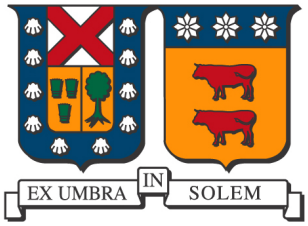
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO – CHILE



DISEÑO Y DESARROLLO DE UNA APLICACIÓN MULTIPLATAFORMA QUE IMPLEMENTA UN SISTEMA DE ALERTAS, GESTIÓN DE VOLUNTARIADO E INFORMATIVO PARA LA COMUNIDAD ANIMALISTA

LORENS ANDRE PÁEZ PASTENES
JORGE ERNESTO FERNÁNDEZ DE LA FUENTE

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL TELEMÁTICO

PROFESORES GUÍAS

BERIOSKA CONTRERAS VARGAS
NICOLÁS JARA CARVALLO

DICIEMBRE 2023

Agradecimientos Lorens

A mi mamá, la persona más importante de mi vida, por darme la oportunidad de poder hacer está travesía, darme la confianza de que siempre haga lo que me de felicidad y su incondicional apoyo.

A mis hermanos Enzo, Diego y Vladimir, por su amor, cariño, enseñanzas y siempre cuidarme como si fuera un hijo para ellos.

A Bárbara por ser mi apoyo incondicional en esta última etapa.

A mis perritas por ser mi soporte emocional durante tantos años.

A mi equipo de memoria y profesores guías, por su apoyo, motivación y cariño, sin duda han sido un gran equipo y referentes para afrontar este periodo.

A mis grandiosos amigos y compañeros de Telemática, por enseñarme lo que significa la verdadera amistad, sin ustedes este viaje no habría sido lo mismo.

A cada una de las personas en la universidad que creyó en mi, me dió una oportunidad, me escuchó o me entregó un minuto de su tiempo.

A mi amigo Pawel, por estar presente en el año más difícil que me ha tocado.

A todos, gracias de corazón.

Agradecimientos Jorge

A mi familia.

A mi abuela Norma Scheggia por su apoyo incondicional tanto moral como económico.

A mi perro QEDP por el apoyo emocional.

A mis compañeros por acompañarme a lo largo de la carrera y compartir los momentos difíciles de ésta.

A mi sensei Sergio por enseñarme el camino y razonamiento del karate para enfrentar los desafíos que tuve que superar.

A mi equipo de memoria peludites, por su gran trabajo y disposición durante este trabajo.

Resumen

Hoy en día las personas y organizaciones que se dedican al cuidado de mascotas abandonadas y maltratadas actúan de forma individual, donde su principal herramienta de comunicación y apoyo mutuo son las redes sociales. Sin embargo, estas plataformas no son del todo adecuadas, ya que no satisfacen las necesidades diarias de manera eficaz y eficiente, además de no crear una comunidad robusta. Para dar solución a esta problemática, la fundación Bienestar Animal, principal interesado en este proyecto plantea la siguiente pregunta desafío.

¿Cómo podemos generar relaciones sinérgicas y dinámicas entre actores del mundo animalista para así formar una red de apoyo que visibilice, sostenga y resuelva el problema del abandono y maltrato de mascotas?

En la presente tesis se diseña y construye una solución al desafío planteado por la fundación Bienestar Animal, se presenta un análisis de contexto y problemática, proceso que entregará como propuesta de solución, una plataforma digital que funcione como nexo efectivo entre todos los actores relevantes en el rescate y cuidado animal.

Con la solución desarrollada como un prototipo funcional de una aplicación móvil, fue posible establecer relaciones efectivas entre voluntarios rescatistas y fundaciones animalistas, para así tomar acciones ante alertas de rescate anunciadas por todo tipo de usuarios. Así como también fue posible generar una plataforma que centralice la información y comunicación necesaria para establecer una comunidad enfocada en el rescate animal.

Palabras Clave: *Ingeniería de software, computación en la nube, desarrollo de aplicaciones multiplataforma, análisis y diseño de software, pensamiento de diseño, red social y colaborativa.*

Abstract

Nowadays, individuals and organizations dedicated to the well-being of abandoned and mistreated pets operate independently, relying on social networks as their primary communication and mutual support tool. However, these platforms prove inadequate, failing to meet daily needs efficiently and effectively, and lacking in the establishment of a robust community. In response to this challenge, the Bienestar Animal foundation, the main stakeholder in this project, poses the following question:

How can we cultivate synergistic and dynamic relationships among stakeholders in the animal welfare community to establish a support network that not only raises awareness but also sustains and addresses the issue of pet abandonment and mistreatment?

This thesis conducts a comprehensive analysis and proposes a solution to the challenge presented by the Bienestar Animal Foundation. It includes an examination of the context and the issues at hand, culminating in a proposed solution—a digital platform serving as an effective nexus among all relevant stakeholders in animal rescue and care.

With the solution developed as a functional prototype of a mobile application, it was possible to establish effective relationships between rescue volunteers and animal foundations, in order to take action on rescue alerts announced by all types of users. It was also possible to generate a platform that centralizes the information and communication necessary to establish a community focused on animal rescue.

Keywords: *Software engineering, cloud computing, cross-platform application development, software analysis and design, design thinking, collaborative and social network.*

Glosario

Concepto	Definición
Base de datos	Recopilación organizada de información o datos estructurados, que normalmente se almacena de forma digital en un sistema informático.
Aplicación web	Programa escrito en un lenguaje interpretable por los navegadores web, a los que se les delega en gran medida el proceso de ejecución. Además suelen contar con conexiones a micro-servicios en servidores externos.
Frontend	Concepto dentro de la programación de aplicaciones web que separa el sistema y sus tareas de desarrollo al apartado visible por el usuario (también llamado client-side)
Backend	Análogo a Frontend, para las tareas ”detrás de escena”, es decir, el procesamiento de datos y solicitudes que no le conciernen al usuario ni al navegador. (también llamado server-side)
Framework	Literal del inglés Marco de trabajo, es un conjunto de herramientas de software, como bibliotecas, que permiten agilizar el desarrollo de aplicaciones web.
Mockup	Traducido del inglés como bosquejo, es un fotomontaje a través del cual los diseñadores gráficos pueden presentar sus propuestas a los clientes.
Stakeholder	Es una persona del público de interés para una empresa que permite su completo funcionamiento
Hash/Hashing	Es una función o método criptográfico que a través de un algoritmo matemático transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija.
HTTP	Es el protocolo de transmisión de información de la World Wide Web.
API	Traducido del inglés como interfaz de programación de aplicaciones, provee mecanismos que permiten a dos componentes de software comunicarse entre sí.
Javascript	Es un lenguaje de programación interpretado utilizado principalmente para desarrollar aplicativos web.
JSON	acrónimo de JavaScript Object Notation, notación de objeto de JavaScript, es un formato de texto sencillo para el intercambio de datos.
Login	proceso que controla el acceso individual a un sistema informático mediante la identificación del usuario utilizando credenciales provistas por el usuario.

Tabla 1: Glosario

Concepto	Definición
Typescript	Es un lenguaje de programación tipificado, libre y de código abierto, desarrollado sobre Javascript.
Endpoint	Traducido del inglés como punto de conexión, es una dirección remota para acceder a rutas y consumir datos almacenados en un dispositivo.
Token	se utiliza para acceder a un recurso restringido (normalmente una aplicación o una red corporativa). Puede considerarse como una llave electrónica que permite a un usuario autenticarse, demostrando su identidad, mediante el almacenamiento de una determinada información personal.
APK	Archivo ejecutable que contiene todos los datos que se necesitan para instalar y hacer funcionar una aplicación Android
CRUD	Es el acrónimo de Crear, Leer, Actualizar y Borrar, que se usa para referirse a las funciones básicas en bases de datos.
Testing	Es el proceso de realizar pruebas funcionales, concepto u otras a un producto digital.

Tabla 2: Glosario

Para simplificar el lenguaje usado, cuando se hable de Organizaciones, se puede estar refiriendo tanto a Organizaciones como Fundaciones legalmente constituidas dedicadas al rescate, rehabilitación y bienestar animal.

Índice

1. Introducción

1.1. Contexto

La convivencia entre humanos y mascotas ha experimentado una notable evolución, donde estos animales, principalmente perros y gatos, son considerados cada vez más como miembros de la familia. A pesar de los avances en la disminución del abandono y maltrato, persisten casos diarios que plantean desafíos continuos en diversas regiones del país. La encuesta reciente de CADEM revela que el 86 % de los entrevistados tiene al menos una mascota, marcando un aumento del 13 % desde 2019[1]. A pesar del creciente afecto hacia las mascotas, la desinformación y prácticas obsoletas persisten, generando desafíos como camadas no deseadas y enfermedades con tratamientos onerosos, factores que contribuyen al lamentable abandono de estos animales.

Para comprender las perspectivas y desafíos que enfrentan los cuidadores de mascotas comunes en nuestro país, se ha llevado a cabo un breve estudio9.2. Este contexto resalta la necesidad de abordar problemas arraigados en la falta de información y en prácticas tradicionales que afectan directamente la calidad de vida de las mascotas y contribuyen al problema del abandono.

En respuesta a este escenario, diversas fundaciones y organizaciones en distintas regiones del país trabajan incansablemente con el objetivo común de erradicar el abandono de animales domésticos. Estas entidades rescatan, rehabilitan y buscan nuevos hogares responsables para perros y gatos rescatados en sectores vulnerables. Sin embargo, la efectividad de sus esfuerzos se ve limitada por la falta de un nexo eficaz que agilice la comunicación con la comunidad interesada, compuesta por individuos y empresas que pueden contribuir con recursos monetarios, materiales o alimenticios.

La presente investigación se centra en abordar esta problemática y busca proporcionar una solución innovadora mediante el desarrollo de una plataforma digital. Esta plataforma tiene como objetivo facilitar la comunicación, coordinación y apoyo entre todas las partes involucradas en el rescate y cuidado animal. La Fundación Bienestar Animal, compuesta por profesionales comprometidos, se erige como la entidad impulsora de este proyecto, utilizando la innovación digital como medio para agilizar el rescate y asistencia a animales maltratados, difundir información sobre tenencia responsable y coordinar las acciones de los diversos actores en pro del bienestar animal.

Esta investigación aborda no solo la problemática del abandono de mascotas, sino también la necesidad crítica de establecer un nexo eficaz que fortalezca la colaboración entre las fundaciones, la comunidad interesada y las empresas dispuestas a contribuir a esta causa. A través de la implementación de una plataforma digital, se busca mejorar la eficiencia de la comunicación y coordinación en el ámbito del rescate y cuidado animal.

1.2. Problema

1.2.1. Situación Actual

Hoy en día las entidades animalistas cuentan con canales de comunicación funcionales con empresas privadas, servicios de animales domésticos, el Ministerio del Medio Ambiente y la posibilidad de postulaciones a fondos concursables fig.1. Sin embargo, también hay medios de contacto ineficientes con personas interesadas en el bienestar animal, las cuales son potenciales proveedores de ayuda, donaciones y adopciones; acciones importantes para llevar a cabo la lucha contra el abandono. También se debe considerar que no mantienen un conducto directo con la información de servicios de animales domésticos, información sobre las leyes vigentes ni un medio para tomar acción en caso de presenciar abandonos y maltratos hacia otros animales.

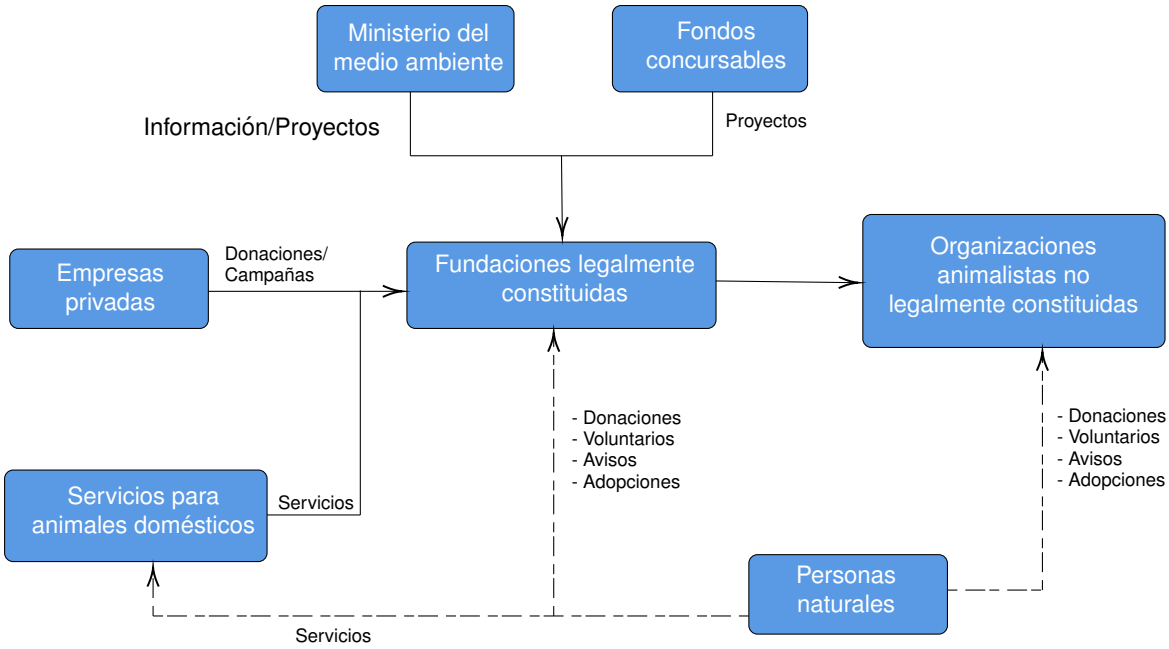


Figura 1: Situación actual con información descentralizada

1.2.2. Problemáticas por resolver

Al analizar el contexto en el que se presenta el desafío, se evidencia que la problemática recae en la ineficiente comunicación que tienen las personas naturales con el resto de los actores que participan en el cuidado y bienestar animal, la información y comunicación descentralizada entre fundaciones, organizaciones y personas naturales impide que el abandono de animales domésticos disminuya.

- **Canales de comunicación descentralizados:**

  La información sobre tenencia responsable de mascotas no cuenta con la difusión necesaria ni es de fácil comprensión, esto influye en que las personas no tengan un conocimiento acabado al respecto debido a que no poseen el hábito de buscarlo de manera independiente.

  Además, las diferentes fundaciones y organizaciones tienen canales independientes de comunicación y esto dificulta que las personas naturales puedan estar permanentemente informadas acerca de las intervenciones que hagan dentro de su entorno.

- **Escasa concientización y educación orientada a los dueños de mascotas:**

  A pesar de existir una ley de tenencia responsable de mascotas, aún existe una alta tasa de maltrato y abandono, las fundaciones, organizaciones y voluntarios animalistas terminan haciéndose cargo del cuidado y esterilización de los animales debido a que los dueños no se responsabilizan de sus necesidades.

  Al entrevistar a una de las directoras de la organización UPPACK de la Región de Valparaíso 30, informa de la falta de cuidado y respeto que tienen algunos dueños con sus mascotas, según menciona, a veces se realizan operativos de esterilización, pero luego los dueños no cooperan con la entrega del animal.

- **Dificultad para identificar todas las organizaciones y fundaciones animalistas existentes en distintas localidades:**

  No existe un directorio general para conocer todas las fundaciones y organizaciones que existen en una ciudad que sea de fácil acceso, por lo que para encontrarlas, las personas deben buscar por su cuenta en plataformas donde posiblemente no se registre la totalidad de estas.

- **Gran cantidad de servicios orientados a mascotas que están interesados en aumentar sus ventas y no son parte de un directorio central:**

  Es difícil contar con un listado de todos los servicios disponibles relativos al cuidado de la mascota y con su reseña, esto genera que las personas naturales dueñas de estos animales deban acudir a servicios disponibles en su cercanía, que no siempre se adaptan a sus necesidades.

1.3. Acercamiento a la Solución

Para atender la problemática, se ha propuesto el desarrollo de una plataforma digital que funcione como un nexo efectivo entre las fundaciones, comunidades animalistas, personas naturales y otros actores relevantes que serán categorizados dependiendo del tipo de usuario y las necesidades que este tenga dentro de la solución.

La potencial aplicación busca contar con distintas secciones que ayude a solucionar los problemas planteados, desde la entrega de información verídica y de fácil comprensión sobre la tenencia responsable de mascotas; que ayude a la mejora de la comunicación entre usuarios y fundaciones, denuncias de maltrato, implementación de protocolos de acción frente a situaciones de emergencia y el ofrecimiento o búsqueda de adopciones.

