un ingreso extra realizando esta actividad, ayuda a crear una comunidad de amantes de perros.

Una aplicación para pasear perros ayuda a la creciente demanda de servicios de cuidado de mascotas, es una oportunidad de aprovechar la tecnología para brindar comodidad y seguridad a los usuarios, y la posibilidad de generar ingresos. Por otro lado, este prototipo permitirá a las personas tener la confianza y poder observar en tiempo real donde se encuentra su mascota.

1.2. Planteamiento del problema

En Ecuador, se observa un elevado número de personas que comparten su vida con mascotas en entornos urbanos; de acuerdo con el último censo de población y vivienda, 4 de cada 10 hogares con niños menores a 12 años tienen perros o gatos, y, a menudo, se enfrentan a limitaciones de espacio y tiempo que les dificultan proporcionar el necesario ejercicio diario a sus perros, el cual se estima en al menos 1 hora al día. Esta carencia de actividad física impacta negativamente en la salud y bienestar de las mascotas. Además, la falta de opciones confiables para encontrar un paseador adecuado agrava aún más esta situación (INEC, 2023).

El problema se centra en la necesidad de encontrar soluciones efectivas, aprovechando la tecnología, que permitan a los dueños de mascotas garantizar que sus animales de compañía reciban el ejercicio necesario, incluso cuando su rutina diaria no lo permite. Actualmente, la falta de una plataforma o herramienta centralizada que facilite el contacto con paseadores de perros se convierte en un obstáculo significativo para que las personas puedan confiar sus mascotas a personas desconocidas.

1.3. Objetivo General

Desarrollar un prototipo de aplicación móvil para brindar una alternativa innovadora a los dueños de mascotas, a través de la conexión con paseadores de perros, de forma segura y eficiente, garantizando el bienestar de la mascota y la tranquilidad del dueño.

1.4. Objetivos Específicos

- Analizar las demandas y expectativas de los dueños de mascotas con el objetivo de mejorar la calidad de vida de los perros.
- Diseñar un prototipo de aplicación para dueños de mascotas.

- Implementar un sistema de localización en tiempo real de los paseadores utilizando la API de Google Maps, para facilitar el rastreo de las mascotas.

1.5. Antecedentes

Las mascotas han acompañado al ser humano desde tiempos muy antiguos. La domesticación de animales comenzó hace miles de años y ha evolucionado a lo largo del tiempo, brindando diferentes beneficios en la vida de sus dueños, siendo los perros los animales que mejor se han acoplado al diario vivir de las personas.

Lastimosamente los nuevos hábitos laborales, sociales, entre otros, han impedido que los dueños puedan brindar los cuidados y necesidades primordiales a sus mascotas, reduciendo el tiempo que estos pueden compartir, lo que repercute en el desarrollo de actividades físicas, salud mental, emocional y de comportamiento de los animales.

Actualmente, para mitigar estos factores existen diferentes alternativas, entre las cuales están los servicios que empresas y personas especializadas ofrecen con opciones como paseo de perros, lo que permite a sus dueños contratar este tipo de servicios que, además fomenta la generación de empleo para personas de diferentes edades que pueden aprovechar tiempos libres.

Para acceder a este tipo de servicios, se identifica la existencia de diferentes factores, que limitan el crecimiento de este modelo de negocio, como son: procesos no definidos e informales; dificultad de gestión de paseadores, dueños y mascotas; precios no normalizados acorde al mercado; inseguridad por no conocer a quienes confían sus mascotas y desconocer la ubicación de ellas; dificultad para conocer la experiencia de quien oferta los servicios, entre otros.

En virtud de lo señalado, el presente trabajo de integración curricular busca generar una alternativa, aprovechando los beneficios de la tecnología para brindar una solución confiable, práctica e innovadora para los dueños de las mascotas.

1.6. Alcance

La propuesta de este trabajo de disertación es crear un prototipo de aplicación llamada DogConnect. Esta aplicación permitirá a las personas obtener ingresos extra a través del servicio de paseo de mascotas en la ciudad de Quito, además ofrecerá a los dueños de perros una solución fácil y rápida para encontrar paseadores sin tener que recurrir a pedir favores a otras personas.

DogConnect se desarrolló utilizando la metodología ágil de prototipado evolutivo, permitiendo conectar a los dueños de perros con paseadores de manera eficiente y segura.

Este prototipo se enfocó en los 2 roles los cuales son el paseador y el dueño.

El dueño puede autenticarse, crear mascota, buscar paseador, seguimiento del paseo, envió de tarifas de ofertas de paseo, historial de paseos y calificar al paseador; mientras que el paseador podrá autenticarse, crear usuario, editar perfil, aceptar los paseos y observar el listado de los paseos realizados y calificar a la mascota al finalizar el paseo.

CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA

2. MARCO CONCEPTUAL

En este capítulo se va a describir todas las herramientas, metodologías, base de datos, lenguajes de programación que se utilizarían para este trabajo de titulación.

2.1. Metodologías de desarrollo de software

En el mundo del desarrollo de software existen varias metodologías de desarrollo de software que permiten a los profesionales estructurar, controlar el proceso de desarrollo de software.

Todos los proyectos de software se enfocan si o si en una metodología lo que permite que se desarrolle de una manera estructurada y se pueda tener al finalizar un producto de calidad.

Las metodologías de desarrollo de software han ido evolucionando a lo largo del tiempo. Pasando así de las metodologías de desarrollo tradicionales a las metodologías de desarrollo ágiles.

Hoy en día la mayoría de los equipos de desarrollo de software utilizan una metodología de desarrollo ágil, debido a que permite reducir la probabilidad de fracaso por subestimación de costos, tiempos y funcionalidades (Navarro, Fernández, & Morales, 2013).

2.1.1. Metodologías de desarrollo Tradicional

Surgen desde las primeras computadoras juntamente con los lenguajes de programación. Estas se basan en las buenas prácticas de programación que permiten tener en un marco estricto y riguroso de procesos de aplicación que garantizan un producto de software de calidad. Estas metodologías son de forma secuencial, es decir que cada etapa se basa en la anterior, es decir que son poco flexibles a cambios. Entre las metodologías tradicionales se tiene Modelo en cascada, Modelo en Espiral, Modelo en V, entre otras.

Figura 1

Metodologías Tradicionales de Desarrollo

Nota. Adaptado de *Metodologías tradicionales de desarrollo de proyectos* [Fotografía], de Diego Calvo, 2018, Diego Calvo (https://www.diegocalvo.es/metodologias-tradicionales-y-metodologias-agiles/).

2.1.1.1. Cascada

Es una metodología tradicional que se enfoca en el desarrollo secuencial de un producto de software. Proporciona una estructura de control rígida que establece que no se pueda continuar a otra fase sin haber completado al 100% la fase anterior. Esta metodología está compuesta de las siguientes fases: requisitos, diseño, implementación, pruebas, despliegue y mantenimiento (Ionos, s.f.).

2.1.1.2. Modelo en V

El modelo en V permite que el desarrollo se divida en tres partes: diseño, implementación y pruebas de integración y cualificación. Esta metodología surge de la evolución de la metodología en cascada, con la diferencia de que describe las actividades de prueba como parte integral del proceso de desarrollo de software (Aptiv, 2023).

2.1.2. Metodologías Ágiles

Hoy en día, existen varias metodologías ágiles que han surgido debido a la necesidad de mejorar los proyectos de desarrollo. Estas metodologías se basan en los requisitos del cliente, priorizando la flexibilidad y la adaptabilidad según las necesidades específicas de cada proyecto. Además, permiten ajustarse a las diferentes demandas de los proyectos tecnológicos, facilitando una respuesta rápida y eficiente a los cambios y desafíos que puedan surgir.

En la Tabla 1 se puede observar las ventajas y desventajas tanto de metodologías tradicionales como ágiles.

Metodologías Tradicionales		Metodologías Ágiles	
Ventajas	Desventajas	Ventajas	Desventajas
- Se enfoca en tener un proyecto de desarrollo documentado - Permite tener una estructura clara al	- Todas las fases son secuenciales. - Se basa siempre requisitos iniciales identificados en la fase de	- Se involucra siempre el cliente en todo el desarrollo - Se enfoca en ciclos iterativos que permite	- El cliente puede ocasionar retrasos, por la adición de nuevas funcionalidades durante el desarrollo

momento de desarrollar - Se basa en las siguientes fases: Especificación de Requisitos. Análisis. Diseño. Desarrollo. Pruebas. Implantación. Mantenimiento.	Especificación de requisitos - Es poco probable a cambios en el camino, esto quiere decir que si existe cambios se retrasará en el cronograma.	realizar entregas más rápidas - Se realizan pruebas tras cada ciclo. - Permite tener cambios sobre la marcha.	- Se puede entregar solo prototipos y nunca cerrar el proyecto. - Los equipos de desarrollo muchas veces son pequeños, algunas personas deben cubrir 2 o más roles en el desarrollo. - La documentación se descuida debido a que se centra en entregables.

Tabla 1 Ventajas y desventajas entre la metodología tradicional y ágil

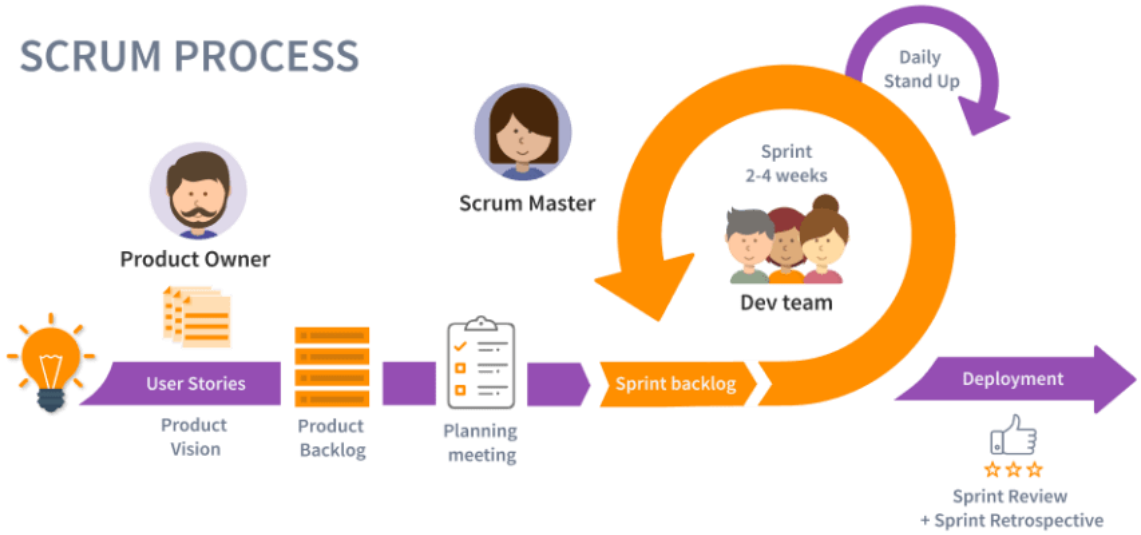
Autor. Elaborado por el Autor

2.1.2.1. Scrum

Scrum es una metodología ágil que es utilizado para trabajos en equipo, cada integrante del equipo cumple un rol en específico y el esfuerzo de todos se logra tener a un desarrollo eficiente, se basa en las entregas periódicas de avances de desarrollo aproximadamente cada 15 días, estas entregas se las conoce como Sprint y se busca sacar un producto de calidad, mitigando los problemas que se van presentando en el camino (Martins, 2024).

Figura 2

Proceso de SCRUM.



Nota. Adaptado de *Proceso de SCRUM* [Fotografía], de Gomes, 2021, DEV (https://dev.to/stanley/entrega-agil-com-scrum-4bn9).

2.1.2.2. XP

La metodología XP, conocida como Extreme Programming, se enfoca en el desarrollo rápido y ágil, con un menor énfasis en la documentación exhaustiva. Una de las ventajas de XP es su estructura en cinco fases: planificación, diseño, codificación, pruebas y lanzamiento. Para recopilar los requisitos del cliente, utiliza historias de usuario, las cuales representan todos los requisitos funcionales que se implementarán en la aplicación (Sinnaps, s.f.).

2.1.2.3. Prototipado

Esta metodología se enfoca en un desarrollo iterativo; es decir que es un desarrollo de evolución continua que siempre se trata de mejorar el producto de software, para al final poder

obtener un desarrollo de calidad. Cabe recalcar que el cliente es muy importante dentro de esta metodología debido a que siempre está en constante participación del cliente (*Prototipo*, 2021).

2.1.2.3.1. Tabla ventajas y desventajas de los prototipos

Prototipos	
Ventajas	Desventajas
- El cliente puede probar el sistema y brindar un feedback durante el desarrollo. - La utilización de la metodología permite la identificación temprana de errores o bugs que surgen durante el desarrollo. - Se puede lograr construir un prototipo operativo en semanas - A medida que se van a implementando las funcionalidades al desarrollo, los usuarios se vuelven más positivos sobre las soluciones implementadas en el prototipo.	- Siempre las iteraciones se basan en la anterior. - Existe una probabilidad de que el sistema final nunca termine y solo se quede en prototipo, debido a que pueden surgir nuevas necesidades durante del proceso de desarrollo por parte del usuario. - La documentación se descuida debido a que se centra en el desarrollo del prototipo y la implementación de las funcionalidades. - Los problemas de copia de seguridad y recuperación, rendimiento y seguridad del sistema pueden pasarse por alto en la prisa por desarrollar un prototipo

Tabla 2 Ventajas y desventajas de la metodología prototipos

Nota. Tomado de Dispositivas de la Ing. Susana Masapanta (2023).

Una vez analizado las metodologías se decidió trabajar por la metodología prototipos debido a que permite tener una mejora continua durante todo el desarrollo.

Además, no es necesario tener un equipo de desarrollo a diferencia de Scrum en el cual si es un requisito debido a que se asignan roles dentro del equipo.

2.2. Desarrollo Móvil

El desarrollo móvil es el procedimiento de crear aplicativos para smartphones, tablets o asistentes digitales, el uso más común en la industria es en sistemas operativos Android y iOS mientras que los lenguajes de programación asociados a este tipo de desarrollo son Java, Swift, C# y HTML5.