Es importante destacar la carencia de un directorio de información centralizada para facilitar la toma de decisiones y entregar múltiples opciones a los usuarios tanto en situaciones cruciales como comerciales, además de lograr eficacia y oportuna resolución de las denuncias realizadas en la plataforma.

1.3.1. Definición de soluciones parciales

Una vez definida la problemática y el acercamiento a la idea de solución, se plantean los siguientes ejes que conformarán la propuesta, los cuales permitirán resolver las diferentes problemáticas expuestas al comienzo de esta memoria:

1. Sistema de aviso y alerta para casos de abandono, maltrato, entre otros, que permita el rápido actuar de organizaciones y voluntarios.
2. Canal formal para búsqueda y ofrecimiento de adopciones.
3. Entrega y difusión de información verídica y de fácil comprensión sobre tenencia responsable de mascotas domésticas, protocolos de acción ante distintas situaciones y Ley vigente.
4. Un directorio de datos centralizados de las fundaciones y servicios asociados a la comunidad animalista.

1.4. Objetivos

1.4.1. Objetivo General

Construir el prototipo funcional de una aplicación que permita resolver situaciones de emergencia de animales y que, a la vez, centralice la información existente sobre tenencia responsable, protocolos de acción, servicios de cuidado de mascotas y de las fundaciones u organizaciones que hay en cada ciudad de los usuarios, para permitir un flujo de información fructuoso entre estas y la comunidad.

1.4.2. Objetivos Específicos

1. Definir problemática del desafío planteado mediante análisis de contexto y usuarios, junto con la idea de solución. **(OE1)**

   Investigación sobre las condiciones en las cuales se encuentra el contexto y las necesidades existentes de cada uno de los usuarios involucrados en el desafío, definiendo la problemática, determinando los conflictos por resolver y la idea de solución para esta.

2. Diseñar el software y arquitectura de la aplicación. **(OE2)**

   Con base en los elementos que debe presentar la plataforma, se diseña su algoritmo y valida la correcta usabilidad de tal modo que genere valor para los distintos usuarios.

3. Definir diseño de la plataforma mediante la implementación de metodología UX/UI. **(OE3)**

   Con el uso de técnicas y métodos de UX/UI se dará forma, orden, despliegue y diseño de los datos a presentar en la plataforma. Esto será guiado principalmente por el perfil de diseño de productos.

4. Construcción del servidor Backend **(OE4)**

   Definidos los lineamientos para la construcción del software, se desarrolla el servidor Backend, que dará las bases para consumir la información necesaria de nuestra base de datos.

5. Desarrollar el prototipo de la plataforma. **(OE5)**

   Con el diseño de los mockups de la aplicación, se iniciará el desarrollo del Frontend que consuma los datos del Backend y así finalizar con un producto mínimo viable (MVP).

1.5. Alcance del proyecto

La escala de Madurez Tecnológica o Technology Readiness Level (TRL) es una medida para describir el estado de desarrollo o madurez de una tecnología fig.2 .

Se considera que el Programa de Memorias Multidisciplinarias de la UTFSM busca llegar a un prototipo mínimo viable en etapa TRL 6, de esta manera se determina el alcance del producto a desarrollar.

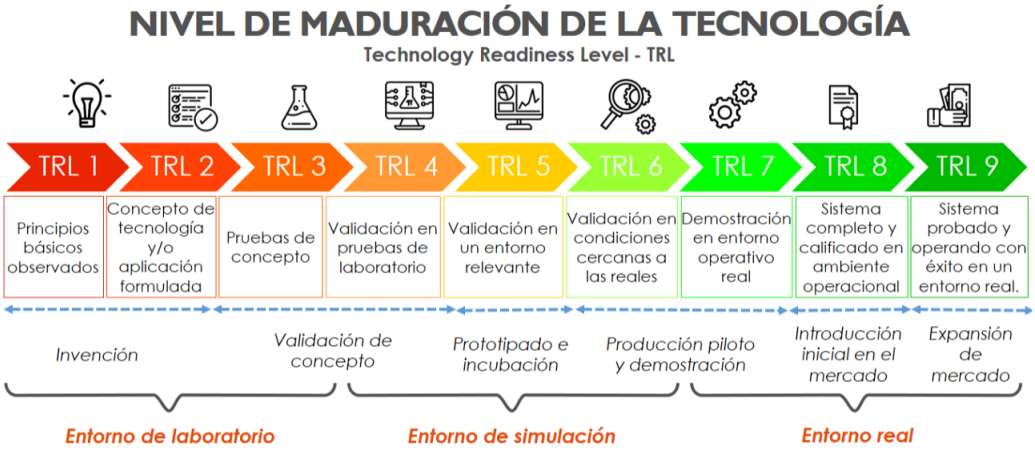


Figura 2: Escala TRL

1.5.1. Elementos presentes en el prototipo

De acuerdo a los intereses de la fundación mandante y la percepción de los usuarios de la solución, los elementos más importantes para que el prototipo cumpla con su objetivo serán la sección Informativa, el Botón de emergencias para el reporte de alerta y un Mapa interactivo para la visualización de estas alertas fig.3, dejando las restantes ideas de propuesta de solución como trabajo a futuro.

El enfoque de este prototipo será proporcionar un espacio que permita coordinar las acciones de los distintos usuarios al momento de existir casos de emergencia para animales en situaciones vulnerables y, junto con esto, entregar toda la información posible sobre el bienestar que se le puede brindar a las mascotas además de protocolos de acción para situaciones específicas y la muestra de las leyes vigentes.

Cabe destacar que los únicos perfiles existentes en este prototipo serán los informativos, es decir, de organizaciones y fundaciones, con el objetivo de que las personas puedan encontrarlos por este medio, obtener sus datos de contacto y localización.

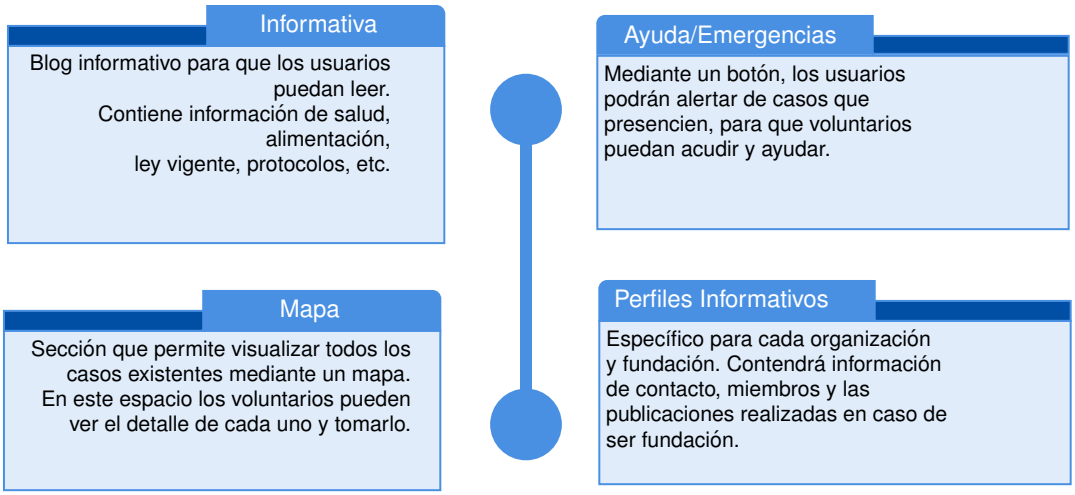


Figura 3: Secciones abarcadas dentro del Programa

En esta sección introductoria se ha delineado el contexto, problemática que motiva el presente trabajo junto con un breve acercamiento a la solución y sus objetivos.

A lo largo de los capítulos subsiguientes se presentará el estudio del estado del arte de otras soluciones y tecnologías que cubren rubros similares, se explorará en detalle el desarrollo de la plataforma, desde el análisis de requisitos hasta la definición de la solución y la elección de tecnologías. Posteriormente, se examinará la implementación del sistema, destacando tanto los componentes del Backend como del Frontend, así como también la sección de Verificación y Validación del Prototipo será crucial para evaluar la efectividad y robustez del producto. Finalmente, la conclusión y los trabajos futuros cerrarán este escrito, resaltando las contribuciones realizadas y proponiendo áreas de mejora para el producto desarrollado.

2. Marco Teórico

2.1. Diagnóstico

En la actualidad, existe un sinnúmero de agrupaciones animalistas, que llevan a cabo actividades de rescate, rehabilitación y operativos médicos entre otros, junto con esto suelen experimentar constantes desafíos y problemas para poder afrontar y llevar a cabo sus labores. Para poder difundir sus acciones utilizan diferentes redes sociales y se comunican de boca en boca, pero es evidente que estos medios no son soluciones eficientes.

Dado lo anterior, se reconoce la importancia de crear un canal formal para permitir que todos los actores del mundo del rescate animal puedan tener comunicación constante. La ausencia de éste mecanismo hace que surjan múltiples problemas asociados a la coordinación de estos entes y también que el alcance de información valiosa que estas organizaciones otorgan sea ineficaz. También se distingue que no existe un directorio centralizado de organizaciones y fundaciones constituidas donde presenten su información de contacto, lo que dificulta que un usuario pueda encontrar los datos de su interés, sin tener que realizar una búsqueda dificultosa.

Plataformas ya existentes no han logrado generar la eficiencia necesaria en el marco nacional para agilizar procesos de ayuda y rescate frente a casos de abandono y maltrato, ni han logrado una llegada eficaz de la información a los tenedores de mascotas para mejorar costumbres que pueden llegar a ser dañinas para las mascotas. Muchas de estas aplicaciones han generado impacto durante tiempos acotados, estando actualmente fuera de servicio o con uso mínimo, de manera que no solucionan la problemática. Es así como destaca la importancia del proyecto que nace con este desafío.

2.2. Estado del Arte

Dentro de las acciones que adoptan los actores que trabajan en busca del bienestar animal, podemos destacar operativos de vacunación, esterilización, rutas de alimentación para animales en situación de calle, jornadas de adopción, publicaciones informativas en distintas redes sociales como Facebook e Instagram; sin embargo, estas medidas no son una verdadera solución sino una forma de disminuir el problema, debido a que solo se pretende mejorar la calidad de vida de un número limitado de animales a cargo de grupos pequeños de personas.

Desde el año 2017 en Chile está en vigencia la Ley de Tenencia Responsable de Mascotas y Animales de Compañía[2], la cual determina las obligaciones y derechos de los responsables, protege la salud y el bienestar animal, además de la salud pública aplicando medidas para el control de la población de animales y regular la responsabilidad por los daños a las personas y a la propiedad. Es importante destacar que el reglamento de la ley señala que es obligatorio que todas las mascotas deben ser registradas utilizando un chip subcutáneo para cumplir de manera eficiente lo establecido por la ley.

En la actualidad en nuestro país se ha utilizado el recurso de las plataformas digitales desarrollando diversas soluciones sobre la base de aplicaciones móviles, aunque ninguna alcanzó a ser lo suficientemente masiva para ser considerada efectiva, tuvo la recepción esperada por parte de los usuarios, logró la sustentabilidad suficiente para continuar en funcionamiento pleno o también otros desarrollos que no han puesto su foco en la comunidad animalista, por lo que no resuelven las carencias de esta.

A continuación se detallan soluciones que buscan resolver alguna de las problemáticas que se han visibilizado dentro y fuera de nuestro país, o que bien implementan herramientas de uso similar a las propuestas:

- **Club Wuf:**[3] Consiste en una aplicación móvil creada por WUF, asociación sin fines de lucro situada en Perú, cuyo objetivo es ser un programa de ayuda a albergues a cambio de beneficios para los usuarios, de tal forma que al afiliarse, junto con su mascota pueden optar a descuentos en diversos servicios como veterinarias, hospedajes entre otros.
- **SoSafe:**[4] Aplicación móvil enfocada en conectar distintos segmentos usuarios para generar comunidades seguras y comunicadas. Esta permite la publicación de todos tipo de alertas y situaciones en un mapa interactivo que puede ser visualizado por todo tipo de usuarios, pero no se enfoca en ningún segmento específico.

- **Voy Contigo:**[5] Es una plataforma digital que implementa un sistema que brinda apoyo y seguridad a las mujeres, generando una comunidad femenina. Esta funciona con la redacción de reportes geolocalizados, los cuales quedan disponibles en un mapa interactivo con marcadores que indican ocurrencias.
- **Life360:**[6] Aplicación móvil enfocada en la agrupación de contactos para dar seguimiento geolocalizado a cada uno de los participantes. En esta se pueden especificar zonas geográficas clave para identificar hogares de los integrantes del grupo, además se pueden visualizar las rutas hechas por las personas e interactuar con mensajes.
- **PetPins:**[7] Aplicación móvil establecida en Europa que provee múltiples herramientas para personas interesadas en el bienestar animal y la tenencia responsable de mascotas. Dispone de ubicaciones geográficas para identificar servicios, tiendas, urgencias y reportes entre otros.

3. Plan de negocios

En esta sección se presenta el plan de negocios y su modelo para la sustentabilidad del proyecto, cabe destacar que este no aplica directamente sobre el trabajo realizado durante esta memoria para la construcción de software, más bien está orientado a una proyección de desarrollo futuro para que la aplicación sea sustentable una vez alcance el estado de producción.

3.1. Problemas

En base al análisis hecho, se presentan varias problemáticas, como la falta de un directorio centralizado de organizaciones y servicios animalistas, considerando esto y entrevistas realizadas a miembros de fundaciones 30, se concluye que a futuro se debe expandir la aplicación con una sección comercial, que funcione como vitrina de productos y servicios dirigidos a mascotas.

- Los voluntarios autónomos no tienen un modo de operar definido, por lo que tampoco es eficiente ni eficaz al actuar ante una emergencia animal en zonas urbanas.
- El común de las personas manifiesta dificultad para acceder a información de fundaciones, organizaciones y campañas a favor del bienestar animal, dado que los canales no están hechos especialmente para eso.
- Debe existir una estrategia de promoción, difusión y retención de clientes. Esto tiene mejores probabilidades de logro si se elabora desde una solución con alto valor agregado para los clientes. Este puede ser la percepción de las personas de hacer algo bien colectivamente, mejoras en la economía del hogar al comparar precios en el sitio o entregarle los mejores cuidados a sus mascotas.

3.2. Modelo de Negocio

Un análisis enfocado en el modelo de negocio describe las bases sobre las que la plataforma crea, proporciona y capta valor al mismo tiempo que resuelve problemas sociales. Al estar desarrollando un proyecto basado en una metodología ágil se opta por implementar un Lean Canvas fig.3.2.1, el cual fue desarrollado como equipo multidisciplinario durante el programa de memorias, este lienzo se representa en la siguiente figura.

En definitiva, los animales que son maltratados no pueden resolver sus problemas por sí mismos, ésta problemática es abordada por diversas organizaciones creadas específicamente con ese fin, las cuales no tienen un canal fijo de comunicación ni un directorio centralizado que permita a otros actores otorgar información relevante para que puedan llevar su acción social a cabo. Además, para quienes están interesados en aportar en el bienestar de los animales no tienen otras alternativas para poder contribuir a ésta labor que no sean donaciones directas a fundaciones u organizaciones específicas. Normalmente estos actores son representados por los tenedores responsables de mascotas, quienes se encargan de otorgar la mejor calidad de vida a sus mascotas, ya que su percepción hacia ellos ha cambiado a través del tiempo y tienden a considerarlos como un integrante más de la familia.

Segmento Objetivo
Este se define como personas entre 18 y 40 años de edad que sean dueños de mascotas o tengan interés en el bienestar de estas. De esta manera, el modelo planteado considera que las personas que realizan compras en la modalidad online, lo hacen con la intención de recibir el producto en su hogar, reducir los tiempos dedicados a ésta acción y tener la posibilidad de comparar precios, con el fin de disponer de más tiempo y dinero para llevar a cabo sus actividades cotidianas con normalidad. Por otro lado los negocios tienen como intención lograr mas visibilidad y finalmente aumentar las ventas, pero aun así no tienen un canal de ventas web, por lo que deben publicitar sus productos y servicios en otros canales que no están especializados en su público y que no necesariamente llegan a su segmento objetivo.

3.2.1. Lean Canvas

1. Problema

- No existe un modus operandi eficiente y eficaz de voluntarios autónomos ante una emergencia animal en zonas urbanas.
- Existe dificultad para conectar las organizaciones y fundaciones del mundo animal con personas naturales.
- Los usuarios de aplicaciones desean percibir valor agregado con sus acciones al utilizarla, es necesario para mejorar la retención de clientes.

Alternativas existentes

- Redes sociales.
- Comercios electrónicos.
- Aplicaciones de difusión de adopciones o informativas.