Se debe además considerar que el desarrollo móvil está creciendo exponencialmente y de hecho hoy en día las aplicaciones de este tipo multiplican su valor debido a que se han convertido en la forma más popular de acceder a Internet, es tal la demanda que, desde la distribución minorista, telecomunicaciones, e-commerce, seguros e incluso el gobierno han implementado aplicaciones para cumplir con las expectativas de los usuarios (IBM, 2023).

2.3. Back-End

Dentro del desarrollo web o móvil, el "Back-End" se encarga de la administración de la funcionalidad general de la aplicación. Cuando el usuario interactúa con el Front-End, el Back-End procesa las solicitudes realizadas en formato HTTP para devolver una respuesta. Para procesar dichas solicitudes, el Back-End interactúa con varios elementos, como servidores de bases de datos para recuperar o modificar datos relevantes, microservicios que realizan un subconjunto de las tareas solicitadas por el usuario y API de terceros para recopilar información adicional o realizar funciones adicionales (*Front End frente a back-end*, s. f.).

2.3.1. NestJS

NestJS es uno de los frameworks más populares para el desarrollo del lado del servidor en Node.js. Se destaca por su enfoque progresivo y su implementación en TypeScript, lo que garantiza una estructura sólida y un código más robusto (NestJS, s.f.).

2.3.2. Django

Es un framework utilizado para el Back-End que permite la creación de aplicaciones de manera efectiva y rápida. Django es conocido por su diseño limpio, que se centra en el desarrollo rápido y fomenta la reutilización de componentes. Estas características lo convierten en una excelente elección cuando se trata de tiempo y esfuerzo para crear aplicaciones complejas (Tokio School, 2022).

2.3.3. ASP.NET

Es un marco de desarrollo creado por Microsoft que permitir la creación de aplicaciones y servicios web dinámicos. Esta estructura utiliza el lenguaje de programación.NET y facilita a los desarrolladores la creación de sólidas aplicaciones escalables. ASP.NET es comúnmente utilizado para el desarrollo del Back-End. Brinda al desarrollador la posibilidad de funciones clave, como la administración de estados, el uso de caché, la autenticación y la autorización, y es compatible con la arquitectura de los patrones de banda y los métodos más recientemente promovidos, como MVC y Web API (Microsoft, s.f.).

2.3.4. Comparación entre Frameworks de desarrollo para el Back-End

Característica	NestJS	Django	ASP.NET
Lenguaje	TypeScript/JavaScript	Python	C#
Arquitectura	Modular y orientada a servicios	Monolítica (soporta microservicios)	Modular, soporta aplicaciones web y microservicios
Patrón de Diseño	MVC (Model-View-Controller)	MTV (Model-Template-View)	MVC (Model-View-Controller)

Compatibilidad a Bases de Datos	Compatible con cualquier base de datos mediante ORM como TypeORM	Compatibilidad a múltiples bases de datos a través de ORM como Django ORM	Compatibilidad a cualquier base de datos a través de Entity Framework
Escalabilidad	Alta, especialmente para microservicios	Alta, adecuada para aplicaciones monolíticas y microservicios	Alta, adecuada para aplicaciones grandes y microservicios
Documentación	Extensa y bien estructurada	Extensa y bien estructurada	Extensa y bien estructurada
Comunidad y Soporte	Activa, con rápido crecimiento	Muy activa y madura	Muy activa y respaldada por Microsoft

Tabla 3 Comparación entre NestJS, Django y ASP.NET

Autor: Elaborado por el Autor

Luego de haber analizado los distintos frameworks disponibles para el desarrollo del Back-End en la Tabla 3, se optó por trabajar con NestJS. La razón principal de esta elección reside en que NestJS está desarrollado en TypeScript, en el cual se maneja por TypeORM, esta es una librería que permite manipular los datos sin la necesidad de usar SQL; y además permite crear aplicaciones eficientes y escalables. Cabe recalcar que NestJS, está construido a base de Node.js, permitiendo así utilizar todas sus librerías al momento del desarrollo.

2.4. FrontEnd

En el desarrollo web, el término "Front-End" se refiere a la interfaz gráfica de usuario que permite la interacción del usuario final con los diferentes componentes de una aplicación, como menús, botones, imágenes y videos. Tres lenguajes de programación principales son fundamentales para esta interacción y la experiencia del usuario: el lenguaje de marcas de hipertexto (HTML), que define la estructura del Front-End y sus diversos elementos; las hojas de

estilo en cascada (CSS), que determinan el estilo visual de la aplicación web; y JavaScript, que añade una capa de funcionalidad dinámica a la aplicación (Amazon WS, 2020).

2.4.1. IONIC

Ionic es un framework de código abierto para construir aplicaciones móviles híbridas y posee una biblioteca que se encuentra construidos a través de estándares web utilizando HTML, CSS y Javascript. Esta biblioteca ofrece una gran variedad de componentes que se utilizan para la interfaz del usuario (Senderos, 2023).

2.4.2. Flutter

Es un framework de código abierto que fue desarrollado por Google para crear aplicaciones móviles compatibles para Android y iOS. Esta herramienta te ayuda a crear interfaces de una manera sencilla y rápida mediante el uso del lenguaje de programación dart (Amazon Web Services, s.f.).

2.4.3. React Native

Es un framework de código abierto desarrollado por Facebook que posibilita a los desarrolladores la creación de aplicaciones móviles mediante el lenguaje de programación JavaScript y React. Con ella, es posible crear aplicaciones nativas en distintas plataformas como iOS y Android, ya que el código base es en ciertos aspectos compartidos entre ambas plataformas, lo que agiliza el tiempo de desarrollo y optimiza la experiencia del usuario (Deloitte, s.f.).

2.4.4. Comparación entre Frameworks Front-End para el desarrollo móvil

Característica	Ionic	Flutter	React Native
Lenguaje	JavaScript/TypeScript	Dart	JavaScript
Componentes UI	Basados en componentes web	Widgets personalizados	Componentes nativos
Desarrollo UI	Uso de HTML, CSS y componentes web	Widgets declarativos de Dart	JSX para componentes nativos
Soporte de Plataforma	iOS, Android, Web	iOS, Android, Web	iOS, Android
Hot Reload	Sí	Sí	Sí
Documentación	Extensa y bien organizada	Extensa y bien organizada	Extensa y bien organizada
Soporte	Comunidad de Ionic	Google	Facebook
Integraciones	Fácil integración con bibliotecas JavaScript y Angular	Buenas integraciones con servicios de Google	Fácil integración con el ecosistema de React y bibliotecas JavaScript

Tabla 4 Comparación entre Frameworks de Front-End para desarrollo móvil

Autor: Elaborado por el autor

Una vez presentada en la Tabla 4 la comparación entre los frameworks para desarrollo de Front-End, se decidió trabajar con Flutter. Esta elección se basa en la necesidad de utilizar los servicios de Google Maps a través de la API, la cual permite la integración de mapas y geolocalización dentro del prototipo y trazar rutas posteriormente. Además, al ser Flutter desarrollado por Google ofrece una buena integración con estos servicios.

2.5. API

Una API o también como interfaz de programación de aplicaciones, consiste en un conjunto de reglas definidas que permiten la comunicación entre aplicaciones. Se puede decir que es una capa intermediaria que procesa las transacciones entre los diferentes sistemas.

Una API ayuda a que las empresas conecten sus aplicaciones entre sí haciendo que se ahorre tiempo y facilitando la colaboración, así como también aporta a la innovación. Las APIS deben contar también con una documentación para simplificar a los desarrolladores su implementación (IBM, 2021).

2.5.1. API REST

Una API Rest es una forma en la que cualquier aplicación con una estructura de red basada en el protocolo HTTP puede interactuar con otra aplicación. REST o Representational State Transfer es un estilo de arquitectura definida por el uso de operaciones HTTP estándar GET, POST, PUT y DELETE, para realizar operaciones sobre los recursos que son representados por las URL. Las API Rest se utilizan de manera generalizada por la simplicidad y flexibilidad, así como la escalabilidad, siendo una forma efectiva de permitir la interacción de los diversos sistemas y plataformas existentes (Red Hat, 2023).

2.6. Base de Datos

En la informática una base de datos es una colección de información que se almacena en forma de datos dentro de un sistema informático, de esta forma se organiza y administra dichos datos para un fácil acceso. Pero a medida que estas colecciones crecen se vuelve cada vez más complejo mantener esta información organizada, accesible y segura. Por dicha razón se usan sistemas de administración de bases de datos (DBMS), herramientas que permiten una correcta administración (Microsoft Azure, sf).

2.6.1. Tipos de Base de Datos

Existen 2 tipos de bases de datos: relacionales y no relacionales. Mientras que las bases de datos relacionales tienen una estructura bastante sólida y a su vez reconocen el lenguaje de consulta estructurado (SQL), las bases de datos no relacionales son diversas entre sí, ya que admiten una gran variedad de estructura de datos e incluso algunas de ellas no utilizan SQL.

Por tal motivo se puede denominar a las bases de datos relacionales como SQL mientras que a las bases de datos no relacionales como NoSQL (Microsoft Azure, s.f.).

2.6.1.1. SQL

Una base de datos SQL es aquella que permite almacenar, administrar y manipular datos estructurados de manera organizada y relacionada, y que utilizan el lenguaje de consulta SQL para realizar operaciones con los datos, desde la creación de tablas, pasando por la inserción de datos, hasta la realización de complejas consultas.

Las bases de datos SQL son aquellas que administran datos de manera estructurada, organizada y relacionada, utilizan el lenguaje de consulta SQL para poder manipular adecuadamente operaciones con dichos datos, las consultas van desde la creación de tablas, inserción de datos, modificación de campos, etc.

Algunos de los tipos más utilizados de bases de datos SQL son: MySQL, Microsoft SQL Server y PostgreSQL (Formadores IT, 2023).

2.6.1.2. No SQL

Las bases de datos NoSQL son aquellas que se han diseñado para manejar grandes volúmenes de datos que no necesariamente deben estar estructurados, es decir que esta información que se administra no sigue un esquema fijo. Los usos más comunes de este tipo

de base de datos son cuando se desea almacenar correos electrónicos o documentos, el tipo de dato es más complejo de organizar con relación a los datos estructurados.

Algunos de los tipos más utilizados de bases de datos NoSQL son: MongoDB, Cassandra, Redis y Neo4J (Formadores IT, 2023).

2.6.2. Comparación entre SQL y NoSQL

SQL		No SQL	
Ventajas	Desventajas	Ventajas	Desventajas
- Se tiene una base de datos estructurada con relaciones entre tablas, lo que permite que integridad y consistencia entre los datos. - Posee el lenguaje estructurado, el mismo que es un lenguaje estándar y muy conocido al momento de manipular los datos - Se utiliza querys para poder extraer	- Es muy estricto y requiere que existan relaciones entre las tablas. - Tiene dificultades al momento de escalar verticalmente debido a las limitaciones del hardware lo que puede llevar más gasto al momento de gestionar	- Están enfocadas para grandes cantidades de datos que no cuentan con una estructura definida - Este tipo de base de datos están enfocadas a la estabilidad horizontal, lo que conlleva el manejo de datos de una manera flexible.	- No cumplen con las propiedades de ACID, lo que conlleva a que pueda existir inconsistencia entre los datos. - No son es recomendado el uso para sistemas transaccionales debido a que no tienen una estructura definida

información de las tablas en la base de datos.			

Tabla 5 Comparación entre SQL y NoSQL

Autor: Elaborado por el Autor

Tomando en cuenta las ventajas y desventajas presentadas en la Tabla 5 de las bases de datos relacionales y no relacionales, se decidió desarrollar con una base de datos de tipo relacional. La razón principal es porque se necesita garantizar las propiedades de atomicidad, consistencia, aislamiento y durabilidad de los datos o más conocido por sus siglas ACID, lo cual es fundamental para prevenir la inconsistencia de los datos durante el proceso de desarrollo de la aplicación.

2.6.3. PostgreSQL

Es un sistema de gestión de bases de datos relacional de código abierto, conocido por su estabilidad, extensibilidad y estándares SQL avanzados. PostgreSQL ha tenido un enfoque en todo caso desde sus inicios y es muy configurable y personalizable. Este motor de base de datos puede manejar los flujos de trabajo más críticos, desde cargar archivos multimedia hasta las cargas de trabajo más críticas de todo el conjunto de datos de la empresa. Sus características avanzadas incluyen soporte para tipos de datos personalizados, índices avanzados, y ser cumplir con concurrencia y transacciones ACID (Platzi, 2015).

2.6.4. Microsoft SQL SERVER

Es un sistema de gestión de datos relacional creado por Microsoft. SQL Server está diseñado para realizar tareas que van desde el manejo de aplicaciones de escritorio a pequeñas y medianas empresas, a grandes aplicaciones de centro de datos para organizaciones y empresas. SQL Server es una base de datos de software que permite a los desarrolladores y administradores de base de datos hacer tareas. También viene con herramientas como servicios

de análisis e integración, herramientas de generación de informes, capacidad de almacenamiento temporal, y tiene soporte con.NET Middleware de desarrollo (Pérez, 2021).

2.6.5. MYSQL

Es un tipo de base relacional que utiliza lenguaje SQL, para gestionar y acceder a bases de datos. Debido a su velocidad, confiabilidad y facilidad de uso, es un sistema popular entre aplicaciones web y servicios en línea. MySQL se ha creado para administrar grandes volúmenes de información, y funciona bien en muchos sistemas operativos. Además, cuenta con transacciones ACID, funciones de replicación y un conjunto de motores de almacenamiento que permite implementar motores óptimos según la situación (Dominguez, 2014).

2.6.6. Comparación entre motores de base de datos relacionales

	MySQL	PostgreSQL	Oracle
Ventajas	Es de código abierto, fácil de usar, altamente escalable, amplia comunidad de usuarios ofrece una alta disponibilidad	Es de código abierto, muy seguro, bueno para manejar grandes conjuntos de datos, gratuito	Conjunto de funciones muy completo, excelente seguridad, el mejor rendimiento para grandes conjuntos de datos, soporte de alto nivel
Desventajas	Es menos seguro que PostgreSQL y Oracle, no es tan bueno para manejar grandes conjuntos de datos.	Es más complicado escalar de escalar a diferencia de MySQL, la comunidad de usuarios no es tan grande como la de MySQL.	Costos de licenciamiento altos, y demasiada dependencia del proveedor. La comunidad no es tan amplia

Tabla 6 Comparación entre motores de base de datos relacionales

Autor: Elaborado por el Autor

Tomando como referencias las ventajas y desventajas de los motores de base de datos presentadas en la Tabla 6, se procedió a seleccionar a MySql como de base de datos para este proyecto de desarrollo. Esta decisión se basó debido a que ofrece una facilidad de uso y configuración al momento de instalar. Además, las librerías de NestJS permiten conectar de una manera rápida y sencilla el Back-End a la base de datos.

CAPÍTULO III: METODOLOGÍA DE DESARROLLO

3. Metodología de desarrollo del aplicativo

Una vez presentado la comparación de las metodologías de desarrollo en la Tabla 1. Se seleccionó la metodología prototipado evolutivo. Esta metodología permite un desarrollo de forma iterativa, el cual permite ir implementando funcionalidades de una manera rápida y corrigiendo fallos que se van presentando en el camino.

3.1. Prototipado Evolutivo

La metodología de prototipado evolutivo permite desarrollar rápidamente un producto de software, implementando desde el inicio las funcionalidades principales. Esta metodología se basa en los comentarios del cliente, ya que el desarrollo avanza en función de estos. La principal característica de esta metodología es que el tiempo de desarrollo es más corto, lo que permite realizar pruebas rápidamente. Además, el prototipo inicial debe contener las características básicas de la aplicación para poder evaluar su correcto funcionamiento, y en base a esto, se realizan cambios continuos gracias a los comentarios de los usuarios.

Esta metodología cuenta con las siguientes fases:

Figura 3

Modelo de Prototipado Evolutivo

Nota. Adaptado de Modelo de Prototipado Evolutivo [Fotografía], de Sequeda, 2015 (https://www.researchgate.net/figure/Modelo-de-Prototipado-Evolutivo-aplicando-refinamientos-sucesivos-Gonzalez-10_fig1_280090865)

Cada fase es esencial para el proceso de la construcción de los prototipos.

3.1.1. Análisis de Requerimientos

En esta fase de análisis de requerimiento se recopila toda la información proporcionada por el cliente, esto es debido a que se debe recabar los requisitos iniciales que debe tener el prototipo inicial.

3.1.2. Diseño y desarrollo del prototipo

En esta fase se centra en diseñar y desarrollar las funcionalidades principales identificadas en la fase de requerimientos. Se procede a crear o diseñar un boceto de cómo quedará la aplicación para su posterior desarrollo.

3.1.3. Pruebas del prototipo

Una vez que el diseño y el desarrollo del prototipo están completos, se deben realizar las pruebas correspondientes. Esto permite identificar y solucionar errores o bugs que puedan surgir. Además, se verifica la funcionalidad del prototipo para asegurar que cumple con los requisitos establecidos.

3.1.4. Retroalimentación del prototipo

Una vez realizadas las pruebas, se presenta el prototipo al cliente para obtener sus comentarios y retroalimentación. Esto es crucial, ya que el feedback del cliente permite mejorar continuamente el prototipo.

3.1.5. Refinamiento del prototipo

Una vez realizadas las pruebas, en la fase de refinamiento, se debe presentar el prototipo al cliente para obtener sus comentarios y retroalimentación. Esto es fundamental, ya que el feedback del cliente permite seguir mejorando el prototipo continuamente.

4. DESARROLLO

4.1. Estándares de desarrollo

4.1.1. Configuración de la base de datos

En este punto se procedió realizar la conexión a la base de datos mediante el Back-End NestJS hacia la base de datos MySQL. La misma que fue elegida anteriormente como gestor de base de datos. Así mismo la conexión permitirá crear las tablas de manera dinámica desde el Back-End hasta la base de datos. Para realizar este proceso se utilizaron los siguientes parámetros de conexión: tipo, host, puerto, usuario, contraseña y nombre de la base de datos.

Figura 4

Conexión a Base de Datos

```
app.module.ts > AppModule
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { TypeOrmModule } from '@nestjs/typeorm';
import { UsersModule } from './users/users.module';
import { AuthModule } from './auth/auth.module';
import { JwtModule } from '@nestjs/jwt';
import { jwtConstants } from './auth/jwt/jwt.constants';
import { RolesModule } from './roles/roles.module';
import { SocketModule } from './socket/socket.module';
import { WalkersPositionModule } from './walkers_position/walkers_position.module';
import { OwnRequestsModule } from './own_requests/own_requests.module';
import { TimeAndDistanceValuesModule } from './time_and_distance_values/time_and_distance_values.module
import { WalkerTripOffersModule } from './walker_trip_offers/walker_trip_offers.module';
import { DogModule } from './dog/dog.module';

@Module({
  imports: [
    TypeOrmModule.forRoot({
      type: 'mysql',
      host: 'localhost',
      port: 3306,
      username: 'root',
      password: 'password',
      database: 'flutter_perro',
      entities: [__dirname + '/**/*.entity{.ts,.js}'],
      synchronize: true,
    }),
```

En la

Figura **4** se puede mostrar la función TypeORM que permite conectarse a la base de datos, en donde se solicita los parámetros mencionados anteriormente como clave valor, lo cual facilita la conexión a la base de datos y la creación de entidades de manera sencilla.

Figura 5

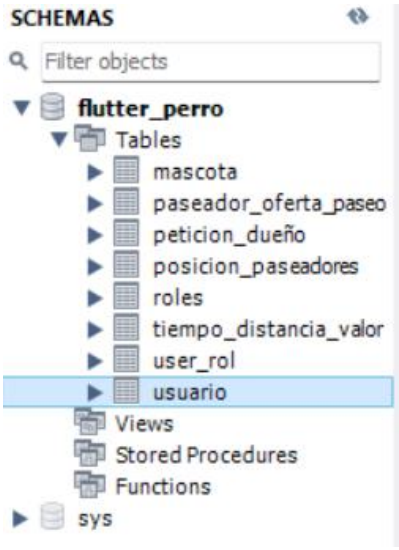
Configuración de la ruta de la conexión Back-End

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { ValidationPipe } from '@nestjs/common';
import * as bodyParser from 'body-parser';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  app.use(bodyParser.json({ limit: '50mb' }));
  app.use(bodyParser.urlencoded({ limit: '50mb', extended: true }));