4. Solución

- Mapa interactivo para visualizar los casos de emergencias y solicitar ayuda específica a los voluntarios de la zona.
- Sección informativa regulada por fundaciones.
- Canal formal de adopciones.
- Espacio deventa y difusión para emprendimientos, servicios y tiendas relacionadas con las mascotas.

8. Métricas Clave

- Valoración de la aplicación y de los oferentes.
- Descargas mensuales
- Recurrencia de clientes
- Casos exitosos v/s Casos totales.
- Conteo de animales rescatados y adoptados.

3. Propuesta de Valor

Sistema de gestión de voluntariado animal en tiempo real. Es gratuita, informativa e incorpora funcionalidades útiles para replicar buenas prácticas ante una eventualidad con animales.

9. Ventajas Competitivas

- Canal para solicitar ayuda en situaciones de emergencia en tiempo real.
- Información gratuita y visibilización de servicios relacionados a la tenencia responsable de mascotas.

5. Canales

- Página web.
- Aplicación móvil.
- Redes sociales.
- Eventos auspiciados

2. Segmento de Clientes

- Integrantes de fundaciones y organizaciones animalistas.
- Voluntarios autónomos.
- Dueños de mascotas.
- Servicios y tiendas relacionados a las mascotas.

Early Adopters

- Personas de entre 18 y 40 años reacias a ser parte de una fundación, pero interesadas en actuar por el bienestar de los animales.

7. Estructura de costos

- Sueldos.
- Levantar servidores.
- Unidad de desarrollo de software.
- Publicidad, promoción y marketing.
- Publicación en AppStore y Google Play
- Hosting y Dominio web.

6. Flujo de ingresos

- Patrocinadores.
- Venta de productos
- Publicaciones pagadas.
- Fondos concursables.
- Donaciones.
- Inversores

4. Prototipado de la Plataforma

Esta sección se desarrolló como equipo durante el Programa de Memorias Multidisciplinarias, guiada principalmente por la estudiante de Ingeniería en Diseño de Productos Carla Parra Rebolledo, encargada del ámbito de diseño del proyecto, para conocer mas sobre el proceso e iteraciones realizadas para alcanzar el producto final se sugiere revisar su memoria.

4.1. Diseño de la Solución

Una vez definidas las diferentes funciones que deben cumplirse en esta propuesta de solución y al tener las negociaciones correspondientes con la Fundación mandante se comienza con el diseño de la plataforma.

4.1.1. Tipos de Usuarios y Perfiles

Junto con esto, se reconoce que hay distintos tipos de usuarios en la plataforma quienes tendrán diferentes facultades dentro de esta fig.4. Los tipos de usuarios son los siguientes:

- **Usuarios no registrados:** Personas que harán ingreso a la plataforma sin la necesidad de registrarse, para esto, solo se les pedirá su nombre. Esto con el objetivo de que, en caso de presenciar una emergencia, la inexistencia de su cuenta en la aplicación no sea un impedimento en la creación de la alerta. De igual forma tendrán acceso a la sección informativa para incentivar su adquisición de conocimiento.
- **Usuarios comunes:** Usuarios que se registran para ingresar a la aplicación, para esto, se les solicitará su nombre, correo electrónico y una contraseña.
- **Voluntarios:** Usuarios con mayores facultades dentro de la aplicación. Se destacan por ser quienes atenderán a los diferentes casos anunciados y tendrán la posibilidad de formar parte de Organizaciones y Fundaciones. Para volverse voluntario, los usuarios comunes deberán ingresar su RUT, en caso de ofrecer transporte deberán dar a conocer la patente del vehículo a utilizar y, por otro lado, si desean ser hogar temporal, deberán ingresar el sector en el que se encuentra. Estos requerimientos son necesarios para velan por la integridad de la persona ayudada y que esta pueda reconocer de mejor manera al voluntario que acude a su ayuda.

  Los voluntarios que pertenezcan a fundaciones tendrán la facultad de agregar publicaciones dentro de la sección informativa debido a que las fundaciones, al estar legalmente constituidas, poseen mayores conocimientos sobre el bienestar animal y tienen acceso a información verídica que podrán difundir a la comunidad.

  Es importante mencionar que los voluntarios tendrán la posibilidad de activar y desactivar su Modo Voluntario el cual, al estar activo, dará facultades a la aplicación para darle aviso mediante notificaciones cada vez que aparezca un nuevo caso de emergencia en sus cercanías.

El detalle de las facultades que tendrán cada uno de los tipos de usuarios en la plataforma se encuentra en la siguiente figura:

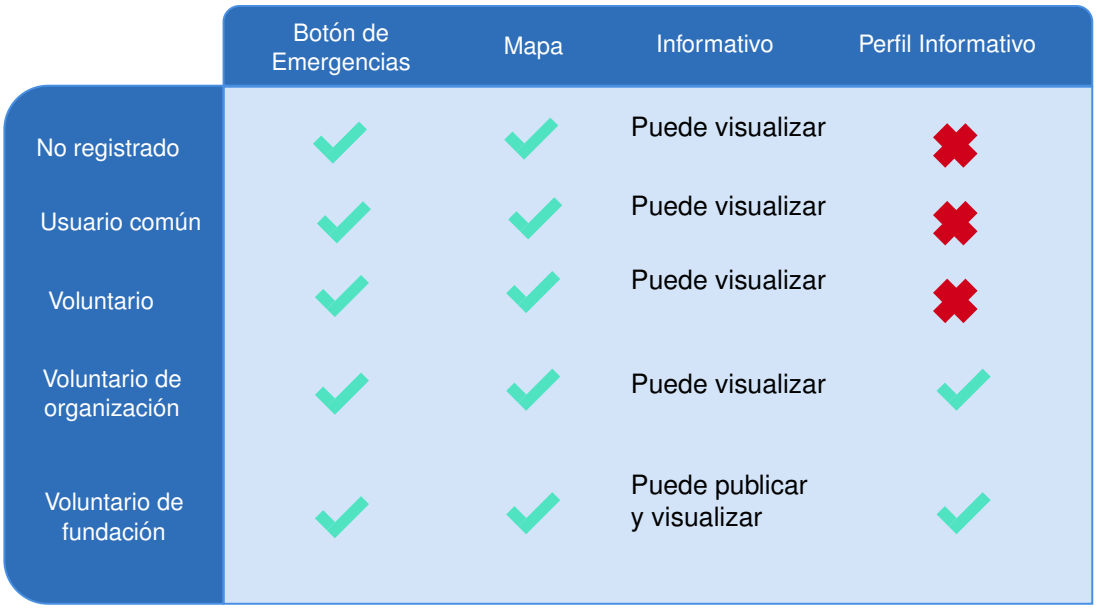


Figura 4: Tipos de usuarios y sus facultades

Las organizaciones y fundaciones estarán presentadas como perfiles informativos, los cuales serán administrados por el conjunto de voluntarios que formarán parte de estos. Podrán encontrarse en la sección informativa y poseerán medios de contacto para que las personas puedan acudir a ellas cuando lo necesiten, además del detalle de sus miembros activos.

- El perfil de Organización podrá ser creado por un voluntario, en donde se solicitará el nombre de la entidad, los medios de contacto y por último se debe señalar al menos tres usuarios voluntarios que formen parte de esta.
- Una vez existente el perfil informativo de la Organización, esta puede pasar a ser una Fundación siempre y cuando sea una organización legalmente constituida, para esto debe presentar el RUT de la personalidad jurídica, los RUT de los miembros de la directiva de esta incluida la presidencia, tesorería y secretaría.

4.1.2. Elementos de la Plataforma

Se determina la existencia de cinco elementos fundamentales que componen esta solución para satisfacer las necesidades del contexto presentado.

- Casos reportados: Se desarrolla un mapa interactivo utilizando la API de Google Maps, en el cual se podrán visualizar los casos previamente reportados por todo tipo de usuarios, en el caso de los usuarios voluntarios ellos podrán tomar los casos reportados, resolverlos y gestionarlos.
- Sección de noticias: Corresponde a la vista de inicio al acceder a la aplicación, en esta se podrán visualizar todo tipo de notas informativas relacionadas a la comunidad y bienestar animal, segmentadas en categorías como Salud, Protocolos, Leyes entre otros.
- Botón de emergencia: Es la sección donde todo tipo de usuario podrá reportar un caso que requiera de un voluntario para resolver la situación, estos casos son categorizados dependiendo de la ocurrencia como por ejemplo un animal perdido, abandono, animal herido entre otros. Posteriormente el usuario que reporta podrá adjuntar descripción del caso, fotografía, especie, que se necesita y por ultimo ubicación geográfica, para que esta alerta sea desplegada en el mapa interactivo.
- Perfiles informativos: Dentro de la sección informativa, existe un directorio con la información detallada de todas las Fundaciones y Organizaciones existentes en la plataforma, una descripción propia, sus medios

de contacto, miembros activos y todas las notas informativas o noticias que ellos hayan generado.

- Menú lateral: Transversalmente a los elementos antes mencionados, la aplicación contiene un menú desplegable lateral, el cual cuenta con distintas funcionalidades dependiendo del tipo de usuario, ya sea invitado, registrado o voluntario, en caso de ser voluntario y pertenecer a una organización se puede visualizar su perfil informativo, y si esta fuera una fundación legalmente constituida, tendrá acceso a un editor de texto para generar y redactar notas informativas.

4.1.3. Nombre y logo de la Plataforma

Para el nombre de la plataforma se decidió por Wellpets, el logo puede verse en la fig.5, estos elementos fueron verificados con diferentes usuarios, fundaciones y cliente, el detalle de este proceso junto con las iconografías y colores pueden ser revisados en la memoria de Carla Parra.

Figura 5: Logo Wellpets

5. Desarrollo de la Plataforma

En la presente sección se expone la solución a los problemas presentados anteriormente, aquí se abordan las diferentes áreas que permiten su desarrollo desde un punto de vista técnico. Se presentan los requerimientos de la aplicación, y se delimitan los esenciales para cumplir con un producto mínimo viable (MVP), se definen en detalle las diferentes tecnologías a utilizar, en cada uno de los diferentes niveles lógicos que permiten un correcto funcionamiento de la plataforma y las correspondientes arquitecturas, modelos y diagramas implementados. Se destaca que si bien se realiza un trabajo en conjunto en el proceso diseño de software, toma de decisiones, diagramación, análisis de requisitos y planificación de la plataforma, el Frontend de esta es desarrollado por el estudiante Jorge Fernández y el Backend por Lorens Páez.

5.1. Análisis de Requerimientos

Tras reuniones con la fundación mandante, se concreta la toma de requisitos funcionales, no funcionales y sus correspondientes refinamientos, para conocer aquellas características a desarrollar desde la perspectiva del producto y su usuario, además de los factores de calidad que se deben tener en cuenta a la hora de comenzar su construcción.

5.1.1. Requisitos funcionales iniciales

RF1: La aplicación debe permitir crear cuentas de usuario

RF2: La aplicación debe permitir ingresar con email y contraseña.

RF3: La aplicación debe permitir ingresar como invitado sin cuenta, teniendo restricciones al utilizarla.

RF4: La aplicación debe permitir que usuarios registrados completen un formulario para ser voluntarios.

RF5: La aplicación debe diferenciar entre las aptitudes que cada voluntario ofrece.

RF6: La aplicación debe permitir que los voluntarios se asocien a organizaciones y/o fundaciones.

RF7: La aplicación debe permitir a los voluntarios crear perfiles informativos para organizaciones.

RF8: La aplicación debe permitir que las organizaciones completen un formulario para validarse como fundación.

RF9: La aplicación debe permitir a los usuarios registrados administrar sus perfiles.

RF10: La aplicación debe permitir que cualquier tipo de usuario solicite ayuda en caso de una emergencia, entregando su ubicación y descripción.

RF11: La aplicación debe permitir solo a los usuarios voluntarios atender situaciones de emergencia.

RF12: La aplicación debe mostrar servicios y organizaciones relacionados al cuidado de mascotas y comercio afín.

RF13: La aplicación debe permitir que usuarios registrados den una valoración a servicios, organizaciones, comercios y fundaciones.

RF14: La aplicación debe mostrar información relacionada al bienestar animal, ley y protocolos.

RF15: La aplicación debe permitir que solo los voluntarios publiquen en nombre de las fundaciones en la sección informativa.

RF16: La aplicación debe permitir la difusión de animales en adopción.

RF17: La aplicación debe permitir que usuarios registrados llenen un formulario de adopción.

RF18: La aplicación debe tener un medio de comunicación abierta para que los usuarios registrados puedan publicar.

RF19: La aplicación debe permitir que solo las fundaciones moderen/administren las secciones de adopciones y foro.

5.1.2. Requisitos no funcionales inciales

RNF1: Encriptación de información sensible (ej: contraseñas).

RNF2: Interfaz intuitiva para cualquier usuario.

RNF3: La aplicación debe ser escalable.

5.2. Requisitos funcionales y no funcionales acotados al Producto Mínimo Viable

Dado el acotado tiempo de trabajo dentro del programa de memorias multidisciplinarias, es que se realiza un proceso de priorización para establecer el alcance del producto mínimo viable junto al cliente, de manera que los requisitos previamente captados se reducen a los siguientes.

5.2.1. Requisitos funcionales

RF1: La aplicación debe permitir crear cuentas de usuario.

RF2: La aplicación debe permitir ingresar con email y contraseña.

RF3: La aplicación debe permitir ingresar como invitado sin cuenta, teniendo restricciones al utilizarla.

RF4: La aplicación debe permitir que usuarios registrados completen un formulario con datos personales para ser voluntarios.

RF5: La aplicación debe permitir que los voluntarios se asocien a organizaciones y/o fundaciones.

RF6: La aplicación debe permitir a los voluntarios crear perfiles informativos para organizaciones y fundaciones.

RF7: La aplicación debe mostrar información relacionada al bienestar animal, ley y protocolos.

RF8: La aplicación debe permitir que solo las fundaciones moderen/administren la sección informativa.

RF9: La aplicación debe permitir que cualquier tipo de usuario solicite ayuda en caso de una emergencia, entregando su ubicación y descripción.

RF10: La aplicación debe permitir solo a los usuarios voluntarios atender situaciones de emergencia.

5.2.2. Requisitos no funcionales

RNF1: Encriptación de información sensible (ej: contraseñas).

RNF2: Interfaz intuitiva para cualquier usuario.

RNF3: La aplicación debe ser escalable.