  app.useGlobalPipes(new ValidationPipe({ forbidUnknownValues: false }));
  app.enableCors();
  await app.listen(3000, '192.168.1.14' ? '192.168.1.14' : 'localhost');
}
bootstrap();
```

En la *Figura 5* se muestra la configuración del Back-End, donde se detalla la IP para realizar los consumos desde el Front-End. Esta configuración permite acceder a todas las operaciones gestionadas por el controlador, facilitando el acceso y la edición de la información en la base de datos.

Figura 6

Tablas de la Base de Datos



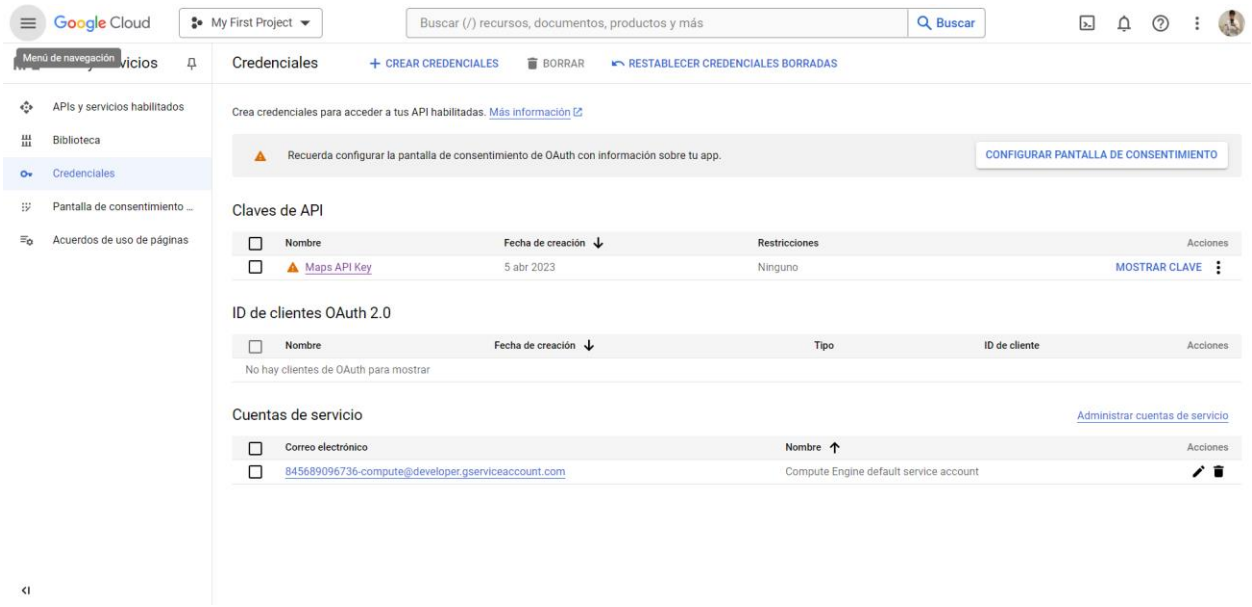
Como se puede apreciar en la Figura 6, se puede observar las tablas que fueron creadas desde el Back-End hacia la base de datos. Esto fue gracias a las entidades que fueron declarados en cada clase.

4.1.2. Configuración de la API Google Maps

Para activar los servicios de Google Maps, es necesario contar con una cuenta de Gmail para acceder a la consola de Google Cloud. En esta plataforma, se encuentran disponibles diversas APIs, entre las cuales se enfocó específicamente en activar la API de navegación. Esta API permitirá la implementación y trazado de rutas en el prototipo móvil, facilitando así la funcionalidad de navegación dentro de la aplicación.

Figura 7

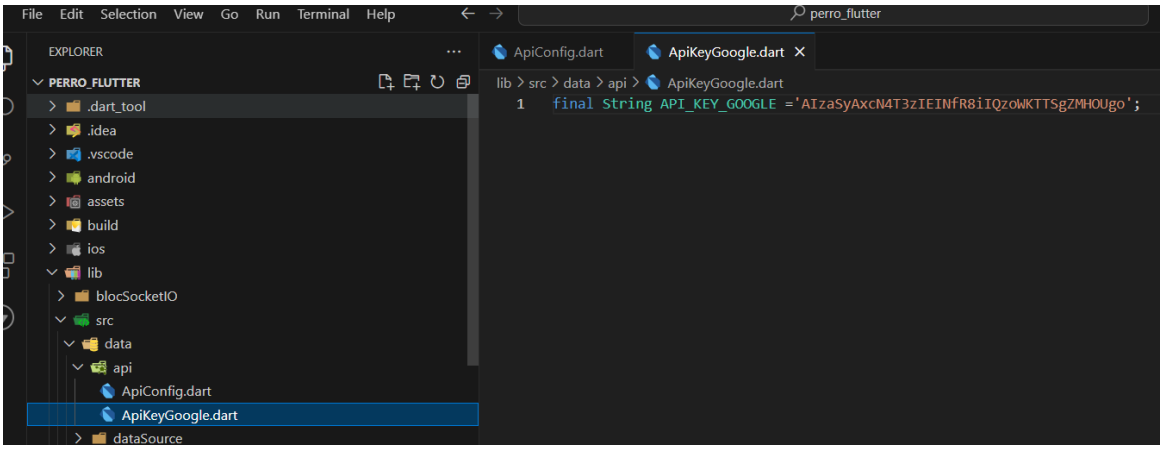
Consola API Google



En la Figura 7 se puede observar la consola de Google, en la cual se realizó la activación de un Api Key, este es un valor único que proporciona Google para poder utilizar sus servicios en la parte del Back-End. Con dicho valor permitirá acceder a la API de Google Maps.

Figura 7

Generación API Google Maps



En la parte del Front-End se realizó la configuración del API de Google Maps. Se creó una carpeta en la que se va a almacenar todas las APIS que se utilizará durante el

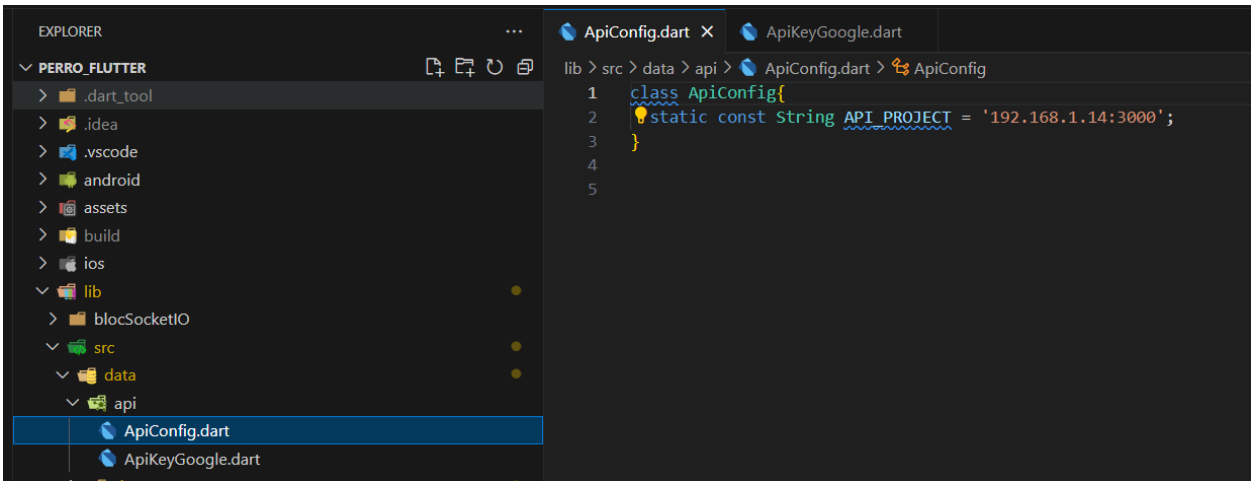
desarrollo. En este ejemplo se realizó la creación del archivo ApiKeyGoogle.dart, en la misma que se almacena la clave del API que fue generada anteriormente en la consola de Google y permitirá acceder a sus servicios.

4.1.3. Configuración conexión desde el Fron-End hacia el Back-End

Para poder realizar la conexión desde el Front-End hacia el Back-End, se debe crear un archivo en el que le llamaremos ApiConfig.dart dentro de la carpeta llamada API. Este archivo tendrá la ruta del Back-End en donde se podrá acceder desde la interfaz.

Figura 8

Conexión desde el Front-End con el Back-End



Como se puede observar en la Figura 8, se declaró una variable constante de tipo String dentro de la clase ApiConfig, donde se especifica la ruta del API que permitirá la conexión con el Back-End.

4.2. Análisis de Requerimientos

Para poder realizar el desarrollo del prototipo, fue importante analizar y comprender las demandas y expectativas de los propietarios de mascotas con el objetivo de mejorar la calidad de vida de los perros. Para ello, considerando que no existe un cliente específico ni una empresa

que haya solicitado este servicio en particular, se consideró necesario diseñar y aplicar una encuesta, que ayude a identificar los requerimientos funcionales y no funcionales, en base a los cuales se desarrollaría la aplicación. Para poder realizar la encuesta, lo primero que se identificó fue el sector donde se puede encontrar dueños de mascotas y paseadores; y él sitio más adecuado que se encontró fue el parque de la Carolina. Posteriormente en base a la población que visita el parque, se definió una muestra de 21 personas con edades comprendidas entre los 20 y 50 años. Este tamaño de muestra fue determinado considerando la población de residentes que viven alrededor del parque y que regularmente pasean con sus mascotas en el parque de la Carolina. Los resultados de la tabulación de la encuesta, permitió identificar los requerimientos necesarios para el prototipo.

Las preguntas que se formularon en la encuesta para el levantamiento de requerimientos son las siguientes:

Figura 9

Pregunta 1 ¿Vives en casa o departamento?

En la Figura 9 se puede observar que el 52,6 % de las personas encuestadas viven en un departamento, mientras que el 47,6 % viven en una casa. Esta diferencia se explica por la escasez de casas en el Barrio La Carolina, donde predominan los edificios. La mayoría de los jóvenes prefieren vivir en ese sector debido a que se encuentran en un lugar céntrico de la ciudad, donde tienen acceso a todas las facilidades y servicios.

Figura 10

Pregunta 2 ¿Dispone de un espacio físico grande para su mascota?

En la Figura 10 se obtuvo un total de 66,7 % personas no cuentan con un espacio grande para que la mascota realice actividad física, mientras que el 33,3% de las personas si cuentan con un espacio físico grande. Esta diferencia se debe a la escasez de viviendas en el Barrio La Carolina que ofrecen áreas verdes y amplias para sus mascotas; la mayoría de las casas y departamentos no tienen espacio suficiente para que puedan los perros ejercitarse adecuadamente.

Figura 11

Pregunta 3 ¿Con qué frecuencia saca a pasear a su mascota?

El 47.6 de las personas nunca pasean o pasean 1 vez a la semana a su mascota. Esta tendencia se debe porque la mayoría de los jóvenes cuentan con un empleo y tienen la mentalidad orientada hacia el trabajo y crecer económicamente, por lo que limita el tiempo disponible para poder dedicar a sus mascotas.

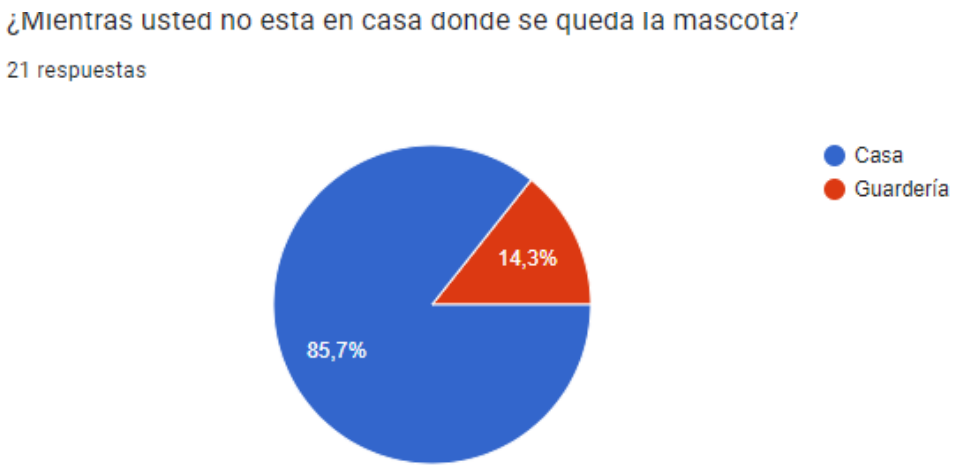
Figura 12

Pregunta 4 ¿Cuánto tiempo a la semana dedica para que su mascota haga actividades físicas o de recreación?

En la Figura 12 se puede observar que se obtuvo que el 28,6% de las personas dedican más de 5 horas para que su mascota realice actividades físicas o de recreación. Además, un 23,8% de las personas dedican 1 hora semanal, otro 23,8% dedican 5 horas y el 23,8% restante nunca dedican tiempo para estas actividades. En vista del resultado, las personas que dedican más de 5 horas representan el mayor porcentaje, lo que podría indicar una tendencia hacia la valoración de la importancia de la actividad física y recreativa para las mascotas.

Figura 13

Pregunta 5 ¿Mientras usted no está en casa donde queda la mascota?



En la Figura 13 el 85,7% de las personas le dejan a su mascota en la casa mientras que el 14,3 % de las personas dejan su mascota en una guardería. Este porcentaje se debe a que el servicio de guardería para mascotas puede resultar muy costoso para muchos dueños, lo que lleva a que la mascota permanezca en casa mientras el dueño está en el trabajo.

Figura 14

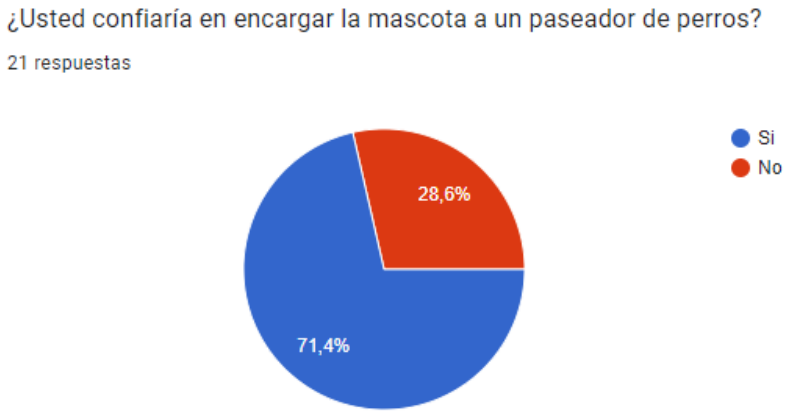
Pregunta 6 ¿Contrata algún tipo de servicio para que su mascota realice actividad física?

En Figura 14 se puede observar que el 81% de las personas encuestadas no contratan ningún tipo de servicio para sus mascotas, mientras que el 19% sí contrata algún servicio para

que sus mascotas realicen actividad física. Este porcentaje se debe a que muchos de estos servicios, como las guarderías, son económicamente costosos, y son pocas personas las que optan por contratarlos.

Figura 15

Pregunta 7 ¿Usted confiaría en encargar la mascota a un paseador de perros?



En la Figura 15, se obtuvo como resultado que el 71,7% de las personas encargarían a su mascota a un paseador de perros, mientras que el 28,6% de las personas no encargarían a su mascota al paseador de perros. Esta tendencia se debe a que existen en la ciudad de Quito personas informales que realizan este servicio las cuales no presentan garantías suficientes para que los dueños de mascotas puedan confiar en ellas.

Figura 16

Pregunta 8 ¿Estaría dispuesto a pagar a un paseador para que le lleve a su perro a pasear?

En la Figura 16 se puede observar que el 66.7 % de las personas encuestadas estarían dispuestos a pagar a una persona que saquen al perro a pasear, mientras que el 33,3% no pagarían a un paseador para que le lleve a su perro a pasear. Este resultado se debe a que otros servicios son muy costosos y no ofrecen las garantías suficientes para que los dueños puedan confiar. Con este prototipo de aplicación permite ver en tiempo real la ubicación de la mascota, la mayoría de las personas encuestadas estarían dispuestas a pagar por el servicio, ya que ofrece seguridad al tener la capacidad de rastreo mientras dure el paseo.

Basándonos en estos resultados se procedió a determinar los siguientes requerimientos.

4.2.1. Levantamiento de requerimientos

4.2.1.1. Historias de Usuario

Para el prototipo de mascotas, se determinó las siguientes historias de usuarios, que va a tener el prototipo de aplicación. Estas historias de usuario son las siguientes:

Nombre de la historia de usuario	Módulo
Autenticarse en el sistema	Autenticación y acceso
Registro de Usuario	
Administración de mascota	Gestión de mascota
Solicitud de Paseo	Gestión de paseo
Ubicación en Tiempo Real	
Calificación de Paseo	
Historial de Paseo	

Tabla 7 Listado de historias de usuario.

Autor: Elaborado por el Autor

Estas serán todas las funciones que realizará el prototipo de aplicación móvil. En total se obtuvieron 7 historias de usuarios.

Historia de usuario Nº1 – Autenticación en el sistema

Historia de Usuario	
Número: 1	**Usuario:** Administrador, paseador, dueño
Nombre de la historia: Autenticación en el sistema	
Prioridad del negocio: Alta	**Riesgo en Negocio:** Alta
Programador responsable: Jorge Andrade	
Descripción: Los usuarios mencionados deberán autenticarse para poder utilizar la aplicación de mascotas.	
Observaciones: Una vez iniciada la sesión solo se puede salir si el usuario presiona el botón salir.	

Tabla 8 Historia de usuario: Autenticación en el sistema.

Autor: Elaborada por el Autor

4.2.1.1.1. Historia de usuario Nº2 – Administración de Mascota

Historia de Usuario	
Número: 2	**Usuario:** dueño
Nombre de la historia: Administración de mascota	
Prioridad del negocio: Alta	**Riesgo en Negocio:** Alta
Programador responsable: Jorge Andrade	
Descripción: El dueño de la mascota solo puede crear a su mascota.	
Observaciones: Las funcionalidades crear a la mascota solo puede realizar el dueño de esta.	

Tabla 9 Historia de usuario: Administración de mascota.

Autor: Elaborado por el Autor

4.2.1.1.2. Historia de usuario Nº3 – Solicitud un paseo de mascota:

Historia de Usuario	
Número: 3	**Usuario:** dueño
Nombre de la historia: Solicitud de paseo de mascota	
Prioridad del negocio: Alta	**Riesgo en Negocio:** Alto
Programador responsable: Jorge Andrade	
Descripción: El dueño puede realizar una solicitud de paseo para su mascota	
Observaciones: En la aplicación, el dueño puede solicitar un paseo especificando el lugar al que desea que lleven a su mascota.	

Tabla 10 Historia de usuario: Solicitud de paseo de mascota

Autor: Elaborado por el Autor

4.2.1.1.3. Historia de usuario Nº4 – Ubicación en Tiempo Real

Historia de Usuario	
Número: 4	**Usuario:** dueño, paseador
Nombre de la historia: Ubicación en Tiempo real para el dueño y paseador.	
Prioridad del negocio: Alta	**Riesgo en Negocio:** Alto
Programador responsable: Jorge Andrade	
Descripción: Los usuarios puede ver la ubicación en tiempo real y la ruta del paseo.	
Observaciones: En la aplicación se trazará la ruta que debe seguir para poder llegar al lugar seleccionado por el dueño de la mascota.	

Tabla 11 Historia de usuario: Ubicación en tiempo real para el dueño y la mascota

Autor: Elaborado por el Autor

4.2.1.1.4. Historia de usuario Nº5 – Calificación de Paseo

Historia de Usuario	
Número: 5	**Usuario:** dueño, paseador
Nombre de la historia: Calificación de paseo mascota.	
Prioridad del negocio: Media	**Riesgo en Negocio:** Media
Programador responsable: Jorge Andrade	
Descripción: Los usuarios pueden calificar el paseo una vez que este termine.	
Observaciones: En la aplicación el dueño puede dar una calificación al paseador por medio de la aplicación y además el paseador también puede calificar a la mascota.	

Tabla 12 Historia de usuario: Calificación de Paseo.

Autor: Elaborado por el Autor

4.3. Desarrollo de Prototipos

4.3.1. Prototipo de Autenticación

Dentro del primer prototipo se encuentra contemplado que se va a desarrollar los módulos de autenticación y acceso. Este módulo representa la historia de usuario autenticación, creación de roles y el registro del usuario.

4.3.1.1. Desarrollo Prototipo Autenticación

Para esta etapa se utilizó los frameworks Flutter y NestJS. El framework Flutter permitió crear la parte de la interfaz. Se implementaron un total de 3 pantallas en el Front-End; una de las pantallas se realizó para la creación de los usuarios, otra para la autenticación de usuarios y por último la pantalla de seleccionar el tipo de rol del usuario. En cambio, el framework NestJS se

utilizó para la parte del Back-End, es decir para la creación de los servicios o APIS, que permitan realizar todas las operaciones necesarias que se necesita para las funcionalidades del primer prototipo. Para el almacenamiento de la contraseña, se utilizó la librería Bcrypt, la misma que permite encriptar la contraseña para su posterior almacenamiento en la base de datos. Esto es una práctica de seguridad, ya que permite tener una gestión segura de contraseñas.

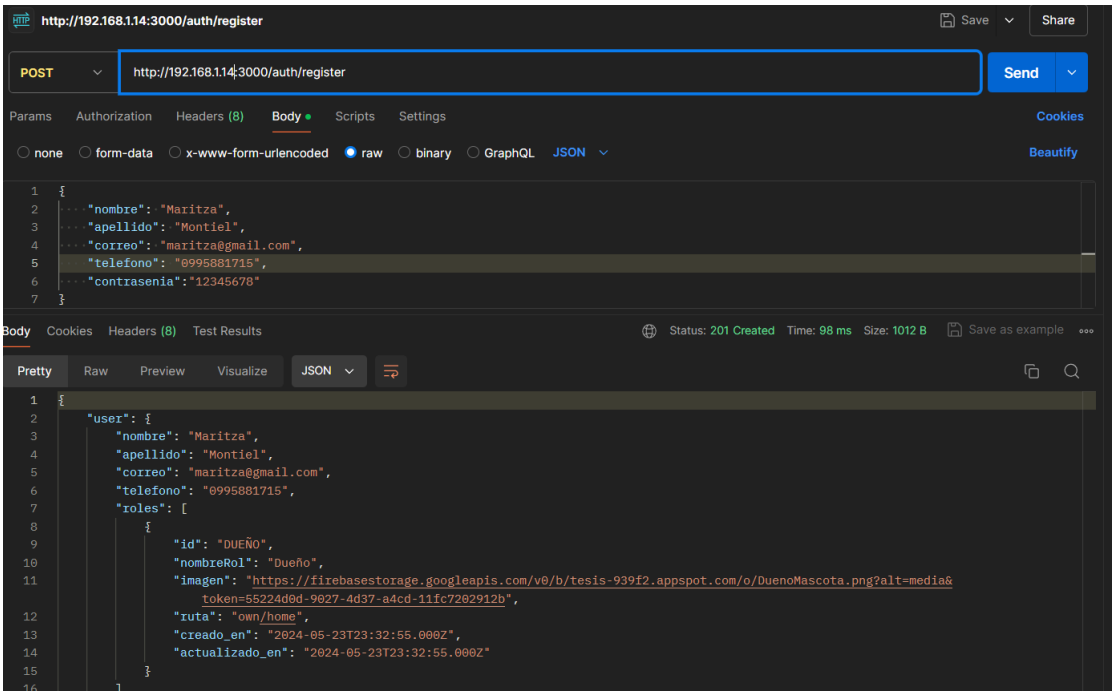
4.3.1.2. Pruebas

Se realizaron las pruebas de integración, las cuales permiten verificar que los diferentes módulos funcionen correctamente. En este caso se utilizó la aplicación PostMan para probar los servicios de creación y autenticación de usuario. Estos son métodos Post, que se utiliza para enviar datos al servidor para su procesamiento.