5.3. Definición de la Solución

Al considerar todos los requisitos funcionales y no funcionales captados, la propuesta de solución completa se define en conjunto con la fundación mandante como una aplicación que cumple con las funcionalidades que se observan en la fig.6:

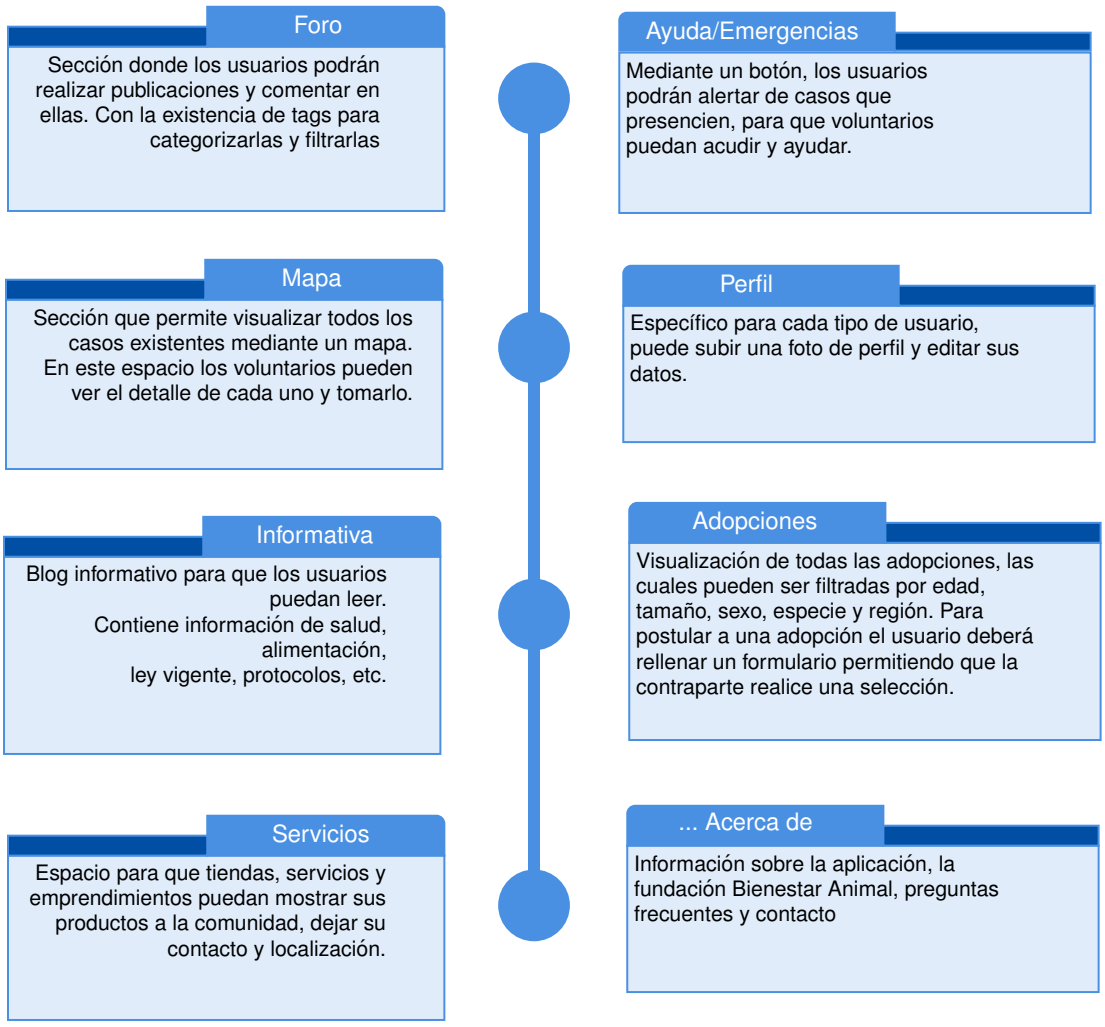


Figura 6: Diagrama Propuesta final secciones

5.4. Producto Mínimo Viable

Como se menciona anteriormente en los requisitos funcionales y no funcionales, durante el periodo de esta memoria se desarrolla un producto mínimo viable derivado de la propuesta de solución completa antes descrita. Este MVP se define junto al cliente en un proceso de priorización para determinar cuales de las funcionalidades propuestas satisfacen los principales dolores de la comunidad animalista, de manera que los elementos presentes en la plataforma se describen en la fig.7

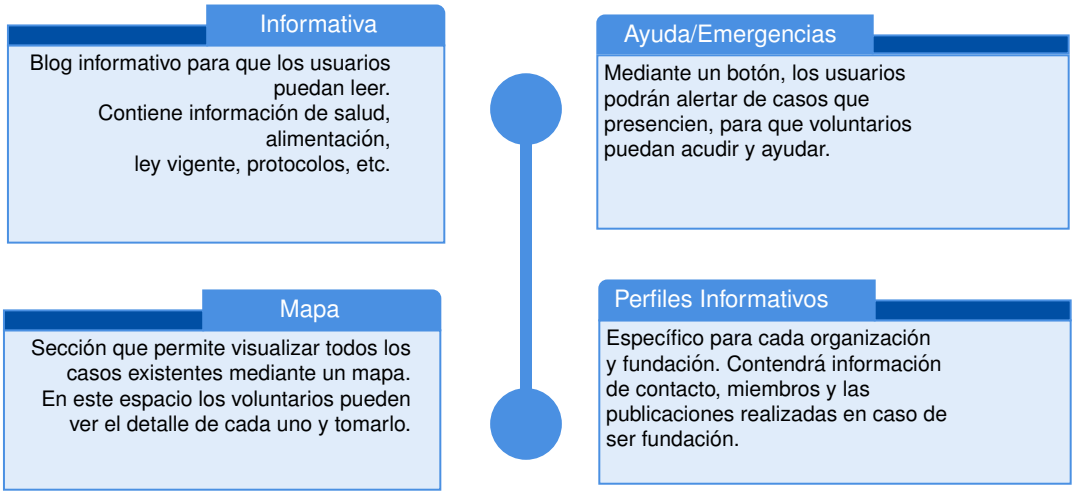


Figura 7: Secciones desarrolladas dentro del Programa de Memorias

5.5. Arquitectura del sistema

5.5.1. Modelo de Dominio del sistema

Un modelo de dominio es una representación visual y estructurada que describe las entidades, relaciones y atributos clave dentro de un sistema. En el contexto del desarrollo de un producto mínimo viable (MVP), este modelo ilustra cómo las entidades identificadas en los requisitos funcionales interactúan entre sí y con las secciones anteriormente definidas de la aplicación.

El enfoque de este esquema ayuda a clarificar la estructura esencial del sistema, facilita la comunicación entre los miembros del equipo de desarrollo y establece una comprensión compartida del dominio del problema que están abordando, lo que es fundamental para el éxito del proyecto.

A continuación en la fig.8se expone el modelo de dominio que se ha propuesto para el desarrollo del producto mínimo viable.

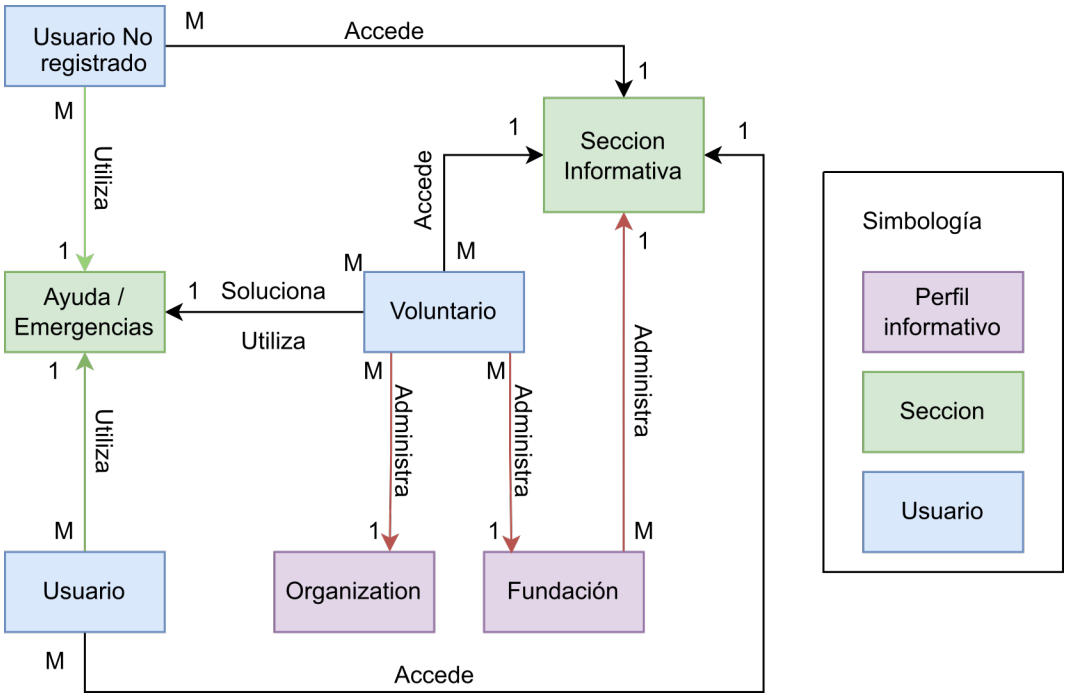


Figura 8: Modelo de Dominio del sistema propuesto

5.5.2. Modelo de datos relacional

Un modelo relacional es una representación estructurada de la información en una base de datos relacional(SQL), la cual utiliza tablas para organizar los datos y establecer relaciones entre ellos, las cuales se ven representadas por los distintos tipos de lineas simbólicas que las unen, denotando la cardinalidad entre estas. En este modelo cada tabla representa una entidad del sistema como lo pueden ser Usuarios, Organizaciones y Alertas; cada una de estas puede poseer múltiples atributos que denotan la información esencial de aquella entidad.

Dados los requisitos funcionales obtenidos y la propuesta de solución para el prototipo funcional que fue validada por el cliente y distintos tipos de usuario, se define el siguiente modelo de datos relacional para el sistema fig.9:

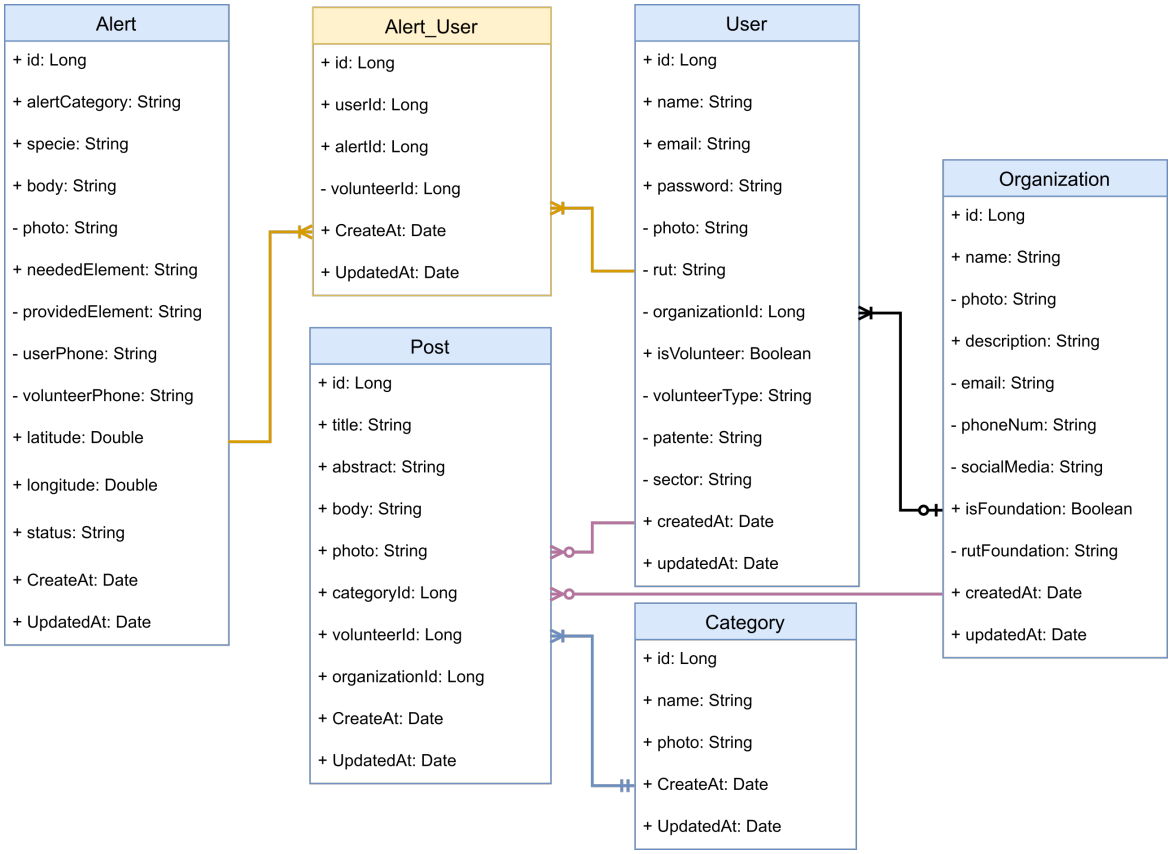


Figura 9: Modelo de datos

5.6. Elección de Tecnologías

Respecto a las tecnologías utilizadas, se debe tener en cuenta los requisitos que debe cumplir el prototipo esperado. Con el objetivo de llegar a la mayor cantidad de público posible, es que se ha requerido que la aplicación a construir sea multiplataforma, lo que implica que debe funcionar para dispositivos móviles y sitio web.

5.6.1. Tecnologías Frontend

Se debe considerar los bajos recursos monetarios que posee la Fundación para dar continuidad al proyecto a futuro y el tiempo disponible por parte del actual equipo desarrollador, ya que el cliente ha solicitado el mayor alcance posible. Tomando en cuenta lo anterior, los requisitos del software y aquellas necesidades mínimas que debe cubrir, la herramienta a elegir debe facilitar el cumplimiento de estos. Entre estas necesidades se destacan las siguientes:

- Facilitar el desarrollo multiplataforma para dar un mayor alcance a los usuarios.
- Poder consultar APIs (Backend y google maps).
- Acceder a la geolocalización del dispositivo.

Como equipo al considerar los criterios anteriores, se propone el uso de Flutter[27] para el desarrollo del Frontend de la aplicación. El cual es un kit de desarrollo de software(SDK) de código abierto, respaldado por Google, para crear aplicaciones multiplataforma compiladas de forma nativa a partir de una única base de código y posee las siguientes características:

- Permite crear aplicaciones para cualquier plataforma o sistema operativo utilizando un mismo código, lo que en un futuro permitirá abaratar costos, ya bastará sólo con un equipo, para que mantenga el código.
- Compila el código directamente a lenguaje máquina, por lo que tiene un mejor rendimiento que su competidor react native, ya que este último utiliza un puente para conectar componentes nativos del sistema operativo con Javascript[26].
- Es posible inyectar código nativo (Swift / ObjC, Java / Kotlin y C++), lo que es una gran ventaja a la hora de querer agregar funcionalidades en un sistema operativo específico.
- Hot reload, permite ver los cambios de forma instantánea y aumenta la productividad del desarrollo.

5.6.2. Tecnologías Backend

El Backend del sistema debe ser capaz de proveer un servicio de autenticación, almacenamiento de datos, permitir hacer operaciones CRUD (crear, leer, actualizar y eliminar) sobre estos datos almacenados y debe recibir y responder a las consultas realizadas en el cliente del Frontend.

Para construir el servidor Backend, se destaca el hecho de que el proyecto no termina en el periodo de memorias multidisciplinarias, dados los requerimientos del cliente se considera la incorporación de nuevos desarrolladores al equipo una vez se construya el producto mínimo viable. Junto con lo anterior, se debe tener en cuenta que el producto completo debe escalar considerablemente, por lo que se propone el uso de del framework NestJS[8] debido a sus cualidades como se describen a continuación:

- Framework construido sobre el entorno de ejecución Node.js[10] que utiliza módulos del framework Express[11] y Fastify[12].
- Desarrollado sobre el lenguaje de programación Typescript y también compatible en su totalidad con Javascript.
- Implementa el patrón de diseño modelo vista controlador(MVC), que permite modularizar y permitir la construcción de software eficiente y escalable.

Dado lo anterior, se plantea que sus características son ideales para el contexto actual y futuro. Al ser un framework que funciona sobre Express, el cual también ya es un framework, proporciona herramientas para producir código con mucha rapidez, lo que es importante considerando el tiempo de desarrollo disponible. Por otro lado con el uso de Typescript y el patrón de diseño MVC sobre el cual opera, nos proporciona las bases necesarias para la construcción de un software altamente escalable.

En cuanto a la definición de la base de datos, dado que el software a desarrollar posee requerimientos donde la jerarquización de entidades y relación entre estas define gran parte de las funcionalidades, se considera oportuno y de mayor utilidad utilizar una base de datos relacional (SQL). Para esto se propone el uso de PostgreSQL[13] con las siguientes cualidades:

- Proyecto open source desarrollado por una comunidad de desarrolladores.
- Es una base de datos relacional de objetos (Object-SQL) lo que nos permite manipular tipos de datos más complejos que otras bases de datos como MySQL[14].
- Ofrece compatibilidad de operar en distintos lenguajes de programación distintos a SQL como suelen ser las bases de datos de tipo relacional.

Por otro lado se decide externalizar el proceso de generar notificaciones para las alertas en tiempo real, por lo que se propone el uso de herramientas de servicios en la nube proporcionadas por Firebase[34] para la implementación de estos módulos. Firebase es una plataforma en la nube actualmente sostenida por Google, utilizada principalmente para el desarrollo de aplicativos web y móviles.

- Utiliza los servidores de Google y permite escalabilidad en todo tipo de aplicaciones.
- Facilita la creación de proyectos sin la necesidad de levantar un servidor gracias a las herramientas que trae en su kit de desarrollo de software (SDK).