En la Figura 17 se observa se realiza la petición post hacia la ruta del Back-End definido en el controlador. Se enviaron los siguientes datos en formato JSON, los cuales son nombre, apellido, correo, teléfono y contraseña. Los mismos que se caracterizan por clave valor.

Figura 17

Método Post del Módulo de Registrarse

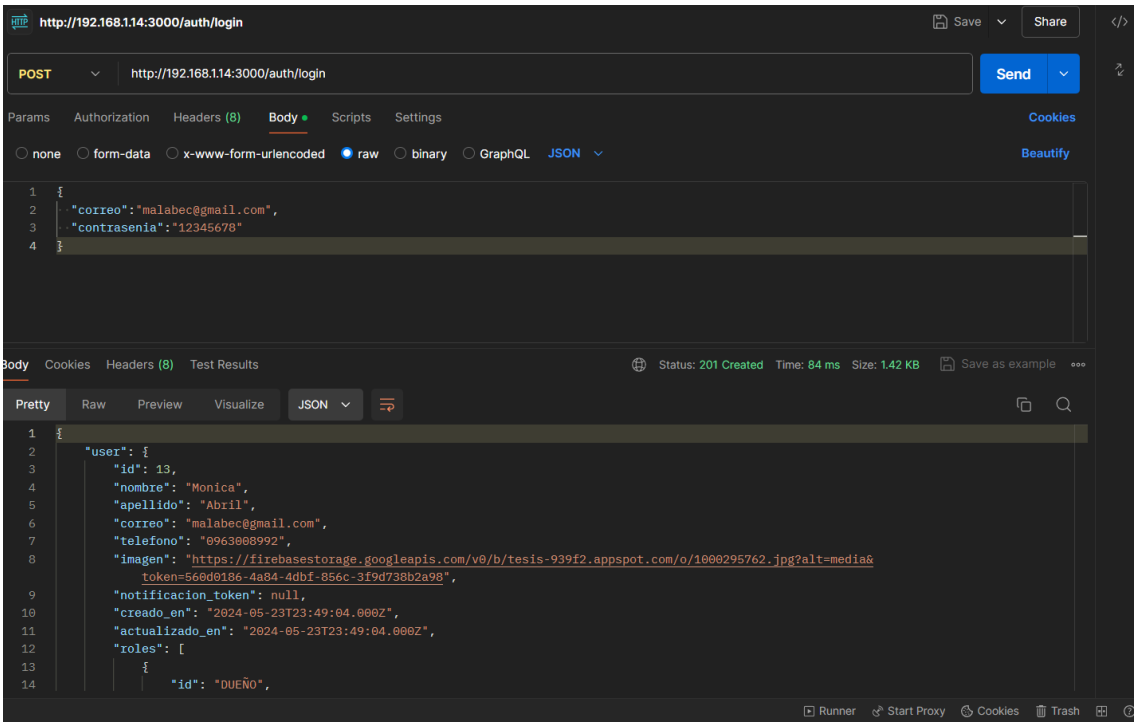


Una vez enviado los datos se tuvo como respuesta un archivo JSON. El mismo que devolvió la información que se ingresó a la base de datos. Se observa además que se le asigna el rol por defecto que es dueño, esto se debe a que se encuentra configurado en el Back-End solo el rol administrador pueda designar más de un rol a los diferentes usuarios.

Para el siguiente caso se realizará la autenticación de usuario y contraseña. Para esto se procedió a apuntar a la ruta descrita anteriormente en el controlador, la misma que recibe los campos correo y contraseña en formato JSON. En el controlador se encuentra definido que se encripte la contraseña al momento de ingresar la petición, esto se realiza debido a que es una práctica de seguridad de no almacenar texto plano en la base de datos.

Figura 18

Método Post del Módulo de Autenticación de Usuario

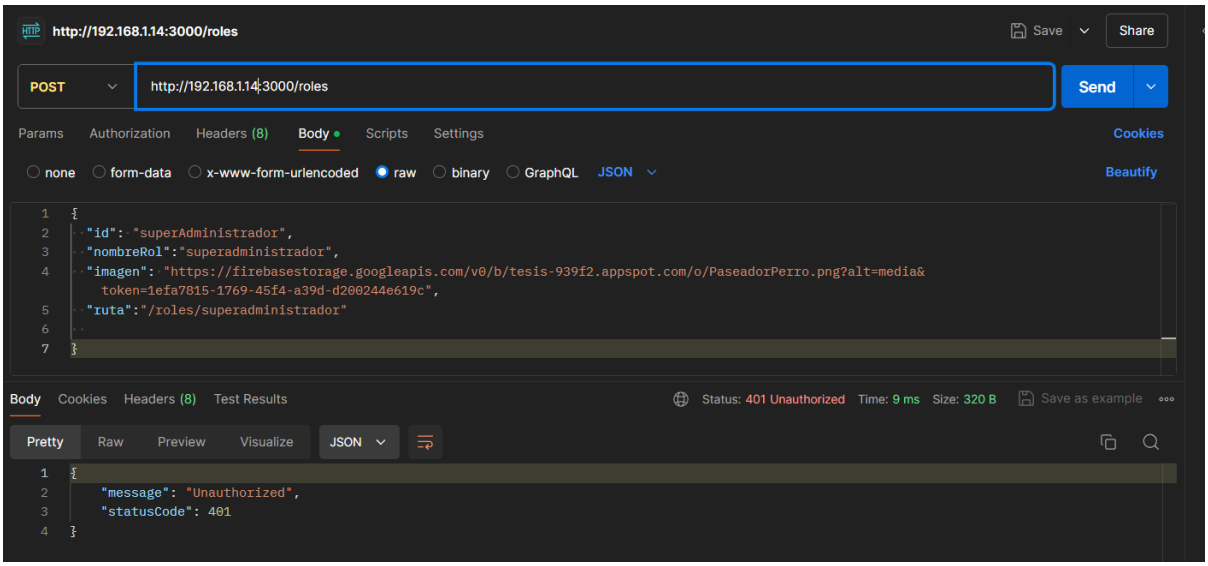


En la Figura 18 se observa que se tiene como respuesta la información ingresada en formato JSON. Esta respuesta se obtiene cuando el proceso de ingreso se realizó correctamente. Además, se puede observar que se retorna toda la información que se encuentra almacenada de dicho usuario en la base de datos.

Para el siguiente caso se ingresa los campos de id, nombreRol, imagen, ruta. Estos campos pertenecen a la entidad rol que se encuentra en la base de datos. Se realiza la petición POST para el ingreso apuntando a la ruta definida anteriormente en el controlado como muestra la Imagen.

Figura 19

Método Post del Ingreso de Roles



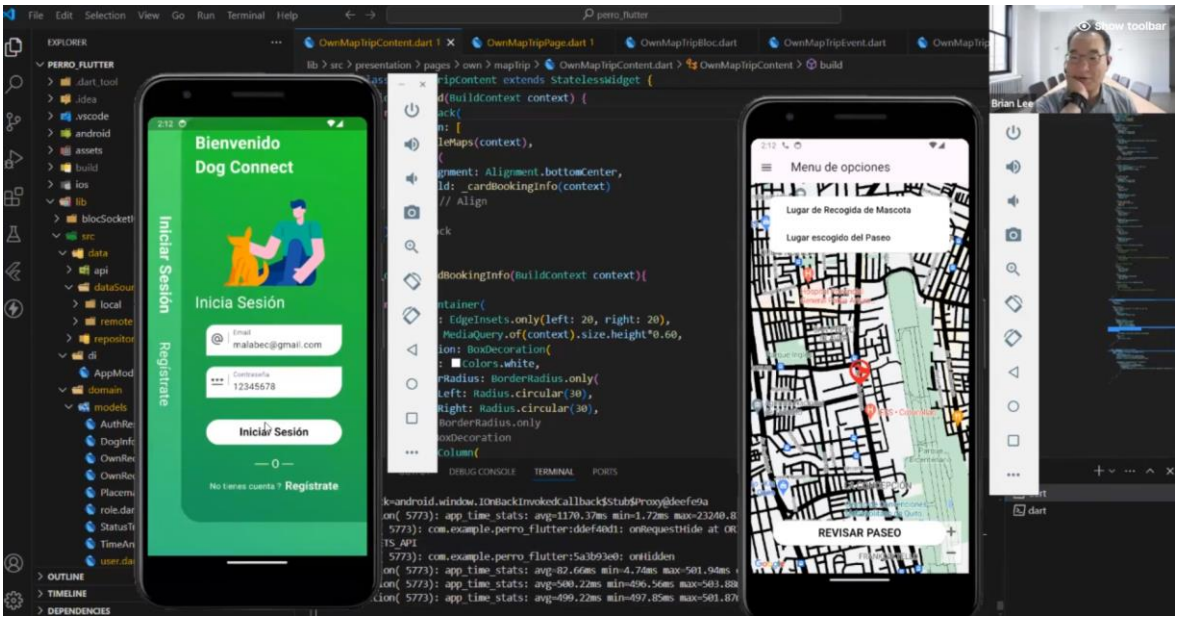
Sin embargo, se obtuvo una respuesta Unathorized, la cual menciona que no se puede ingresar dicha información a la base de datos. Esto es debido a que se encuentra protegida la ruta de que solo se ingrese siempre y cuando el usuario se encuentra autenticado y posee el rol de administrador.

4.3.1.3. Feedback del Usuario

Para la retroalimentación del prototipo se realizaron consultas con personas de Korea mediante reuniones en línea, los cuales son especializados en aplicaciones móviles. Los mismos que sugirieron que se implementen ciertas condiciones para poder validar los campos al momento de ingresar los datos. Este comentario fue aceptado por mi persona y se realizará en el refinamiento del prototipo 1.

Figura 20

Feedback por parte del usuario prototipo1



4.3.1.4. Refinamiento del Prototipo

Con base en el feedback recolectado anteriormente, se realizaron las correcciones propuestas, para poder así validar que la información que ingrese el usuario sea correcta. Se implementó tanto en el Back-End como en el Front-End dichas validaciones para validar que no ingrese datos duplicados o erróneos.

4.3.2. Prototipo de Gestión de Mascota

Dentro del segundo prototipo se encuentra contemplado que se va a centrar en el desarrollo de la gestión de mascota. Este módulo representa la historia de administración de la mascota, en el que se centra en el ingreso de mascota.

4.3.2.1. Desarrollo Prototipo Gestión Mascota

Para esta etapa, se utilizaron los frameworks Flutter y NestJS. Flutter facilitó la creación de la interfaz de usuario, incluyendo una pantalla en el Front-End para el ingreso de mascotas. Se diseñó la entidad "mascota" en el Back-End con campos esenciales como id_dueño, nombre, raza y color, implementados mediante TypeORM. En el controlador correspondiente, se desarrolló un servicio para gestionar el ingreso de mascotas. Para garantizar la seguridad, solo los usuarios autenticados con el rol de dueño pueden realizar este proceso, mediante una solicitud POST definida en el controlador de mascotas.

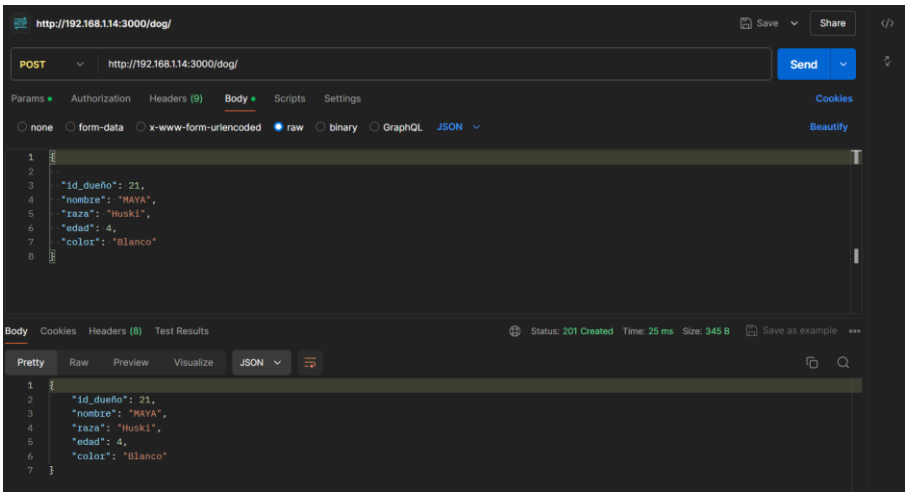
4.3.2.2. Pruebas

Se realizaron las pruebas de integración, las cuales permiten verificar que el módulo de gestión de mascota funcione correctamente. Para esto se utilizará la herramienta PostMan que permite realizar pruebas del servicio de mascota configurado anteriormente en el controlador. Se debe pasar la ruta del controlador para poder realizar dicha petición de tipo POST.

Para este caso de prueba se utilizó el dueño 21, el cual ya se encuentra registrado dentro de la aplicación. Se le añadió una mascota con el método POST. En el cual se especificaron los campos en formato JSON, con su respectiva clave valor. Para esta demostración se añadió la siguiente información como se puede observar en la Figura 21.

Figura 21

Método Post de Registro Mascota



Finalmente se obtuvo el mismo objeto en formato JSON como respuesta del servidor, lo que quiere decir que se realizó correctamente el ingreso de la información de mascota a la base de datos.

4.3.2.3. Feedback del Usuario

En el feedback del usuario se realizó la misma demostración a las personas mencionadas anteriormente, pero no se obtuvo una retroalimentación de mejora como tal, por lo tanto, se procederá con el siguiente prototipo.

4.3.2.4. Refinamiento

Dado que no se obtuvo un feedback por parte del usuario, no se procederá a realizar el refinamiento, por lo que se continuará con la siguiente interacción.

4.3.3. Prototipo de Gestión de Paseo

Dentro del tercer prototipo se encuentra contemplado que se va a centrar en el desarrollo de la gestión del paseo. En este módulo se representa las siguientes historias de usuario como son: Solicitud de Paseo, Ubicación en Tiempo Real, Calificación de Paseo, Historial de Paseo.

4.3.3.1. Desarrollo Prototipo Gestión de Paseo

Para esta etapa se utilizó los frameworks Flutter y NestJS. El framework Flutter permitió crear la parte de la interfaz. Se implementaron un total de 8 pantallas en el Front-End; las cuales 4 pertenecen al rol de paseador y las otras 4 pertenecen al rol de dueño. En estas pantallas se encuentran contemplados el proceso de solicitud de paseo, el historial de paseos realizados, la negociación de la oferta del paseo que puede darse entre el paseador y el dueño de mascota y el seguimiento en tiempo real de ambas partes.

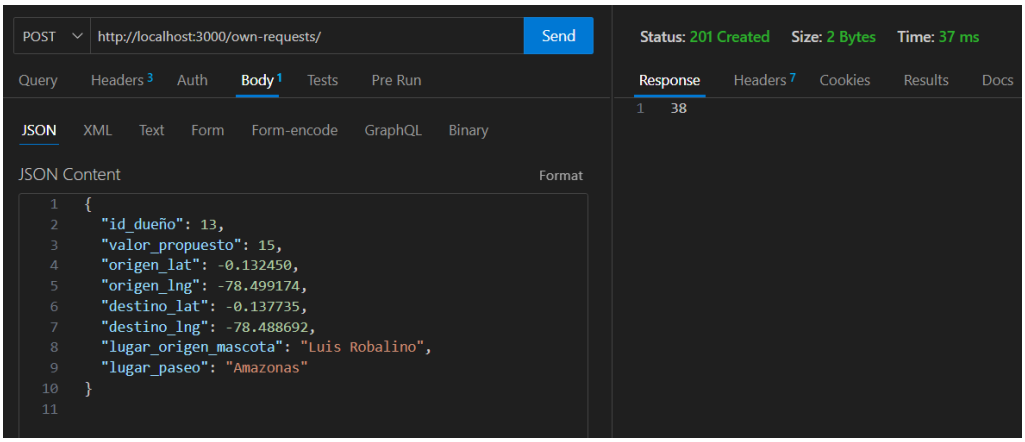
4.3.3.2. Pruebas

Se realizó pruebas de integración, las cuales permiten verificar que el módulo de solicitud de paseo funcione correctamente. Para esto se utilizarán las herramientas PostMan y Thunderclient, las cuales permiten realizar pruebas de los servicios del paseo.

Para este caso de prueba se realizará la solicitud de paseo por medio del método POST. Para esto se debe tener la ruta a donde se quiere apuntar en este caso se va a apuntar a own-request que son las peticiones del dueño realizadas. Esta ruta se encuentra especificada en el controlador. Para esto se requiere que se envíe datos en formato JSON como muestra la Figura 22

Figura 22

Método Post Solicitud de Paseo

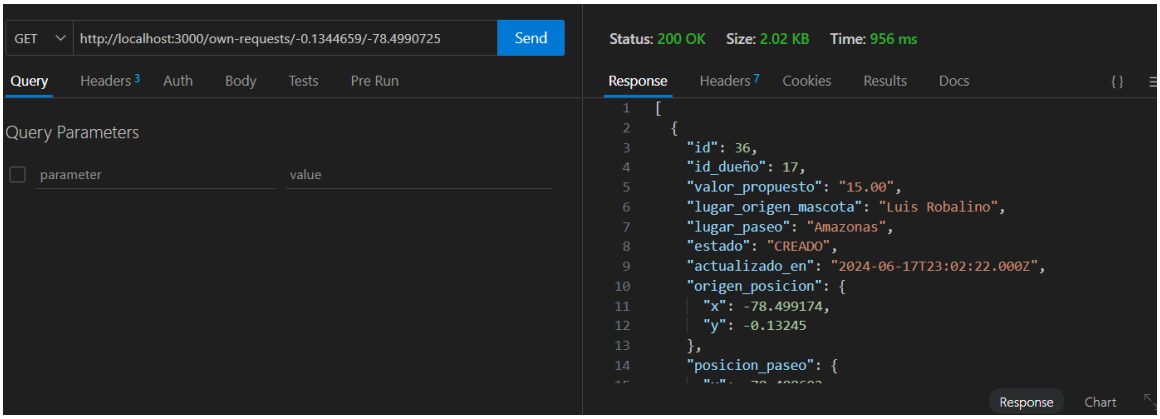


En la Figura 22 **¡Error! No se encuentra el origen de la referencia.** se observa que se ingresa los datos en formato clave valor, para esto se debe tener estar registrado al usuario dentro de la aplicación. Como se puede observar se obtuvo una respuesta 201 que menciona que la información ha sido ingresada correctamente a la base de datos.

Para el siguiente caso prueba se procederá a realizar una petición GET al servicio del controlador, enviando como parámetros la latitud y longitud. Este servicio permite mostrar que la información del paseador que se encuentra en esa posición, esto es fundamental, ya que lo utilizaremos para la ubicación en tiempo real.

Figura 23

Método Get para la obtención del paseador a través de las coordenadas



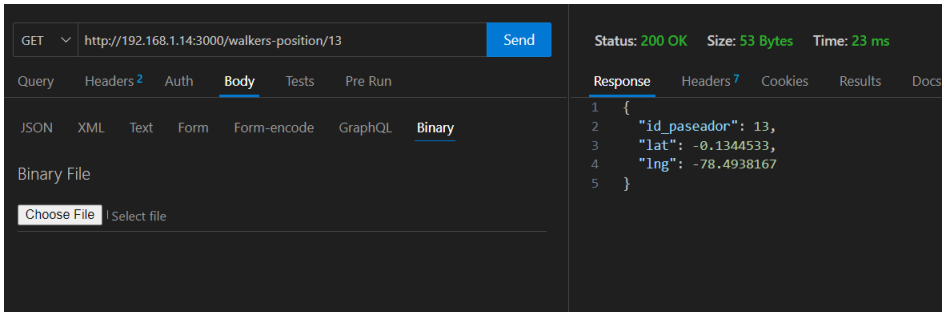
Como se muestra en la Figura 23**¡Error! No se encuentra el origen de la referencia.** se observa que la petición arrojo el código 200OK, que menciona que la petición se realizó de manera correcta. Además, envía la información desde la base de datos con los campos relevantes de la solicitud de paseo.

Para este caso de prueba se desea verificar la posición del paseador a través del ID. Para esto se realiza una petición de tipo GET a la especificada dentro del controlador. Recibe

como parámetro un campo, el cual es el id_paseador y permitirá ver en que latitud y longitud se encuentra dicho usuario.

Figura 24

Método Get para la obtención de la posición del paseador a través del ID

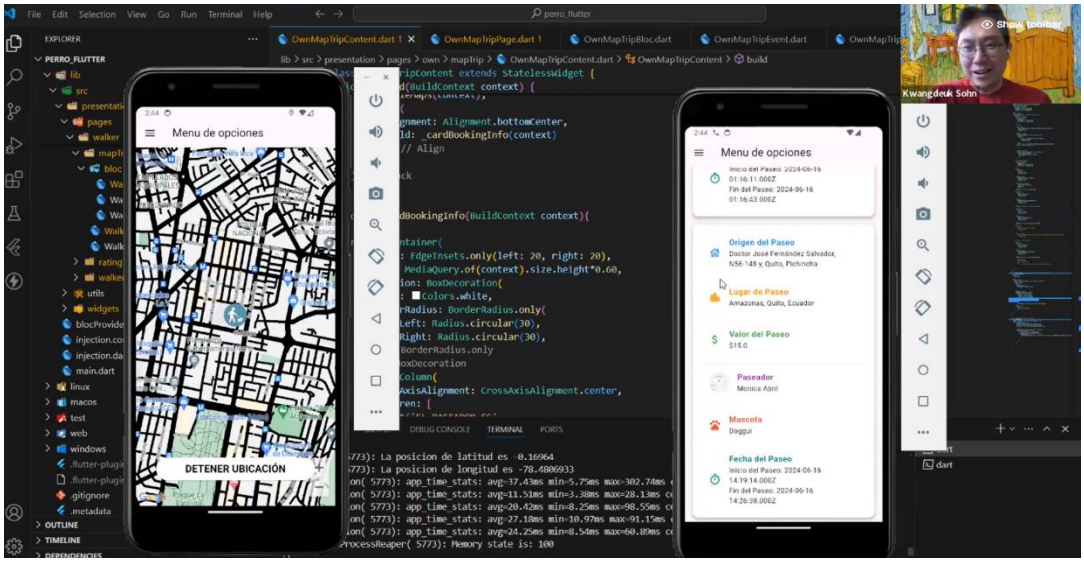


4.3.3.3. Feedback del Usuario

En el feedback del usuario se realizó la misma demostración a las personas mencionadas anteriormente. Como sugerencia, se propuso que dentro del historial de paseos realizados se incluya el nombre del dueño en el rol de paseador, así como la tarifa del precio del paseo en ambos roles.

Figura 25

Feedback por parte del usuario prototipo3



4.3.3.4. Refinamiento

Dado el feedback por parte del usuario que se mencionó anteriormente se procederá a añadir esos campos en la pantalla de historial de paseos.

4.4. Pantallas de los prototipos

4.4.1. Pantalla de Inicio de Sesión

Figura 26

Pantalla autenticación de usuarios

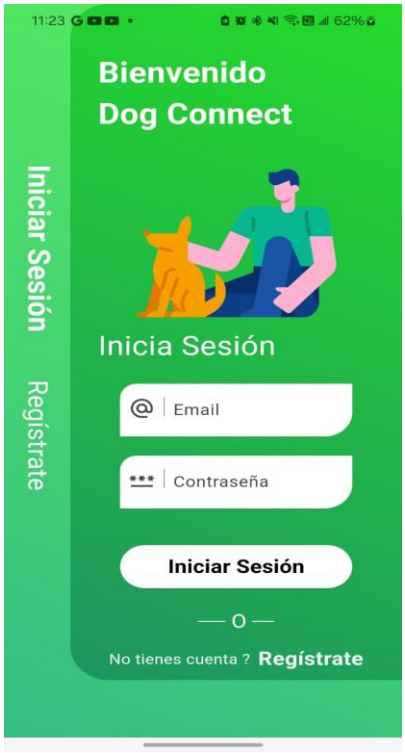
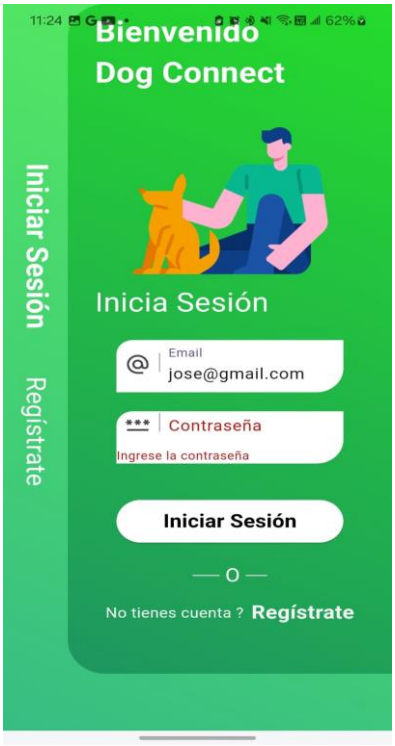


Figura 27

Pantalla autenticación de usuarios con validación de campos



En la Figura 26 se muestra la pantalla de login, se puede observar que para el inicio de sesión únicamente se necesitan los campos del correo y la contraseña que fueron ya registrados previamente, caso contrario de no estarlo, la aplicación mostrar un mensaje para que el usuario pueda registrarse. De la misma manera, la aplicación tendrá la validación de campos, ya que debe ingresar un correo válido, es decir que tenga tanto el signo de ¨@¨ como el ¨.com¨, también se agregó una validación al campo de la contraseña que mínimo debe tener un total de 9

caracteres para que pueda ingresar, esta validación se hizo en la pantalla de registro y debe coincidir al momento de iniciar sesión.

4.4.2. Pantalla Registro de Usuario

Figura 28

Pantalla registro de usuario

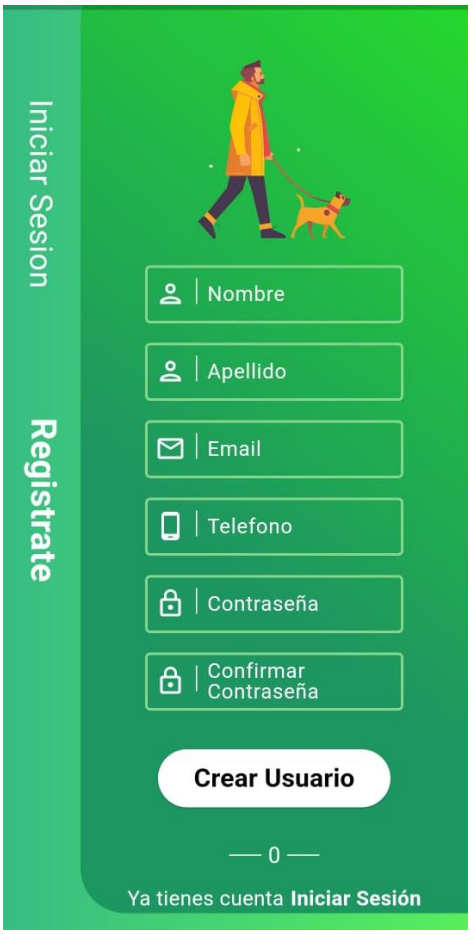
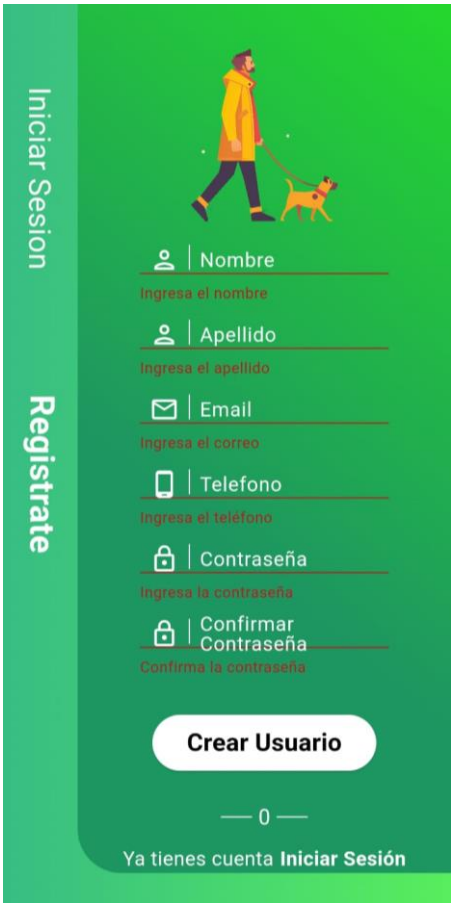


Figura 299

Pantalla registro de usuario con validación de campos



La pantalla de registro en la aplicación se muestra en la **¡Error! No se encuentra el origen de la referencia.** y **¡Error! No se encuentra el origen de la referencia.**, en donde se tomaron en cuenta los datos del usuario, lo cuales fueron, nombre, apellido, email, teléfono y contraseña. Estos datos son esenciales al momento de ingresar a la aplicación, ya que con los

datos que se han registrado anteriormente se podrán realizar peticiones ya sea cómo dueño de una mascota o como un paseador.

4.4.3. Pantalla Actualización de Perfil

Figura 30

Pantalla perfil usuario

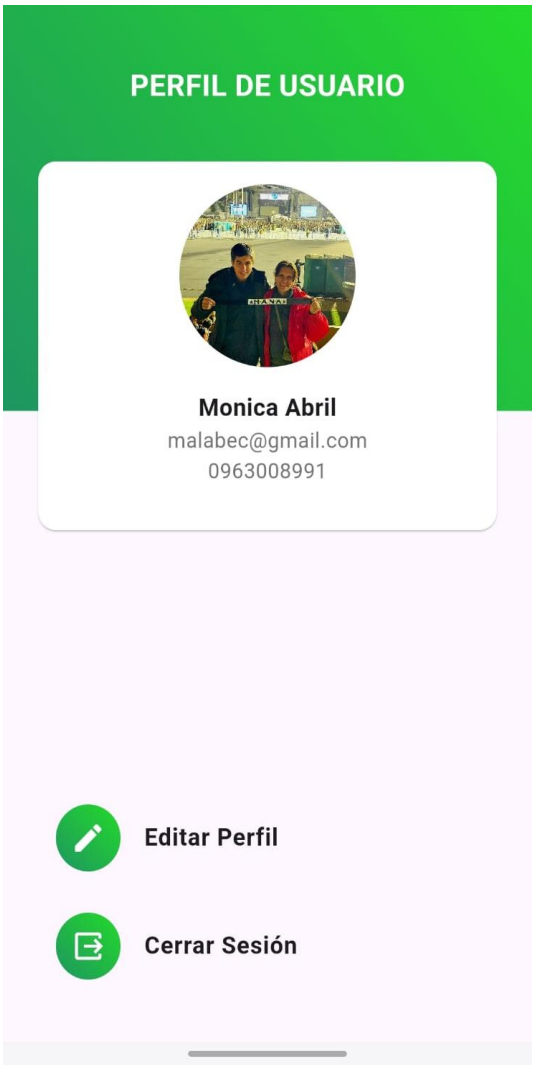
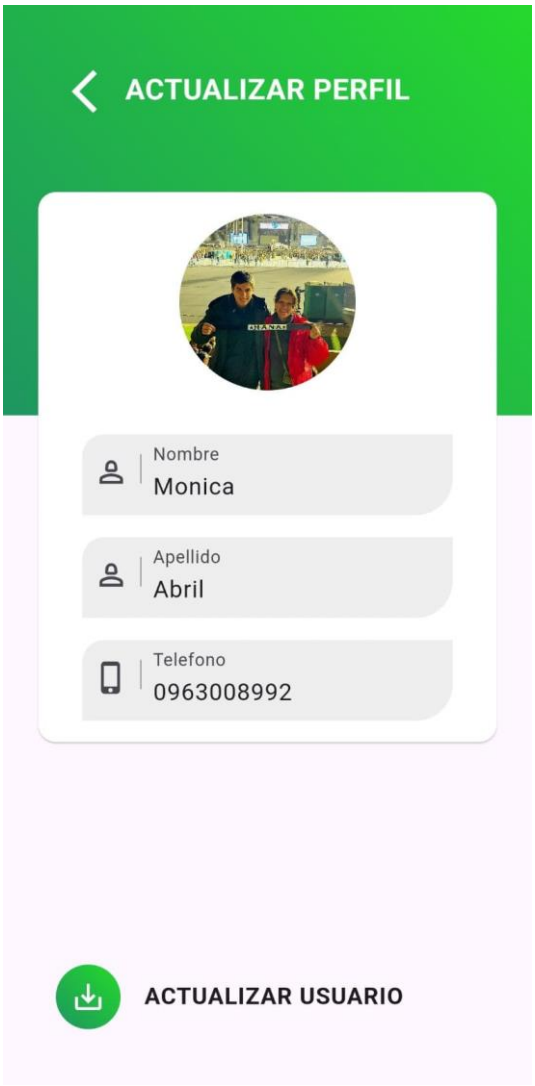


Figura 31

Pantalla actualizar perfil usuario



Una vez que se registró dentro de la aplicación y se permitió el ingreso a la misma, el usuario puede editar la información que tiene en su perfil, por otro lado, se agregó un campo para que el usuario pueda ingresar una fotografía a su perfil en caso de lo que desee. En la pantalla también se consideró poner un botón en donde el usuario pueda salir de la aplicación como se muestra en la figura, haciendo mucho más intuitiva la parte de edición y salida de la aplicación.

4.4.4. Pantalla Menú de Paseador y Dueño

Figura 33

Pantalla menú dueño

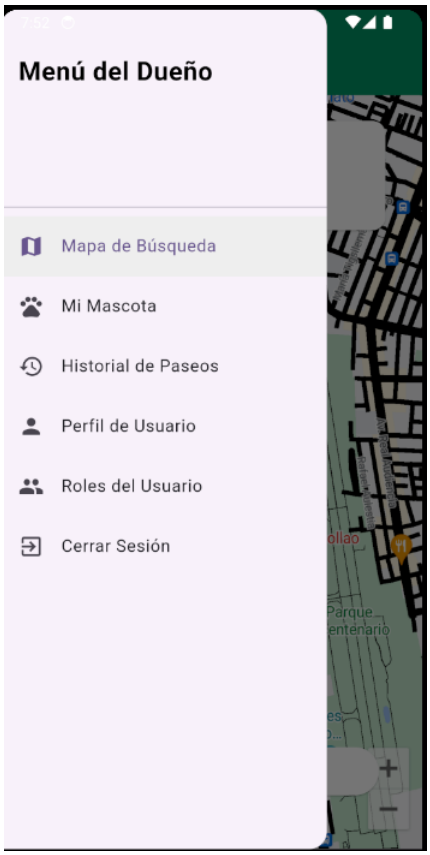
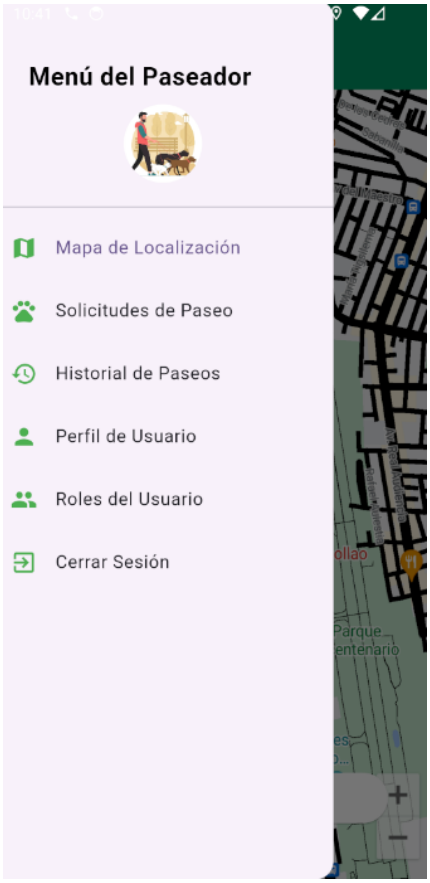


Figura 32

Pantalla menú paseador



En la Figura 33 se muestra las opciones que tiene el paseador, como son el mapa de localización en donde puede tener el control de su ubicación en tiempo real, las solicitudes del paseo, la cual muestra si un usuario hace una petición para un servicio de paseo a su mascota, también se tiene el historial de paseos que el paseador ha cumplido a lo largo de su estancia en

la aplicación, asimismo, el paseador podrá editar su perfil y la salida de la aplicación. Así mismo el dueño tiene las opciones de ver historial de paseos, el perfil del usuario, la mascota que tiene asociado, como lo muestra la Figura 32.

4.4.5. Pantalla Mapa de Localización

Figura 34

Mapa localización dueño

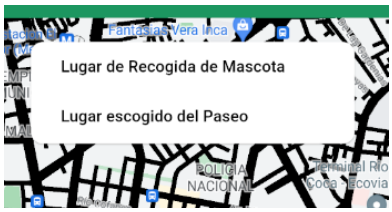


Figura 35

Mapa localización paseador

En la **¡Error! No se encuentra el origen de la referencia.** se visualiza la ubicación en tiempo real del paseador, permitiendo activar o desactivar la opción de geolocalización para recibir ofertas de paseo. En la **¡Error! No se encuentra el origen de la referencia.** se muestra la ubicación del dueño, quien puede agregar un punto de partida y un destino para solicitar un

paseo. Desde aquí se generará la ruta que el paseador seguirá hasta el lugar del paseo con la mascota del usuario. De esta manera, el dueño puede seguir en tiempo real el progreso del paseo, incluyendo el momento en que el paseador recoge a su mascota. Una vez seleccionados el punto de partida y el destino, se puede revisar un resumen del paseo presionando el botón revisar paseo.

4.4.6. Pantalla Solicitudes de Paseo

Figura 36

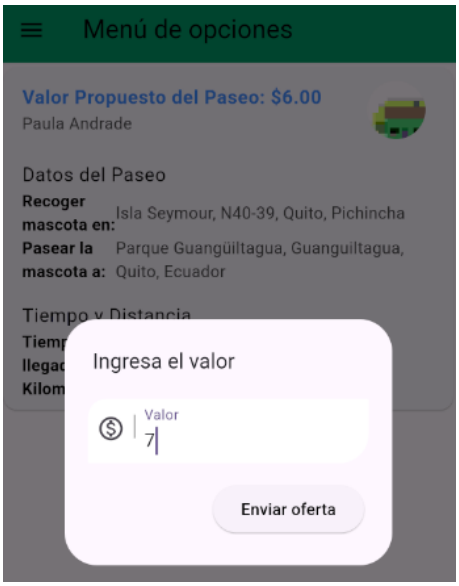
Solicitudes de paseo del dueño hacia el paseador

En la Figura 36 representa las solicitudes que los dueños de las mascotas hacen a paseadores, dentro de la pantalla se muestra información relevante como es, el valor que se

ofrece, la ubicación donde se recogerá a la mascota y el lugar en donde la mascota deberá ser paseada, así también como la distancia y el tiempo que se demorará en llegar a su destino.