5.7. Estructura del Sistema

Para la construcción del sistema, se considera que la arquitectura del Backend de la plataforma tendrá un formato REST API como se indica en el diagrama de la fig.10:

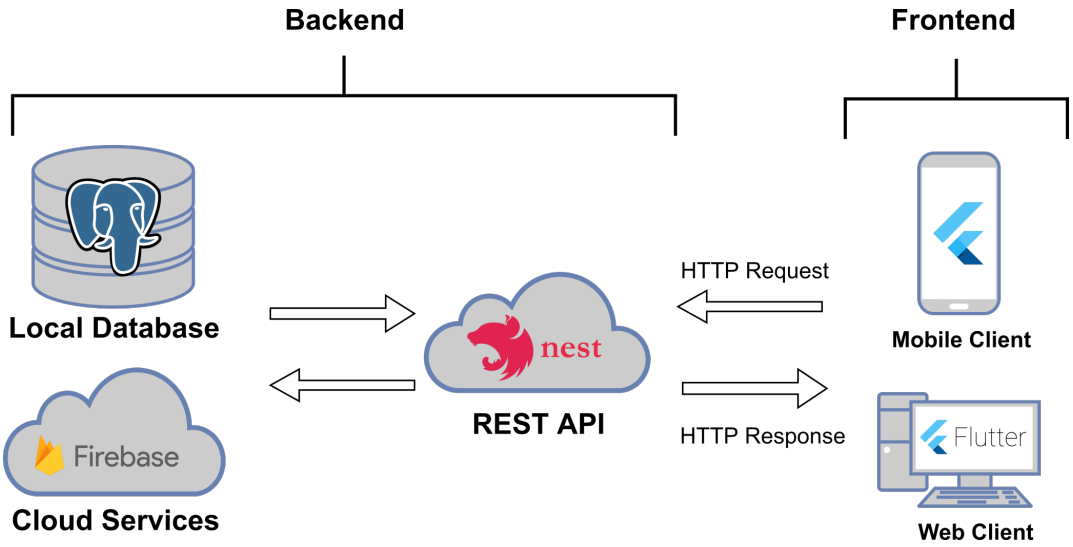


Figura 10: Arquitectura de Sistema

5.8. Implementación Backend

El servidor Backend permite que el cliente de la aplicación móvil y web puedan almacenar y obtener datos desde la base de datos, como también hacer uso de servicios de autenticación, registro, notificaciones entre otros.

Como se presentó anteriormente, el Backend es construido utilizando Nest, Framework de Node, el cual funciona sobre la base de Express pero en un nivel de abstracción superior, además es pertinente destacar que Nest está fuertemente inspirado en AngularJS[15], utilizando también Typescript como lenguaje principal y admitiendo la utilización de Javascript puro en su código.

El servidor es implementado aplicando una arquitectura modular para construir software escalable, de manera similar al patrón de diseño Modelo-vista-controlador, con la diferencia de que la generación de vistas es provisto por el Frontend de la plataforma. Así es como a nivel de código, al implementar los distintos módulos de las entidades presentes en la base de datos, se hace la separación de controlador, servicio y su modelo.

- **Modelo de datos:** Es la representación de las entidades de la base de datos en la API, aquí se define el modelo utilizando el patrón Data Transfer Object[16](DTO). Dado que el servidor Backend es el intermediario entre el Frontend y la base de datos, la manera de representar las entidades se hizo utilizando estos objetos que transportan datos entre procesos.
- **Controladores:** Estos contienen todas las rutas para captar las peticiones HTTP enviadas desde el Frontend, estas deben ser recibidas por la API. El controlador recibe estas peticiones en cada una de las rutas un JSON[17] String el cual debe coincidir con el modelo de la entidad definida en el DTO, posteriormente cada ruta utiliza el o los servicios necesarios para operar los datos.
- **Servicios:** Aquí se definen todos los métodos asociados a una entidad, utilizando los DTO definidos para recibir los datos que llegan a las rutas en el controlador y manipular la información si fuera necesario antes de almacenar los datos en la base de datos.

Para manipular, gestionar, filtrar los datos entre otras operaciones, se utiliza Prisma, un ORM[18] (Object-Relational mapping) desarrollado para complementar con aplicativos que utilizan Node y Typescript. Prisma no solo proporciona la asignación de objetos al modelo relacional, si no que también provee una API interactiva para desarrolladores, de manera que no es necesario utilizar un gestor de base de datos. Este ORM también proporciona métodos gracias a su componente Prisma CLI[20], para generar migraciones automatizadas.

Es también compatible con múltiples bases de datos, dentro de las cuales se puede conectar ampliamente y sin problemas con PostgreSQL. Con este ORM se genera un documento esquema que puede ser construido tanto en Node como Typescript, aquí se definen todas las entidades del modelo de datos, atributos, tipos, relaciones, condiciones y su jerarquía. Finalmente para generar el modelo en la base de datos se aplican los comandos que nos provee Prisma CLI, para realizar las migraciones correspondientes.

5.8.1. Integridad de los Datos

Para que el Backend proporcione un servicio API REST seguro, se implementa el uso de tokens para mantener sesiones de usuario, prevenir secuestro de estas y también para proveer un servicio de autenticación. Lo anterior se realiza utilizando JSON Web Tokens[21](JWT), cuya estructura se muestra en la fig.11, estos corresponden a un estándar utilizado para asegurar la identidad entre dos partes comunicantes.

A nivel de código se debe importar la biblioteca Passport[23] de Node.js, la cual implementa JWT respaldado por el estándar RFC 7519[22], de manera que este importe nos permite generar, verificar y decodificar tokens.

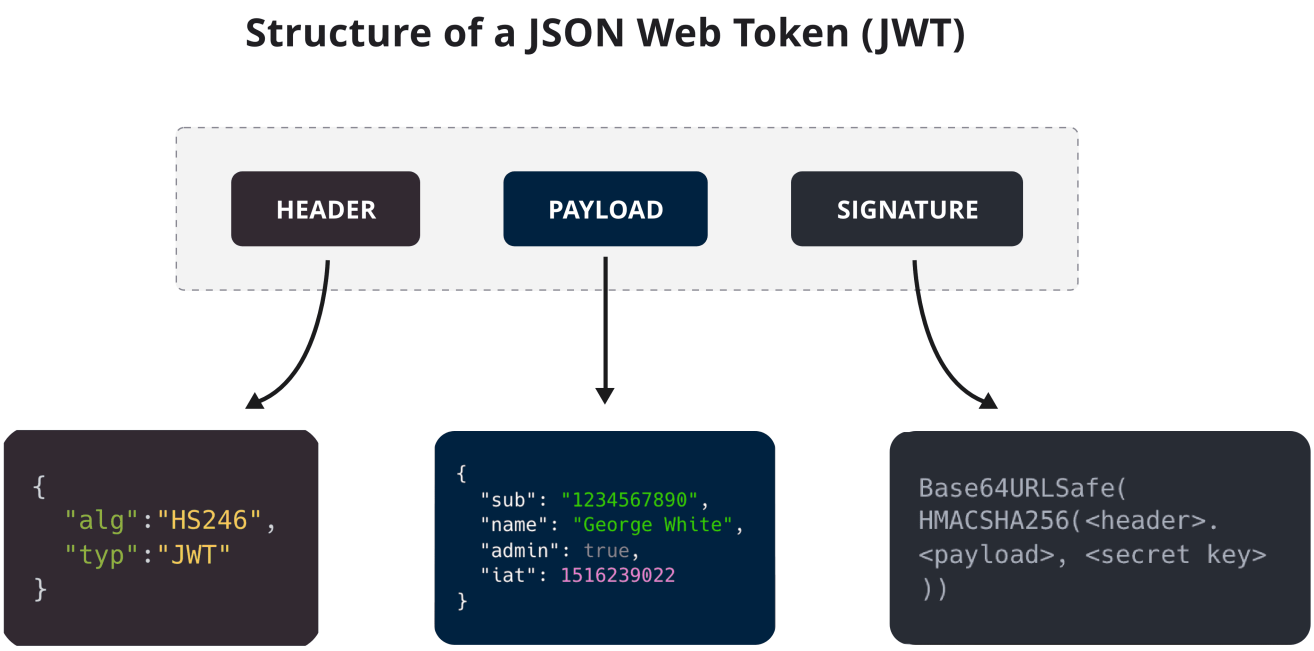


Figura 11: Estructura de un JWT

Dado que los requisitos funcionales y no funcionales requieren de un servicio de registro e ingreso de usuarios a la plataforma, que asegure integridad de los datos, aquí se aplica el uso de JWT para asegurar la identidad del usuario al momento de acceder a la aplicación, previniendo el secuestro de sesiones o suplantación.

Passport también nos provee el elemento Guard[24], esta herramienta es la encargada de aplicar la estrategia de la biblioteca, de forma que cuando se invoca al Guard en una ruta definida en un controlador, este se encarga de la recuperación de credenciales, ejecución de la función de verificación, creación de la propiedad del usuario entre otras funcionalidades. Como elemento complementario a la seguridad de los datos, se implementa la biblioteca Argon2[25], que nos provee métodos de hashing para proteger datos sensibles como las credenciales de los usuarios, de esta manera en la base de datos solamente se almacena el hash generado y nunca la contraseña provista por el usuario.

5.8.2. Gestión de Usuarios

Usuario invitado
Como se presentó en los tipos de usuarios que la plataforma requiere según la toma de requisitos, existe el usuario invitado, el cual puede acceder a la plataforma sin necesidad de tener una cuenta registrada, recordando que esto tiene la finalidad de permitir un rápido acceso para reportar casos de urgencia y no limitar la posibilidad de difundir información que educativa a la comunidad en temas de bienestar animal.

Para delimitar y restringir el accionar de usuarios con identidad desconocida para el sistema, se utilizan las herramientas que nos provee Passport. Esto se aplica invocando a un Guard en cada endpoint que implique manipulación de datos, de manera que solo se deja sin Guard a las rutas asociadas a la creación de alertas y lectura de notas informativas, para que no se requiera de validar la identidad del usuario.

Usuario registrado y voluntario
Como se menciona anteriormente, todo endpoint que permita la manipulación de datos, requiere la invocación de un Guard para reconocer la identidad del usuario. A nivel de código, lo anterior de complementa con el uso

de un decorador que recibe los datos del token y entrega al sistema un objeto que contiene todos la del usuario identificado.

Se recalca que estos usuarios no tienen las mismas facultades, ya que existen 4 casos distintos. El primer paso para diferenciar los roles y las facultades de cada usuario ocurre cuando el Guard identifica al usuario e informa al sistema su identidad, esto es cuando un usuario realiza una petición HTTP a alguna ruta del Backend. Luego cada ruta hace un llamado a los servicios de la correspondiente entidad, donde se aplica la siguiente lógica de negocio:

1. **Usuario común registrado:** Sus datos son almacenados en la base de datos del sistema, la única diferencia con el invitado en cuanto a sus facultades, es que adquiere la capacidad de rellenar el formulario para convertirse en un voluntario.
2. **Usuario voluntario:** Mediante un atributo en el modelo de la base de datos que define su rol de voluntario, este usuario adquiere acceso a nuevas funcionalidades al utilizar la aplicación. En el Backend esto se ve reflejado de manera que al identificar su rol de voluntario, puede ahora enviar peticiones HTTP desde el Frontend a los endpoints de Crear Organización, Tomar Alertas, Cerrar Alertas, Eliminar Alertas, Activar Modo Voluntario entre otros.
3. **Usuario voluntario que pertenece a una organización:** Como se menciona antes, el voluntario tiene la capacidad de crear una organización y también puede ser agregado a una por otro voluntario. Al ser miembro de una, esto se ve reflejado en la base de datos, donde ahora existe la relación entre el usuario y la fundación a la que pertenece.
4. **Usuario voluntario que pertenece a una Fundación legalmente constituida:** En caso de que la organización sea una Fundación legalmente constituida, los voluntarios miembros tienen la capacidad de actualizar el tipo de organización dentro de la plataforma. Lo anterior se diferencia mediante un atributo en el modelo de organización, al hacer esto, los miembros de la fundación adquieren nuevas funcionalidades en la plataforma, donde se destaca la facultad de redactar notas informativas, que serán publicaciones en la sección informativa, es decir, estos voluntarios tienen ahora la capacidad de enviar peticiones HTTP a los endpoints correspondientes a la entidad de Publicaciones para generar nuevos registros.

Push Notifications

Para brindar un eficaz funcionamiento del sistema de alertas que se construye, se implementan Push Notifications[35], un servicio provisto por la plataforma Firebase para generar notificaciones a los dispositivos de usuarios de la aplicación, en particular se crean notificaciones cuando los usuarios crean alertas y luego estos deben ser notificados cuando el caso creado ha sido atendido por un voluntario.

Se define un atributo en el modelo de los usuarios el cual contiene un token de acceso al servicio de Firebase, la generación de este token se produce en el Frontend, la cual es luego enviada al Backend y almacenada en la base de datos. De esta manera cuando un voluntario toma un caso abierto, el sistema se encarga de revisar a que usuario pertenece dicha alerta y así podemos conocer su token de notificaciones, para posteriormente realizar una operación POST a la API de firebase, la información enviada utilizando el módulo HTTP contiene el token de dicho usuario, una cabecera y cuerpo del mensaje que luego sera enviado al dispositivo del usuario.

5.8.3. Módulos de Backend

Para el desarrollo del servidor Backend se consideran 5 módulos que representan las principales entidades del modelo de datos MB1(Usuario), MB2(Organización), MB3(Alerta), M4(Publicaciones) Y MB5(Categorías) que son descritos en las siguientes tablas:

- Módulo de Usuario **(MB1)**

ID	Método HTTP	Ruta	Recibe	Entrega	Función
1	POST	/users/signup	Datos del usuario	Entrega un usuario registrado	Registrar un usuario
2	POST	/users/signin	Datos del usuario	Entrega un usuario ingresado	Iniciar sesión de un usuario
3	GET	/users/:id	Id del usuario	Entrega un usuario	Obtener información de un usuario
4	GET	/users/		Entrega todos los usuarios	Obtener todos los usuarios registrados
5	PATCH	/users/:id	Id del usuario	Entrega un usuario actualizado	Actualizar información de usuario
6	PATCH	/users/:id	Datos de voluntario	Entrega un usuario actualizado a voluntario	Actualizar su estado de voluntario
7	PATCH	/users/activate Mode	Id del voluntario	Entrega un usuario actualizado	Activar o desactivar modo voluntario
8	DELETE	/users/:id	Id del usuario	Entrega un usuario eliminado	Eliminar un usuario

Tabla 3: Tabla de Módulo Usuario

- Módulo de Organización **(MB2)**

ID	Método HTTP	Ruta	Recibe	Entrega	Función
1	POST	/org/	Datos de la organización	Entrega una organización creada	Crear una organización
2	GET	/org/:id	Id de organización	Entrega una organización	Obtener información de una organización
3	GET	/org/		Entrega todas las organizaciones	Obtener todas las organizaciones existentes
4	GET	/org/users/:id	Id de organización	Entrega los usuarios pertenecientes a una organización	Obtener usuarios de una organización
5	PATCH	/org/:id	Id de organización	Entrega una organización actualizada	Actualizar la información de una organización
6	PATCH	/org/:id	Datos de la Fundación	Entrega una organización actualizada a fundación	Actualizar una organización a fundación
7	DELETE	/org/:id	Id de organización	Entrega una organización eliminada	Eliminar una organización

Tabla 4: Tabla de Módulo Organización

- Módulo de Alerta **(MB3)**

1	POST	/alert/	Datos de alerta	Entrega una alerta creada	Crear una alerta
2	GET	/alert/:id	Id de la alerta	Entrega datos de una alerta creada	Obtener información de una alerta
3	GET	/alert/		Entrega una lista de alertas	Obtener las alertas
4	GET	/alert/myalerts	Id del usuario	Entrega una lista de alertas	Obtener alertas de un usuario
5	GET	alert/:alertKey	alertKey de la alerta	Entrega una alerta	Obtiene la alerta de un usuario
6	PATCH	/alert/:id	Datos de alerta	Entrega una alerta actualizada	Actualizar el estado de una alerta
7	PATCH	/alert/takeAlert/:id	Id de la alerta	Entrega una alerta ahora tomada por un voluntario	Tomar alerta por un voluntario
8	PATCH	/alert/closeAlert/:id	Id de la alerta	Entrega una alerta ahora cerrada	Cerrar una alerta por un voluntario
9	POST	/alert/alerts/cloneAlert/:id	Id de la alerta	Entrega una alerta que hereda información de la primera	Generar una alerta hija vinculada a la primera
10	DELETE	/alert/:id	Id de la alerta	Entrega una alerta eliminada	Eliminar una alerta

Tabla 5: Tabla de Módulo Alerta

- Módulo de Publicación **(MB4)**

1	POST	/post/	Datos del post	Entrega un post creado	Crear un post
2	GET	/post/:id	Id del post	Entrega un post	Obtener información de un post
3	GET	/post/:catId	Id de la categoría	Entrega una lista de post pertenecientes a una categoría	Obtener los post por categoría
4	GET	/post/foundation/:orgName	Nombre de la organización	entrega una lista de post	Obtener los post de una organización
5	GET	/post/category/:categoryName	Nombre de la categoría	entrega una lista de post	Obtener los post de dada categoría
6	PATCH	/post/:id	Datos del post	Entrega un post actualizado	Actualizar la información de un post
7	DELETE	/post/:id	Id del post	Entrega un post eliminada	Eliminar un post

Tabla 6: Tabla de Módulo Publicación

- Módulo de Categoría **(MB5)**

1	POST	/cat/	Datos de la categoría	Entrega una categoría creada	Crear una categoría
2	GET	/cat/:id	Id de la categoría	Entrega una categoría	Obtener información de una categoría
3	GET	/cat/		Entrega una lista de categorías	Obtener las categorías
4	PATCH	/cat/:id	Datos de la categoría	Entrega una categoría actualizada	Actualizar la información de una categoría
5	DELETE	/cat/:id	Id de la categoría	Entrega una categoría eliminada	Eliminar una categoría

Tabla 7: Tabla de Módulo Categoría

5.9. Implementación Frontend

5.9.1. Arquitectura de software Frontend

Se utiliza la arquitectura bloc[28], introducida por Google en su conferencia I/O de 2018[29], la idea detrás, separar la lógica de las vistas y escribir un código más fácil de testear y corregir.