Figura 37

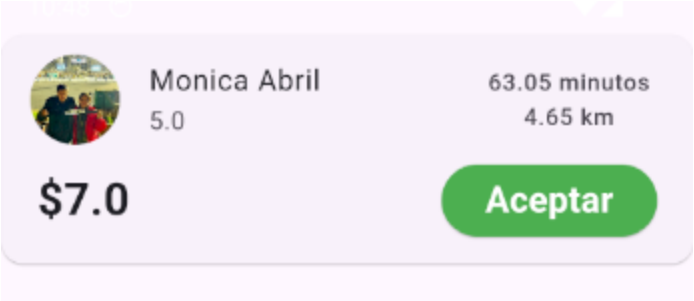
Pantalla contraoferta del paseador para el dueño



En la Figura 37 se puede apreciar una opción de contraoferta que podrá hacer el paseador en caso de que la oferta presentada no sea muy atractiva, con esta opción el paseador se asegurará que tenga un beneficio en relación con el tiempo y la distancia que empleara durante el paseo que el usuario ha propuesto.

Figura 38

Pantalla oferta enviada por el paseador al dueño



Una vez realizada la contraoferta se le presentará al dueño de la mascota una la contraoferta realizada por el paseador, él podrá aceptar o ignorar la oferta.

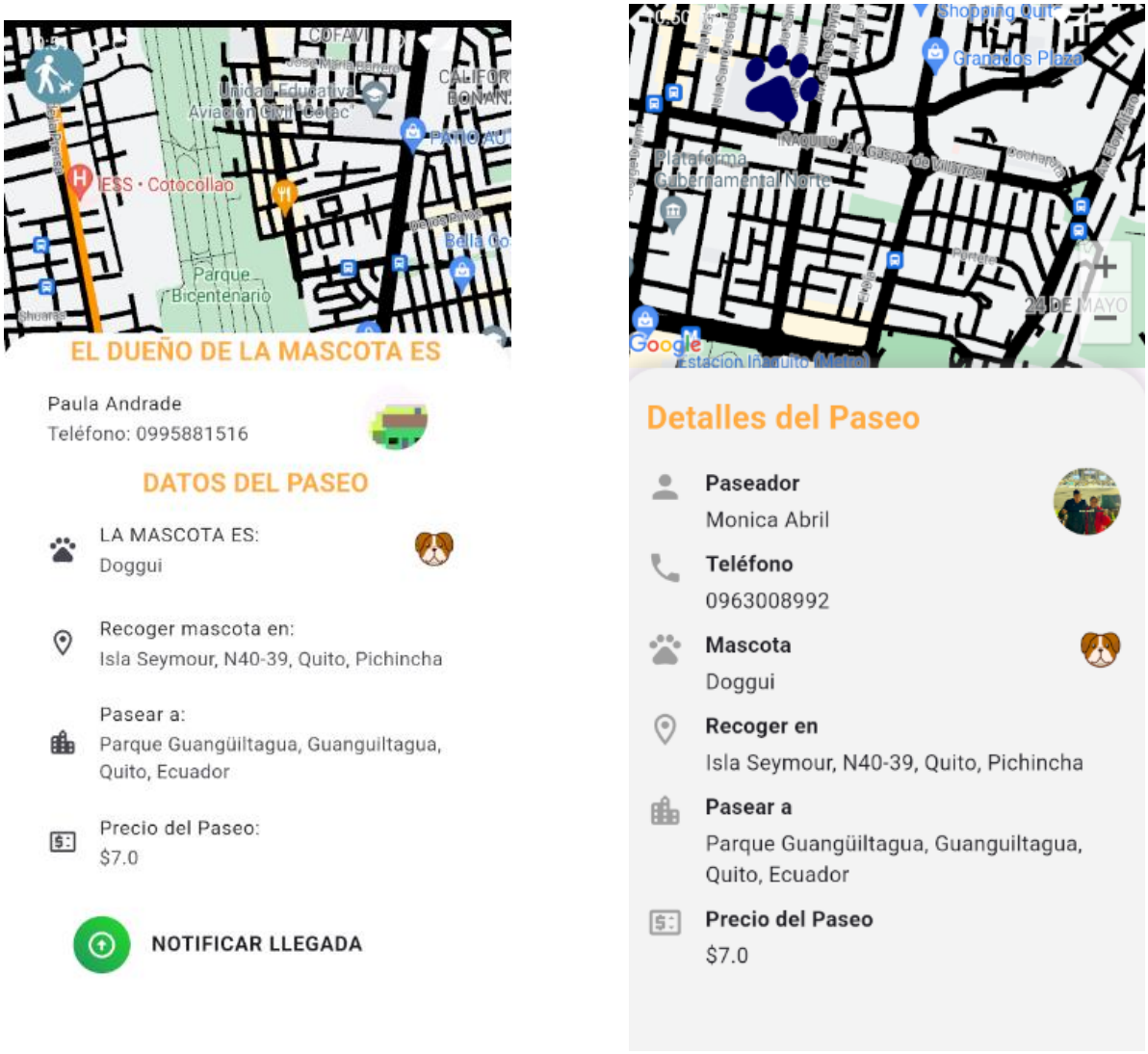
4.4.7. Pantalla Paseo

Figura 39

Pantalla paseo paseador

Figura 40

Pantalla paseo dueño



En la Figura 39 y Figura 40 se muestra los datos del paseo, estas pantallas se presentan siempre y cuando el dueño de la mascota ha aceptado la oferta del paseador. Se presentará esas 2 pantallas tanto al dueño de la mascota y como al paseador, El paseador tendrá la opción de modificar los estados del paseo, y podrá observar la ruta por donde se debe dirigir para poder retirar a la mascota.

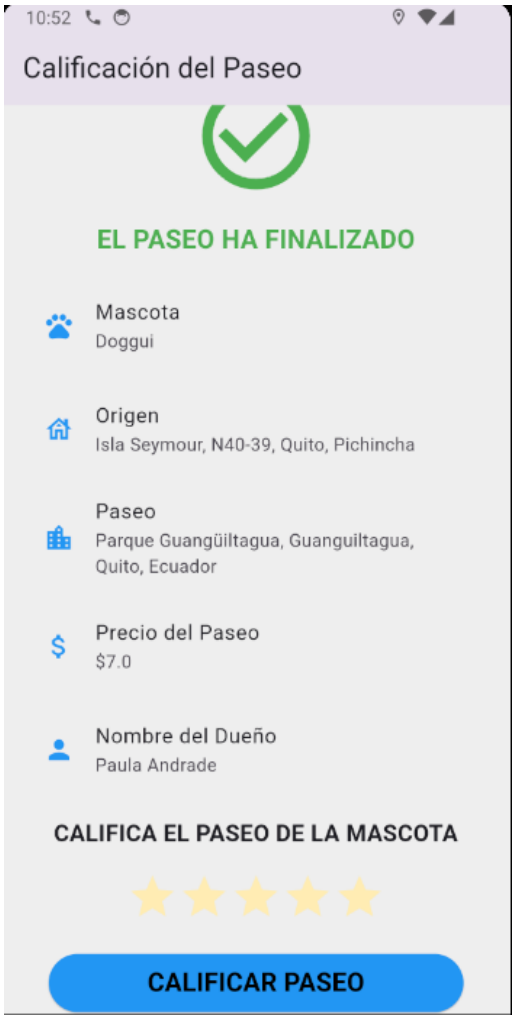
4.4.8. Pantalla Calificación

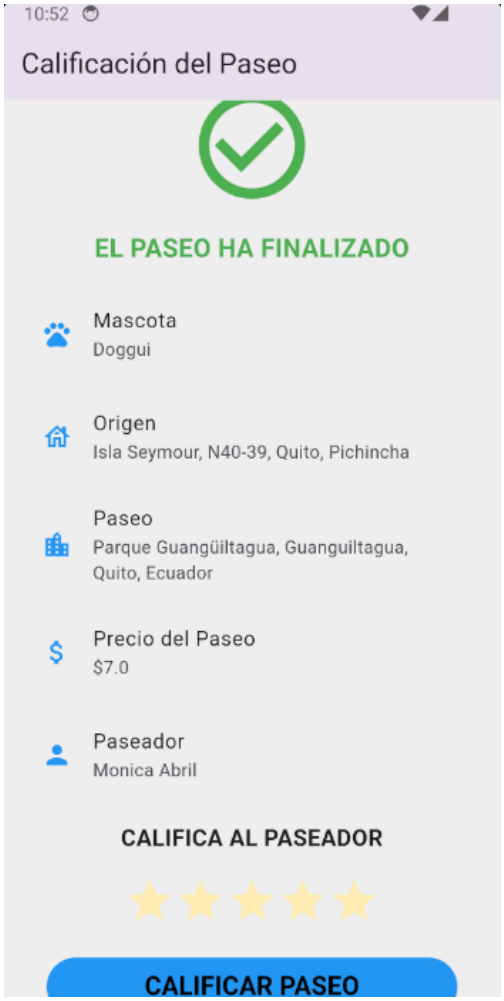
Figura 41

Pantalla calificación paseo paseador

Figura 42

Pantalla calificación paseo



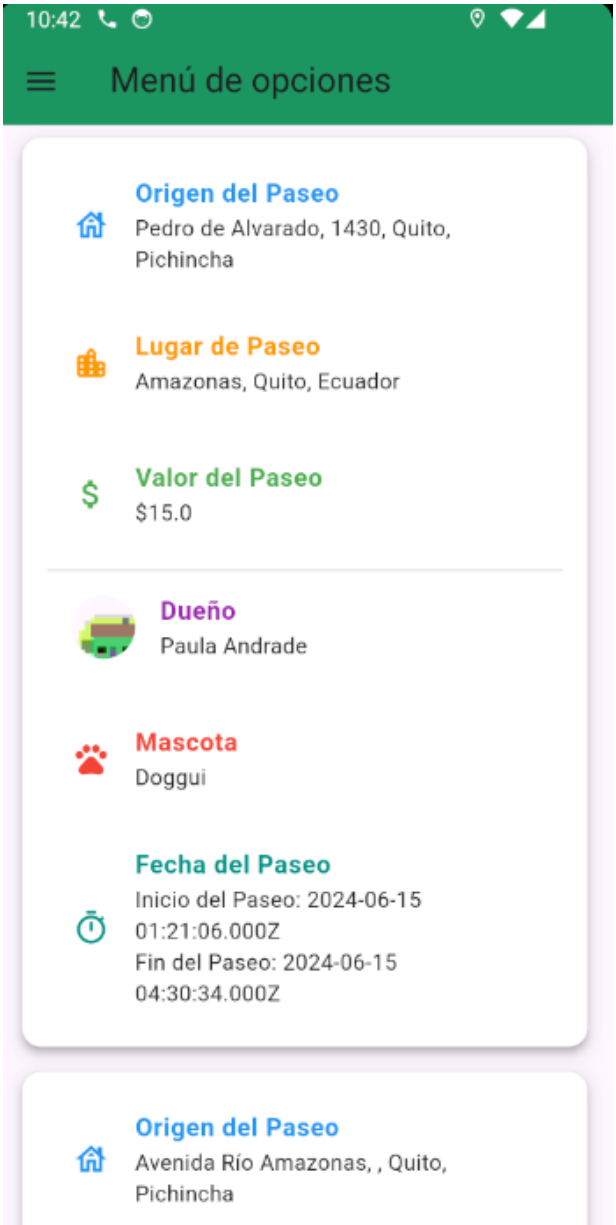


En la Figura 41 y Figura 42 se presenta la ventana de la calificación del paseo, esto se presentará tanto al dueño de la mascota como al paseador, de esta manera se asegurará tener una retroalimentación tanto al dueño como a la persona que hizo el servicio de paseo, con el fin de que próximos usuarios de la aplicación puedan tener en cuenta y la certeza de que sus futuros viajes estarán siendo realizadas con personas confiables.

4.4.9. Pantalla Historial de Paseos

Figura 43

Pantalla historial de paseos



En la Figura 43 se muestra el historial de todos los viajes que se han realizado tanto la mascota como con el paseador, de esta forma se podrá llevar un registro mucho más completo de todos los viajes y las se han realizado con el dicho usuario que la han realizado.

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES

5. Conclusiones y Recomendaciones.

5.1. Conclusiones

En conclusión, durante el levantamiento de requerimientos se observó que la mayoría de las personas que tienen mascotas no disponen de un espacio amplio, lo cual afecta negativamente la salud y bienestar de sus animales.

El diseño y desarrollo del prototipo de la aplicación se realizó con un enfoque centrado en el usuario, asegurando un prototipo que satisfacen las necesidades identificadas. La creación de este prototipo no solo fue un paso crucial para visualizar la solución propuesta, sino también para recibir retroalimentación temprana de los usuarios, lo cual permitió ajustes y mejoras para lograr tener el prototipo final.

La integración de la API de Google Maps para el seguimiento en tiempo real de los paseadores fue un componente esencial en el prototipo de nuestra aplicación. Esta funcionalidad se implementó para abordar las preocupaciones de los dueños de mascotas sobre la confianza en los paseadores, identificadas durante la recopilación de los requerimientos.

El uso de metodologías ágiles durante el desarrollo del prototipo permitió un ciclo de retroalimentación constante, asegurando la incorporación eficiente de nuevas funcionalidades y mejoras basadas en la retroalimentación de los usuarios.

Durante las pruebas de funcionalidad del prototipo, se verificó la operatividad efectiva de las características implementadas, asegurando que cumplan con los requerimientos y especificaciones establecidas.

5.2. Recomendaciones

Se recomienda que, para futuras iteraciones, se delimite el rango de aceptación de paseos, ya que actualmente no se cuenta con esta funcionalidad y pueden aparecer todas las solicitudes de paseos sin importar la distancia a la que se encuentre el paseador.

Se recomienda implementar un sistema de retroalimentación donde los usuarios puedan dejar comentarios y sugerencias de los paseadores, permitiendo así a los demás usuarios poder observar las reseñas de cada paseador.

Se recomienda la utilización de la librería TypeORM debido a que permite crear entidades y tablas en la base datos sin la necesidad de realizar una sentencia SQL como tal.

Se recomienda para futuras iteraciones realizar pruebas de rendimiento para garantizar que la aplicación funcione sin problemas en diferentes dispositivos y condiciones de red.

Se recomienda utilizar herramientas de control de versiones como GitHub, ya que permiten respaldar el proyecto en la nube y trabajar en diversas ramas para desarrollar nuevas funcionalidades de manera independiente, sin afectar la rama principal.

BIBLIOGRAFÍA

Amazon WS. (2020). *¿Cuál es la diferencia entre el front end y back end en el desarrollo de aplicaciones?* Obtenido de ¿Cuál es la diferencia entre el front end y back end en el desarrollo de aplicaciones?: https://aws.amazon.com/es/compare/the-difference-between-frontend-and-backend/

Aptiv. (08 de 03 de 2023). Obtenido de https://www.aptiv.com/es/tendencias/art%C3%ADculo/que-es-el-modelo-v-en-el-desarrollo-de-software

Dominguez, R. (20 de 12 de 2014). Obtenido de https://www.slideshare.net/slideshow/servidor-mysql/42903726

Formadores IT. (21 de 08 de 2023). Obtenido de https://formadoresit.es/sql-vs-nosql-en-que-se-diferencian/

IBM. (2021). *IBM.* Obtenido de IBM: https://www.ibm.com/mx-es/topics/api

Ionos. (s.f.). Obtenido de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/

Martins, J. (15 de 02 de 2024). Obtenido de https://asana.com/es/resources/what-is-scrum

Navarro, A., Fernández, J., & Morales, J. (20 de 09 de 2013). *redalyc.* Obtenido de Revisión de metodologías ágiles para el desarrollo de software: https://www.redalyc.org/pdf/4962/496250736004.pdf

Pérez, S. D. (18 de 10 de 2021). Obtenido de https://intelequia.com/es/blog/post/qu%C3%A9-es-microsoft-sql-server-y-para-qu%C3%A9-sirve

Platzi. (2015). Obtenido de https://platzi.com/blog/que-es-postgresql/

Red Hat. (31 de 07 de 2023). Obtenido de https://www.redhat.com/es/topics/api/what-is-a-rest-api

Senderos, L. L. (20 de 03 de 2023). *seidor.* Obtenido de https://www.seidor.com/blog/ionic-que-es-es-el-futuro-de-las-apps

Sinnaps. (s.f.). Obtenido de https://www.sinnaps.com/blog-gestion-proyectos/metodologia-xp

Tokio School. (06 de 06 de 2022). Obtenido de https://www.tokioschool.com/noticias/que-es-django/

UNIR. (17 de 06 de 2021). Obtenido de NoSQL vs. SQL: características, diferencias y contextos de uso de estas tecnologías de SGBD: https://www.unir.net/ingenieria/revista/nosql-vs-sql/