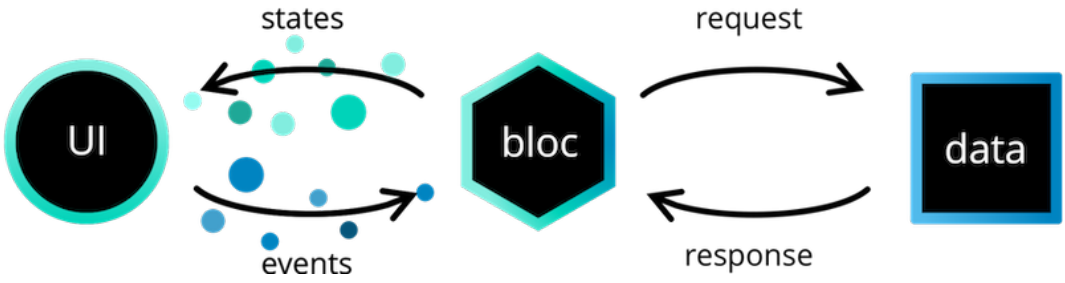


Figura 12: Arquitectura Bloc

Para esto se separa la aplicación en tres capas fig.12, que se explicarán más adelante:

- **Presentación**
- **Lógica de negocio (Bloc)**
- **Datos**
  - **Repositorio**
  - **Modelo**
  - **Proveedor de datos**

Datos - Proveedor de datos
Esta capa es el nivel más bajo de la aplicación e interactúa con bases de datos peticiones http al servidor y otras fuentes de datos asíncronas.

Peticiones Http Para comunicar la aplicación con el servidor se sigue la siguiente lógica.

1. **Verificar conexión a internet:** Usando el paquete Connectivity Plus[37] se verifica que el usuario tenga conexión a internet, en caso contrario se entrega un error.
2. **Petición http:** Si hay conexión a internet se utiliza el paquete Http[36], el cual permite instanciar un cliente para hacer peticiones http.
3. **Manejar la respuesta http:** Al hacer la petición el cliente entrega una respuesta con un código de estado y un contenido. Si el código de estado es 200, entonces se entrega el objeto solicitado, decodificando el JSON que se encuentra en el contenido, en caso contrario se entrega un error con el mensaje enviado por el servidor.
4. **Guardado de datos:** Esta estrategia se utiliza para evitar sobre-cargar al servidor con peticiones http, para esta aplicación existen dos escenarios:
   - **Memoria:** Para datos que no suelen cambiar mucho y que podrían provocar múltiples peticiones al salir y entrar de una vista, es mejor guardarlos en memoria y proveer de una funcionalidad para actualizarlos en caso de ser necesario.

- **Base de datos local:** En caso de necesitar persistencia se utiliza la base de datos local, para que los datos sigan guardados aunque la aplicación se cierre, hacer esto, es necesario en el caso de guardar un token o un usuario invitado.

Firebase Firebase con su modelo de negocio freemium, servicios gratuitos, pero limitados, que se mejoran al hacer un pago, es una gran aliado a la hora de crear un MVP.

- **Guardado de imágenes:** La aplicación permite incluir imágenes en publicaciones y alertas, para esto se utilizó el servicio Storage[34] de Firebase. Este servicio ofrece un almacenamiento bastante generoso gratuitamente, que con una buena gestión, podría nunca necesitar de pagar para utilizar el almacenamiento ilimitado.
- **Messaging:** Firebase ofrece su servicio Cloud Messaging[35] totalmente gratuito y sin limitaciones, por lo que fue utilizado en el Backend, para mandar Push Notifications[38] a las aplicaciones de Android e iOS, que las consumian y mostraban al usuario.

Base de datos local Como base de datos se utilizan dos paquetes, Flutter Secure Storage[39] para mobile y Shared Preferences[40] para web, debido a que actualmente los navegadores no soportan encriptado y guardan sus datos en un almacenamiento local del navegador. En caso de Flutter Secure Storage[39], permite usar Keychan[32] en iOS y Encrypted Shared Preferences[33] en Android para almacenar los datos encriptados en el teléfono del usuario.

Datos - Modelo
Contiene las entidades de la aplicación y permite codificar en formato JSON, en caso de querer guardarla en base de datos local y decodificar de JSON a la entidad, en caso de recibir los datos desde el proveedor de datos.

Datos - Repositorio
La capa de repositorio es un envoltorio alrededor de uno o más proveedores de datos con los que se comunica la capa de bloque, es decir abstrae la lógica de los proveedores de datos y en caso de ser necesario la mantiene en memoria, para evitar realizar más peticiones de las necesarias.

Lógica de negocio (Bloc)
La responsabilidad de la capa Bloc (Business logic), es responder a la interacción del usuario con la capa de presentación, entregando diferentes estados dependiendo de los eventos enviados desde la capa de presentación.

Gestor de estados En Flutter existen vistas con y sin estado, utilizar un gestor es beneficioso[30], ya que, permite utilizar vistas sin estado que consuman el estado del gestor, quitando la necesidad de usar vistas con estado y separando la lógica de las vistas.

Para esta aplicación se utiliza Cubit[31] como gestor de estados, este gestor permite llamar funciones desde la capa de presentación que desencadenan diferentes estados que pueden ser escuchados por esta.

Presentación
La responsabilidad de la capa de presentación es mostrar información y funcionalidad al usuario, según el estado actual y permitir enviar eventos a un Bloc por medio de la interacción con esta.

5.9.2. Módulos de Frontend

El Frontend permite al usuario interactuar y obtener feedback del sistema de forma agradable e intuitiva. Tiene la tarea de mostrar interfaces a partir de la información que consume del Backend e interpretar las interacciones

del usuario, que serán transformadas en peticiones REST hacia el Backend y obtener feedback, que luego le mostrará al usuario.

Para el desarrollo del Frontend se contemplan 4 módulos MF1, MF2, MF3 y MF4, con sus respectivas secciones.

- **MF1 Usuario**
  - **Sección Autenticación:** Permite al usuario, registrarse e iniciar sesión a la aplicación, ya sea como usuario registrado o invitado. Al iniciar sesión se utiliza la base de datos local, para guardar el token, si está registrado, o el nombre, en caso de ser invitado y así mantener la sesión disponible cuando este quiera ingresar otra vez a la aplicación.
  - **Sección Voluntario:** Esta sección permite al usuario solicitar ser voluntario y activar/desactivar su modo voluntario, para elegir si recibir notificaciones sobre alertas cercanas. Una vez el usuario es voluntario se le permite gestionar alertas dentro de la aplicación y crear o pertenecer a una comunidad.
- **MF2 Alertas**
  - **Sección Mapa:** Sección encargada de mostrar todas las alertas en tiempo real en un mapa e interactuar con ellas, de modo que un voluntario pueda hacerse cargo de una.
  - **Sección Alerta:** Permite a un usuario crear una alerta, editarla y eliminarla.
- **MF3 Organización y Fundaciones:** Este módulo permitirá a usuarios voluntarios crear o unirse a una fundación/organización y actuar en su nombre por medio de un perfil informativo que administrarán todos los voluntarios pertenecientes a esta.
- **MF4 Publicaciones**
  - **Sección Informativa:** Cuenta con categorías y cada categoría con sus respectivas publicaciones.
  - **Publicar:** Permitirá a voluntarios pertenecientes a una fundación crear una publicación utilizando formato markdown.

5.10. Matriz de requisitos funcionales y módulos

5.10.1. Matriz del *Backend*

	RF1	RF2	RF3	RF4	RF5	RF6	RF7	RF8	RF9	RF10
MB1	X	X	X	X	X	X			X	X
MB2					X	X		X		
MB3									X	X
MB4							X	X		
MB5							X	X		

Tabla 8: Matriz de Requisitos Funcionales y Modelos de Backend

5.10.2. Matriz del *Frontend*

	RF1	RF2	RF3	RF4	RF5	RF6	RF7	RF8	RF9	RF10
MF1	X	X	X	X	X	X				
MF2					X				X	X
MF3					X	X		X		
MF4							X	X		

Tabla 9: Matriz de Requisitos Funcionales y Modelos de Frontend

6. Verificación y Validación del Prototipo

6.1. Validación de la Interfaz

Como equipo, se realizaron dos iteraciones de testing 9.3 con distintos tipos de usuarios para evaluar la efectividad del diseño y la navegación de la interfaz, estas fueron guiadas por la estudiante de Diseño de Productos con participación del perfil técnico. Estas iteraciones se efectuaron con 7 y 5 usuarios respectivamente, las reuniones fueron de carácter individual con cada uno a través de la plataforma Zoom y también algunas se llevaron a cabo en el Laboratorio de Experiencia Usuaria (LabUX) ubicado en las dependencias de la Universidad Técnica Federico Santa María.

Las preguntas y tareas planteadas para los usuarios que son voluntarios de fundaciones fueron principalmente sobre la usabilidad y el funcionamiento de los procesos de creación y toma de alertas, para saber si estos son eficaces y eficientes para el contexto de uso, y así también conocer la perspectiva que pueden entregar los usuarios en base a su experiencia en el proceso de rescate de los animales. Por otro lado, las validaciones con usuarios comunes, al ser inexpertos en el ámbito del rescate animal, el testing tuvo un enfoque dirigido a la intuitividad de la interfaz y la percepción de la jerarquización de la información entregada.

De igual forma, se solicitaron las mismas tareas para ambos tipos de usuarios:

1. **Creación de una cuenta:** Se le solicita al usuario que debe crear una nueva cuenta para ingresar a la aplicación.
2. **Navegación libre:** Se le da al usuario dos minutos para realizar una navegación libre de la interfaz para conocer las diferentes secciones.
3. **Crear una alerta:** La tarea corresponde a la creación de una alerta dentro de un caso hipotético: *"Vas camino a tu casa y te encuentras con una caja con cachorros dentro, no te los puedes llevar a tu casa y no tienes auto como para llevarlos a algún lugar, por lo que decides emitir una alerta."*
4. **Volverse voluntario:** Se le solicita al usuario que se vuelva voluntario dentro de la plataforma.
5. **Tomar un caso:** La última tarea de la sesión será que el usuario tome un caso de alerta existente dentro de la aplicación.

Luego de llevar a cabo las tareas, se realizaron las siguientes preguntas:

1. ¿Cómo encuentras que es la navegación dentro de la aplicación? ¿Sientes que esta fue complicada o intuitiva?
2. ¿Crees que hubo alguna complicación para completar las tareas?
3. ¿Qué opinas acerca del diseño de la aplicación? ¿Hay alguna cosa que crees que se puede mejorar?

6.1.1. Resultados

Al concretar ambas iteraciones de validaciones se obtuvieron resultados positivos, al comprobar que se cuenta con una interfaz intuitiva, un diseño amigable para el usuario, además de contar con un prototipo funcional para el contexto de uso. Si bien ningún usuario tuvo complicaciones para resolver las tareas, de igual forma se lograron determinar correcciones para llegar a nuestro prototipo final, las cuales fueron aceptadas y validadas también por la fundación mandante. Para mas detalles sobre el proceso de validación e interacción con usuarios revisar memoria de Carla Parra Rebolledo.

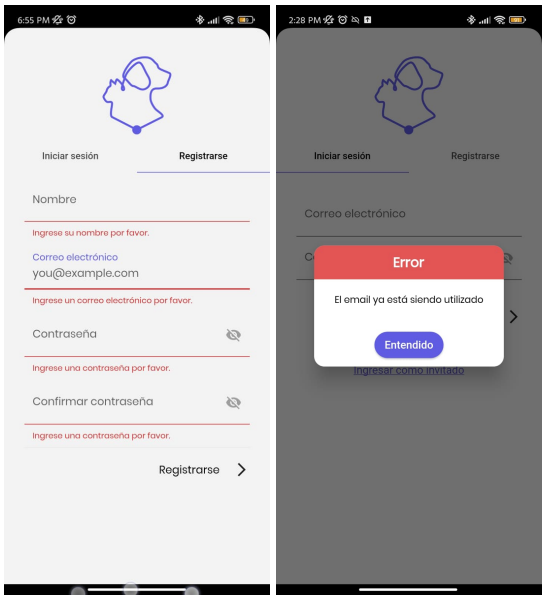
6.2. Verificación del Backend y Frontend

Para verificar la integridad de los datos, el correcto funcionamiento de la plataforma, servidor, integración de Backend con Frontend y servicios cloud implementados se procede a realizar pruebas de concepto y funcionalidad que aseguren:

- Inicio de sesión únicamente utilizando correspondientes credenciales para usuarios registrados.
- Control de autenticación efectivo para prevenir el riesgo de secuestro o abuso de una sesión de usuario, utilizando JWT.
- Protección de datos sensibles al aplicar una función Hash que encubre información como contraseñas.
- Coherencia de la información almacenada en la base de datos contra la que es visualizada en la aplicación móvil.
- Control de los tipos de datos ingresados en los distintos formularios en la aplicación, además de la verificación de campos no nulos.
- Alertas se vinculen correctamente con usuarios, voluntarios y categorías de alertas.
- Correcta visualización de alertas en el mapa, según la ubicación geográfica seleccionada.
- Correcta actualización al convertir usuarios en voluntarios y también la activación del modo voluntario, reflejando cambios en el menú lateral.
- Correcta vinculación entre voluntario, fundación y categoría de publicación al redactar una nota informativa.

6.2.1. Manejo de errores en la Plataforma

Para evitar funcionamientos inesperados en la aplicación se realizó un manejo de errores. Existen dos formas de manejarlos, una directamente en el Frontend y así evitar llamadas innecesarias al servidor si un formulario no cumple con sus restricciones fig.13(a) y otra consumiendo el error recibido desde el Backend y desplegando una respuesta en el cliente móvil , en este caso el error se le muestra al usuario a través de un dialogo fig.13(b).



(a) (b)

Figura 13: (a) Campos vacíos (b) Correo ya registrado

Todos los errores manejados, pueden revisarse en la sección de anexos 9.4.

6.3. Entorno de Trabajo

Durante el proceso de desarrollo de software, se genera un entorno de desarrollo local y otro de producción al que se accede remotamente. A continuación se describen las plataformas, servicios y tecnologías utilizadas.

Despliegue a Heroku
Heroku[9] es una plataforma de aplicaciones en la nube, la cual nos brinda servicios y herramientas para montar aplicativos. En este caso se utiliza para desplegar el servidor Backend en un ambiente de NodeJs y utilizar los servicios de PostgreSQL que la plataforma provee, de esta manera se tiene una API REST disponible para realizar consultas HTTP remotamente y almacenar información enviada desde múltiples clientes que utilizan la aplicación en sus dispositivos o tienen acceso a los endpoints del servidor. Es importante destacar que gracias a Heroku CLI[41], se tiene acceso a los registros del servidor, para verificar el estado y respuesta de las peticiones realizadas fig.14.

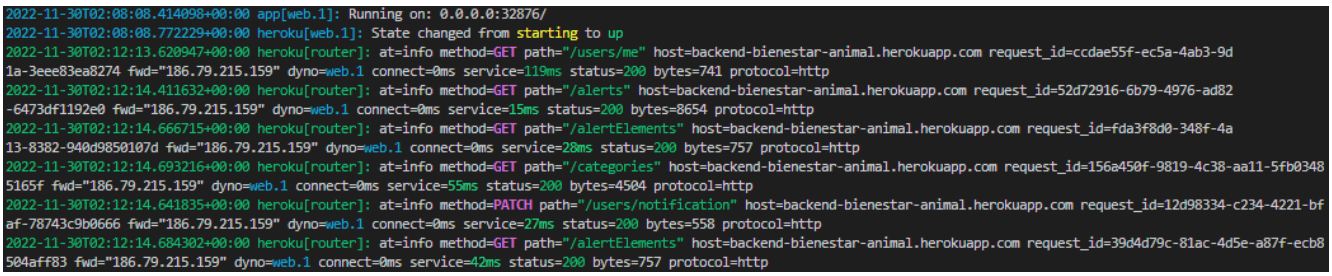


Figura 14: Heroku Command-line Interface

Pruebas desde Postman
Antes de realizar pruebas funcionales desde los clientes móviles o web provistos por el Frontend, se utiliza la plataforma API de Postman[42]. En esta se crea un espacio colaborativo con el fin de probar cada uno de los endpoints desarrollados, enviando peticiones HTTP para verificar correcto funcionamiento de tokens, atributos de la base de datos y respuestas del Backend entre otros.

Gestión de Base de Datos
Para visualizar y manipular los datos almacenados en la Base de datos se utiliza el ORM Prisma, particularmente su herramienta Prisma Studio fig.15, que a través de una interfaz en el buscador web facilita el contraste entre los datos esperados y obtenidos, para la toma de decisiones y verificación de que la información provista desde el Frontend es coherente con lo recepcionado por el Backend, así como también la manipulación de datos y tablas, evitando el uso de gestores de bases de datos.

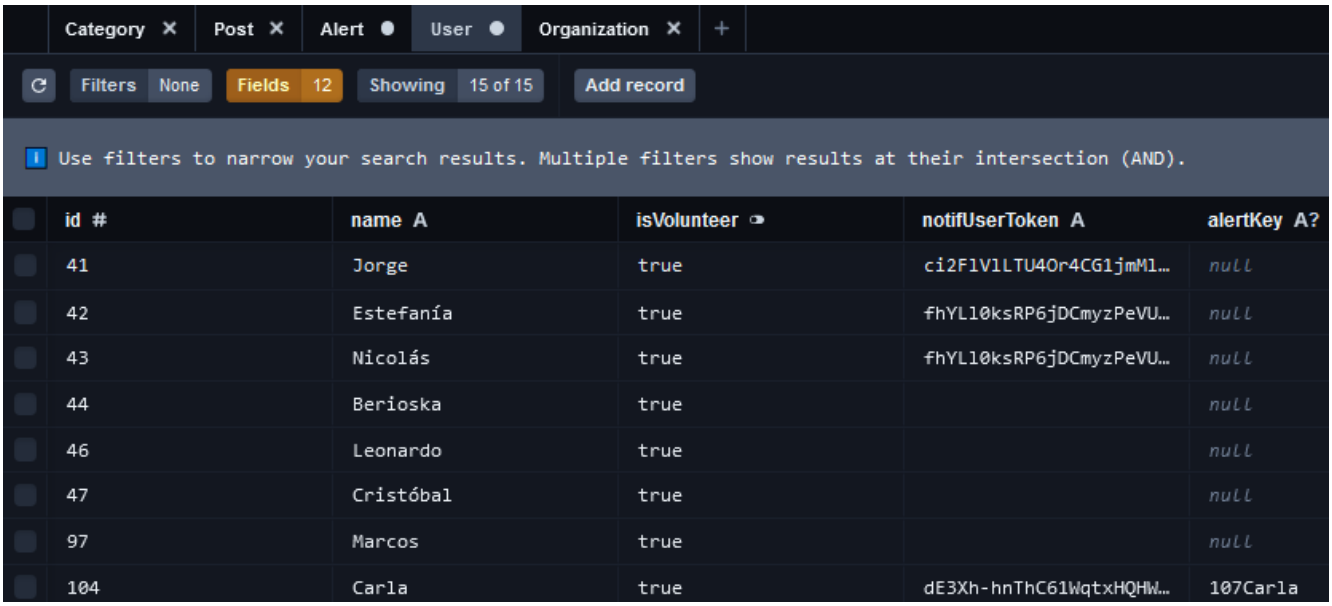


Figura 15: Interfaz usuaria de Prisma Studio

Generación de APK de prueba
Para verificar el funcionamiento de la aplicación en modo producción, se genera un APK para distribuirla entre los miembros del equipo y realizar pruebas funcionales

Se visualiza en la fig.16 el comando para generar un APK, se notan avisos en rojo que indican advertencias en paquetes de terceros, utilizados en el desarrollo.

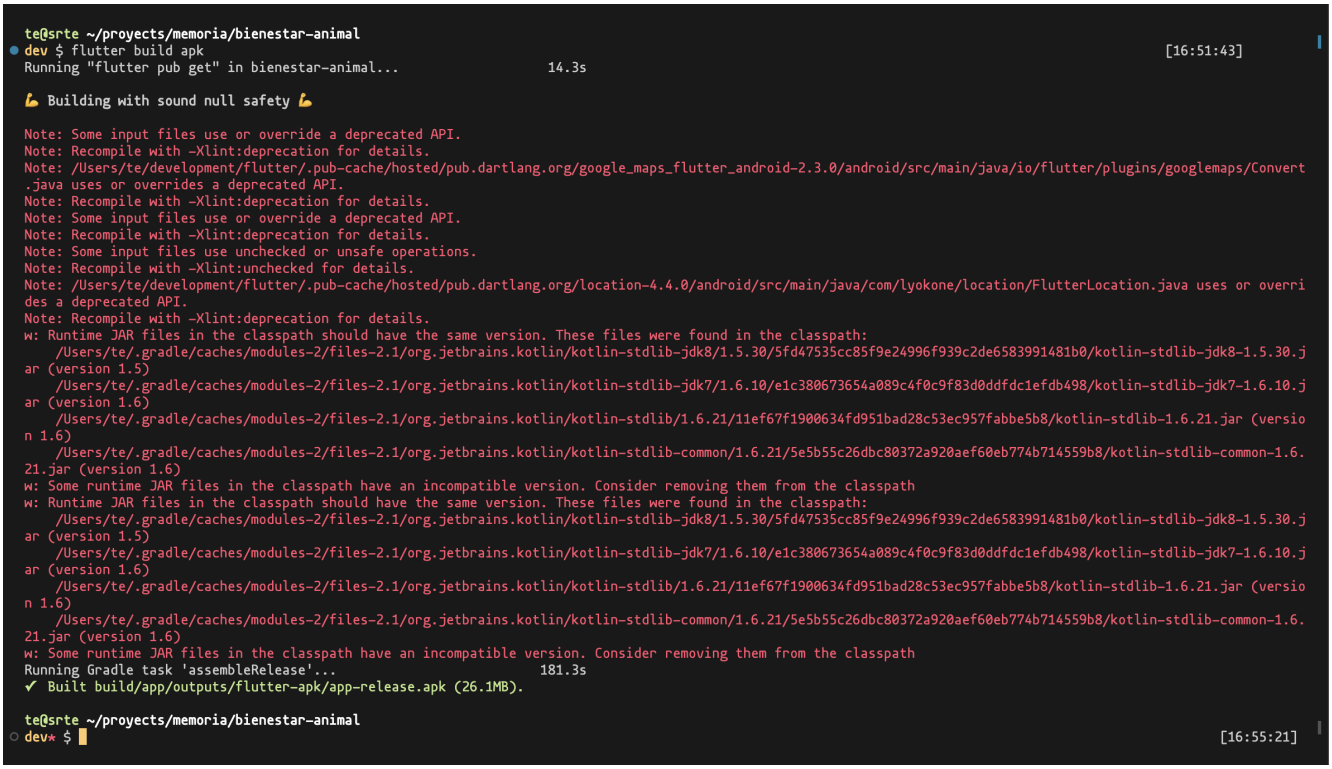


Figura 16: Generación de APK

6.4. Análisis Crítico y Evaluación de Riesgos

Luego de recibir retroalimentación por parte de distintos tipos de usuarios, Stakeholders y realizar pruebas funcionales y de concepto al prototipo en desarrollo, se logró percibir riesgos que podrían afectar el correcto uso de la aplicación como dañar la integridad de esta.

- Se reconoce la posibilidad de que usuarios puedan crear alertas falsas con fines maliciosos.
- Posibles alertas creadas, podrían ser hechas con fines de señalar a un agresor y no con la intención de velar por el bien de un animal.
- Casos peligrosos que podrían tener desenlaces violentos hacia voluntarios de la aplicación.

Solo algunas de las consideraciones mencionadas son reflejadas en propuestas que apacigüen estas problemáticas, esto debido a que el tiempo del que se dispone en el programa es acotado y que soluciones propuestas por los mismos usuarios requieren de participación de terceros como fuerzas policiales. De esta manera las consideraciones serán propuestas como trabajo a futuro para la construcción de la aplicación en su totalidad, velando por la seguridad que los usuarios requieren.

7. Resultados

En esta sección se presentan los resultados del producto mínimo viable desarrollado, el cual ha sido ejecutado en un entorno de pre-producción o staging, siendo éste desplegado en un servicio cloud de Heroku, recibiendo consultas HTTP desde la plataforma API Postman junto a pruebas funcionales realizadas desde múltiples dispositivos móviles físicos como se explico en el capítulo anterior. Se destaca que este prototipo está siendo construido siguiendo la priorización de funcionalidades acordadas con la fundación Bienestar Animal, las cuales fueron obtenidas de los requisitos funcionales propuestos con anterioridad.

7.1. Modelo de navegación

En la fig.17 se muestra el modelo de navegación implementado en la aplicación, el usuario inicia en la pantalla de Login y una vez ingresadas las credenciales correctamente se le lleva a la pantalla de home, la cual muestra las novedades y categorías de publicaciones.

En este modelo se observan flechas de colores diferentes, **negro**, cuando la navegación es exitosa y **morado**, en caso de que la navegación tenga una restricción y el usuario deba ser redirigido a otra vista. Por ejemplo, si un usuario desea ir a la pantalla para solicitar ser voluntario y este no está autenticado, la aplicación lo va a redirigir a la pantalla de Login.

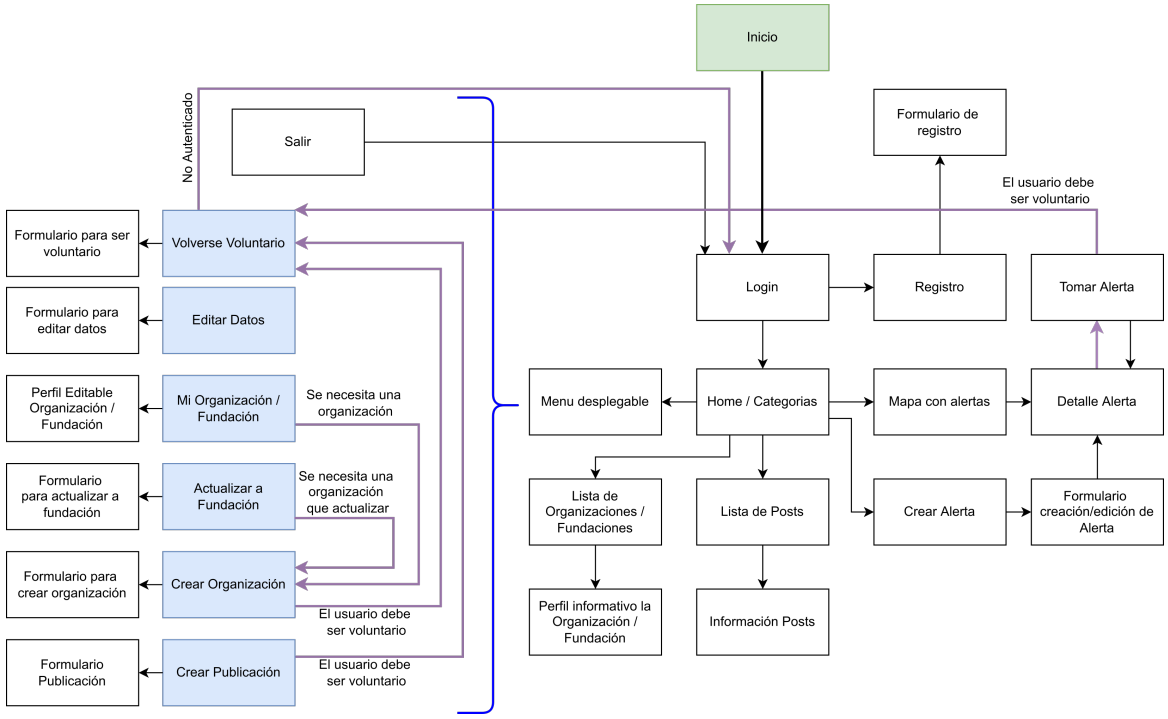
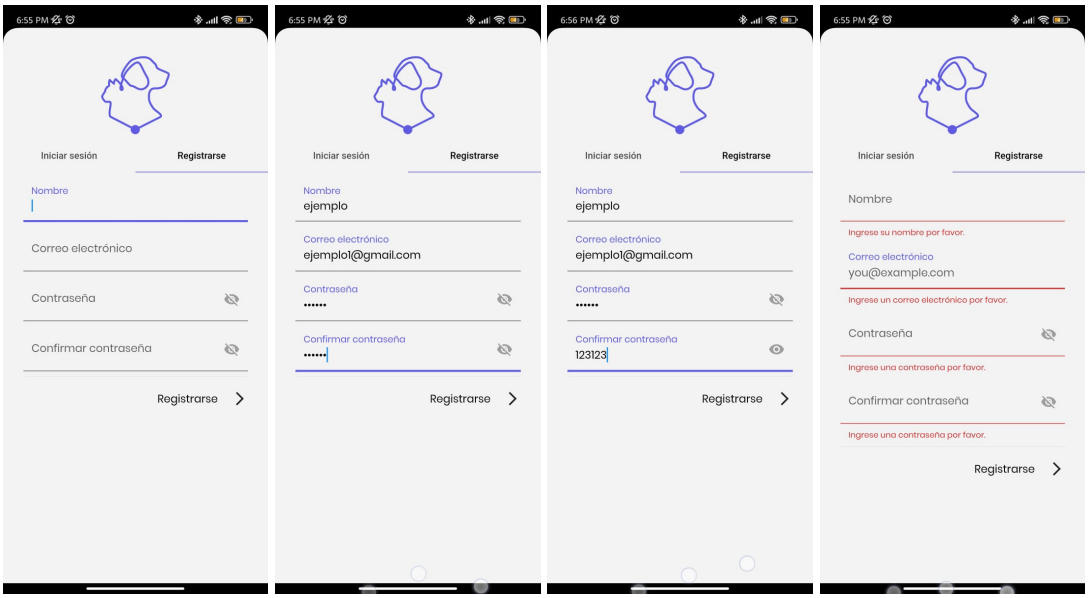


Figura 17: Modelo de navegación

7.2. Interfaces implementadas

Siguiendo la priorización planificada, se presenta a continuación el estado actual del prototipo funcional y las interfaces implementadas:

- **Registro de Usuarios:** Formulario con datos del usuario. Para el registro todos los datos pedidos son necesarios como se puede ver en la fig.18, por lo que al intentar registrarse se indicará al usuario si falta uno de los datos fig.18(d). Además como muestra la fig.18(c) el usuario puede revelar la contraseña que escribió.

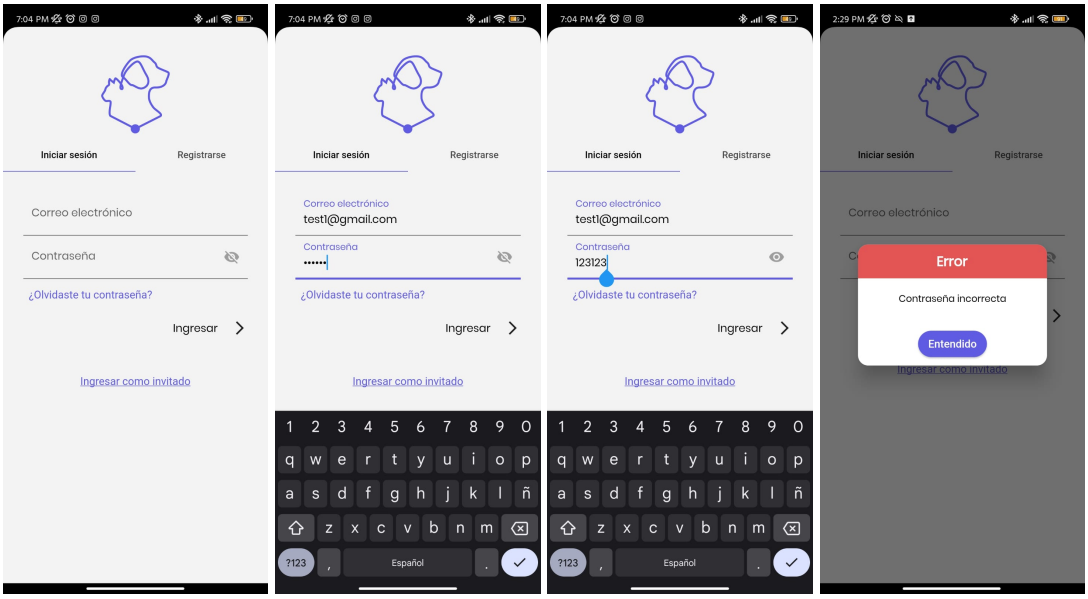


(a) (b) (c) (d)

Figura 18: (a) Formulario inicial (b) Formulario completado (c) Revelar contraseña (d) Error

- **Ingreso de Usuarios Registrados:** Un usuario registrado podrá entrar a la aplicación ingresando su correo y contraseña fig.19(a)(b), además para mayor comodidad, se implementó una funcionalidad para poder ver la contraseña fig.19(c), ya que esta normalmente se oculta por ser un dato sensible.

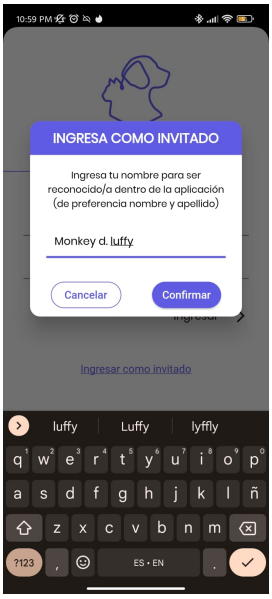
  Si el usuario intenta ingresar habiendo puesto mal su contraseña, el servidor responde con su error respectivo fig.19(d)



(a) (b) (c) (d)

Figura 19: (a) Formulario inicial (b) Formulario completado (c) Revelar contraseña (d) Error: Contraseña incorrecta

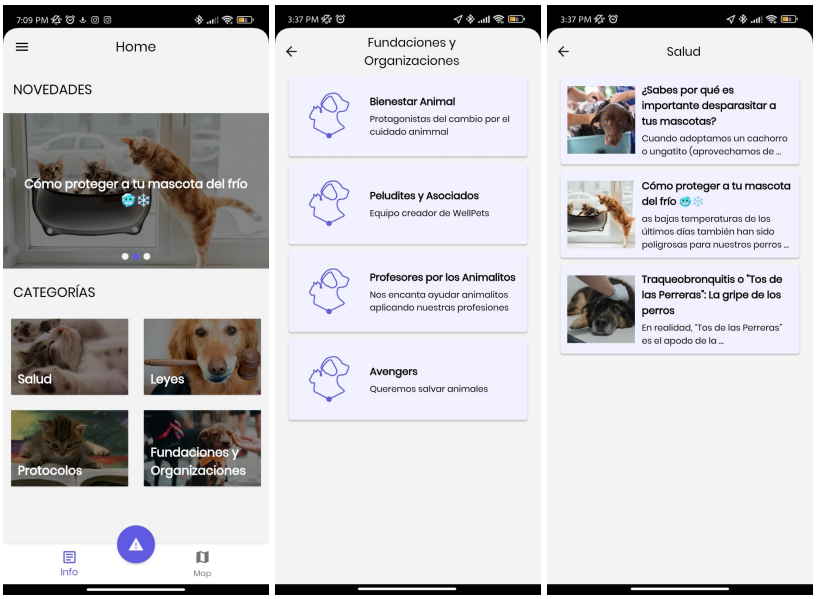
- **Ingreso de Invitados:** Un usuario que no desea registrarse, puede ingresar a la aplicación como invitado, indicando un nombre como muestra la fig.20, una vez dentro de la aplicación, éste puede crear alertas y acceder al contenido de la aplicación, teniendo como restricción el no poder solicitar ser voluntario.



(a)

Figura 20: Ingreso invitado

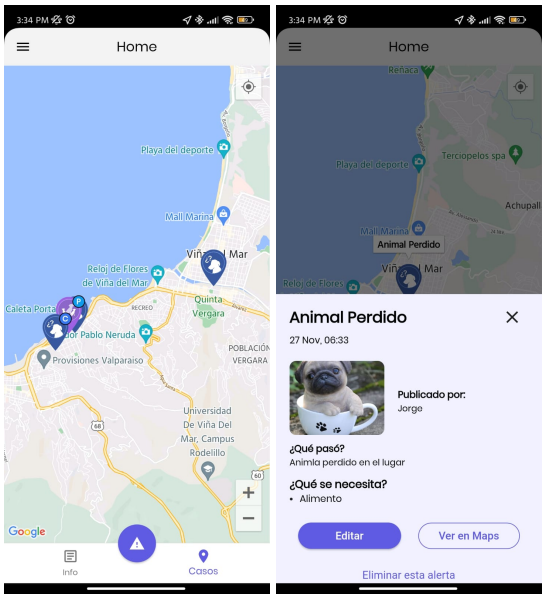
- **Sección Informativa:** Consta de una sección de novedades, que muestra las tres publicaciones más recientes y una sección de categorías fig.21(a). Al ingresar a una categoría se muestran todas las publicaciones asociadas a esta fig.21(c), además existe una categoría especial que contiene todas las organizaciones y fundaciones existentes en la aplicación fig.21(b), para que así los usuarios puedan buscar la información de una fundación en especifico.



(a) (b) (c)

Figura 21: (a) Novedades y categorías (b) Sección fundaciones (c) Sección salud

- **Mapa:** Vista en la que se visualizan las alertas creadas por los usuarios fig.22(a), al seleccionar una alerta se abre un panel que muestra su detalle y permite realizar diferentes acciones, como se muestra en fig.22(b).



(a) (b)

Figura 22: (a) Mapa (b) Detalle alerta

- **Creación de alerta:** Un usuario sólo puede mantener activa una alerta a la vez, por lo que sólo puede acceder al menú de creación si no tiene ninguna, en caso de tener una alerta creada se le redirige a la vista fig.23(g) que muestra todos los detalles de esta.

  Para crear una alerta, el usuario debe elegir categoría fig.23(a), escribir su detalle fig.23(b) y la ubicación a desplegar fig.23(f), también tiene la opción de subir una foto desde su galería fig.23(d).

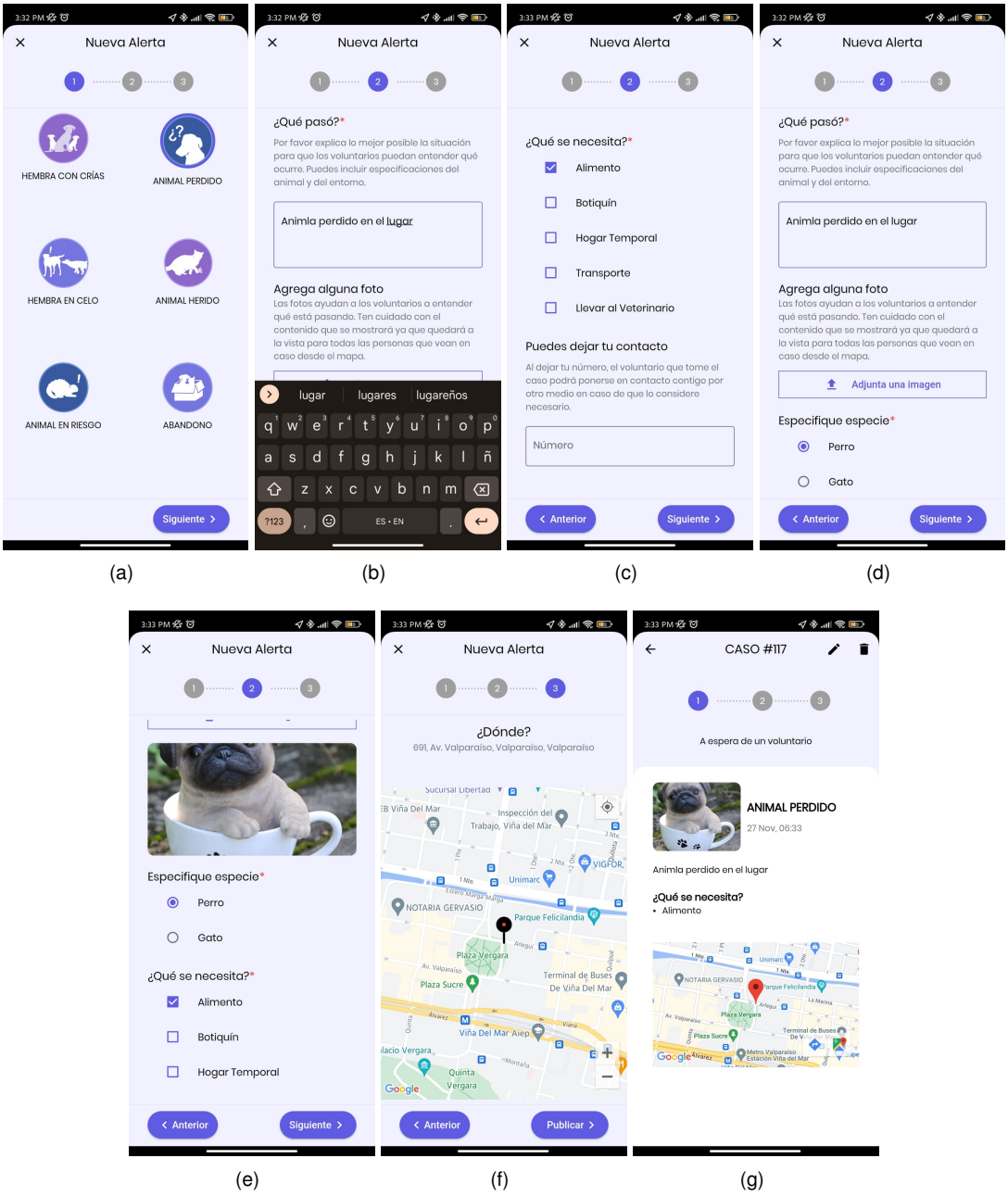
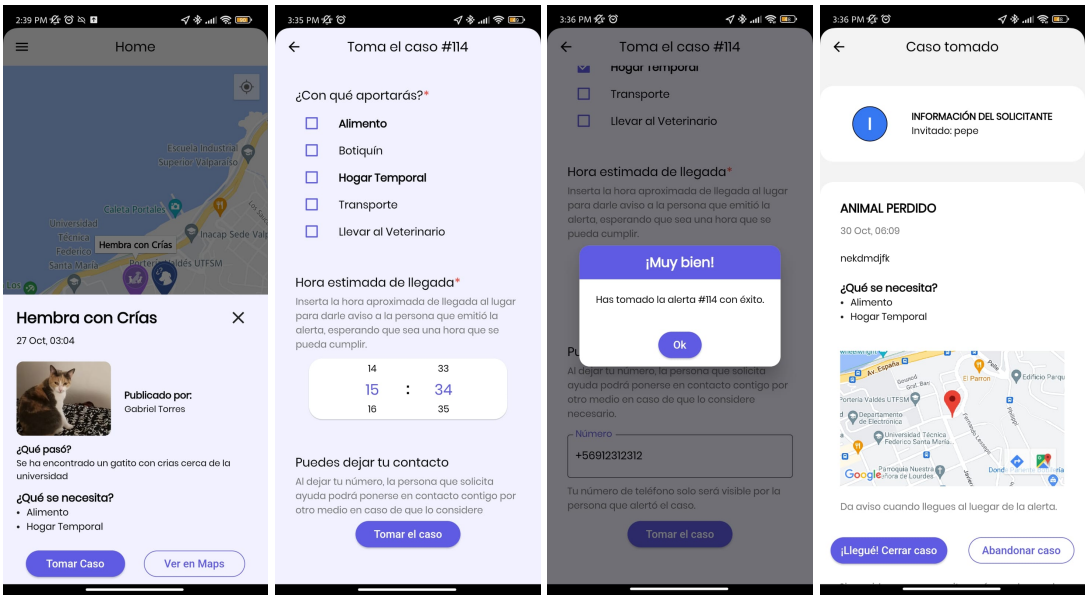
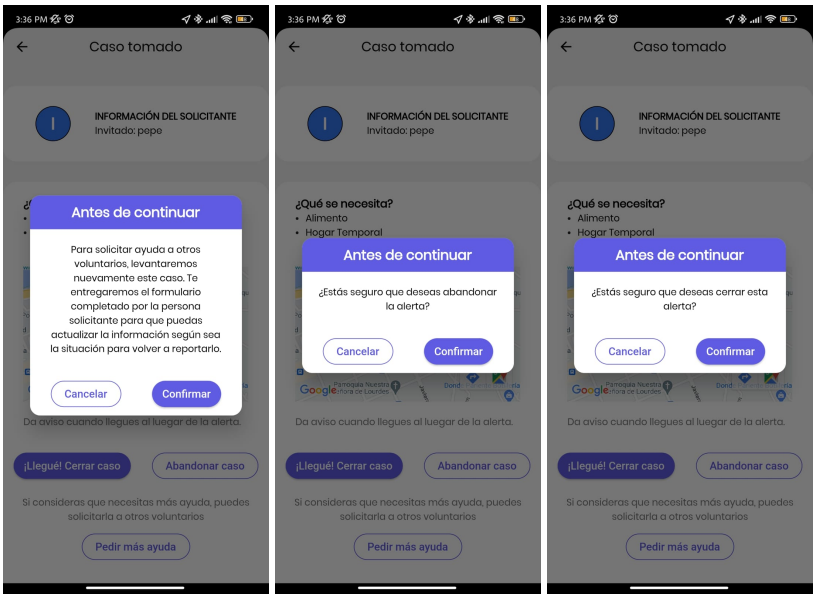


Figura 23: (a) Categorías (b) ¿Qué pasó? (c) ¿Qué necesita? (d) Sin foto (e) Foto subida (f) Ubicación (g) Creada

- **Tomar alerta:** Un voluntario al entrar al mapa con los casos y seleccionar uno de estos, tiene ahora la capacidad de tomarlo fig.24(a). Posteriormente deberá completar un formulario indicando lo que aportará, hora de llegada, su contacto fig.24(b) y finalmente al confirmar se visualiza una ventana de éxito fig.24(c). A continuación el voluntario es direccionado a otra vista fig.24(d) donde tendrá la opción de pedir más ayuda a otros voluntarios fig.24(e), abandonar el caso fig.24(f) y cerrar el caso al llegar al lugar fig.24(g).



(a) (b) (c) (d)

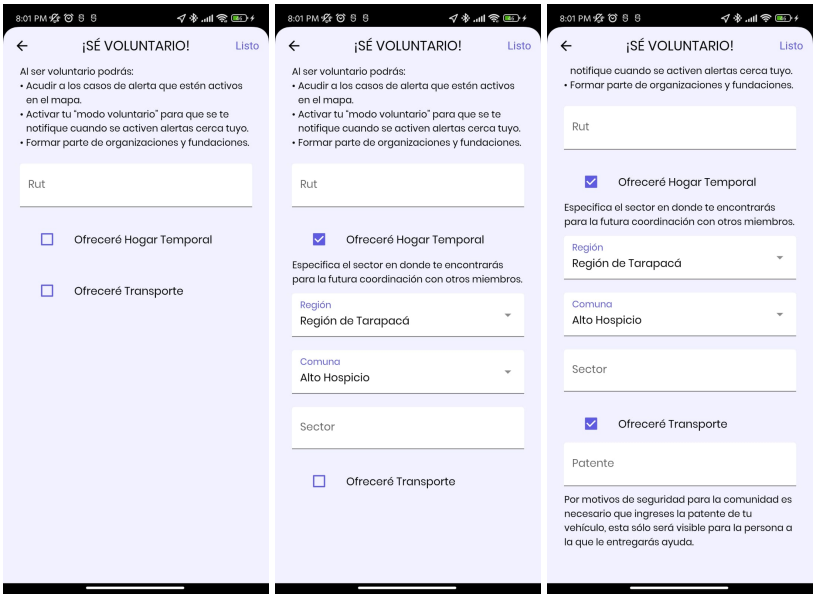


(e) (f) (g)

Figura 24: (a) Ver caso (b) Completar formulario (c) Caso tomado (d) Vista con 3 opciones (e) Confirmación de pedir mas ayuda (f) Confirmación abandonar caso (g) Confirmar cerrar caso

- **Quiero ser voluntario:** Un usuario puede ser voluntario solamente si está registrado, por lo que si el usuario ingreso como invitado, se le advertirá que debe ingresar con sus credenciales.

  En primera instancia, para ser voluntario se requiere solamente el RUT fig.25(a), pero si ofrece transporte deberá indicar la patente del auto fig.25(c) y en el caso de ofrecer hogar temporal, debe indicar el sector de la vivienda, pero no la dirección fig.25(b).



(a) (b) (c)

Figura 25: (a) Formulario inicial (b) Ofrecer hogar temporal (c) Ofrecer transporte

- **Crear Organización:** Un usuario voluntario puede crear una organización, para esto necesita ingresar el nombre fig.26(s), incluir al menos 2 voluntarios fig.26(b) y agregar al menos una red social a esta fig.26(c), una vez creada con éxito, se mostrará un mensaje para darle la retroalimentación respectivo al usuario fig.26(e).

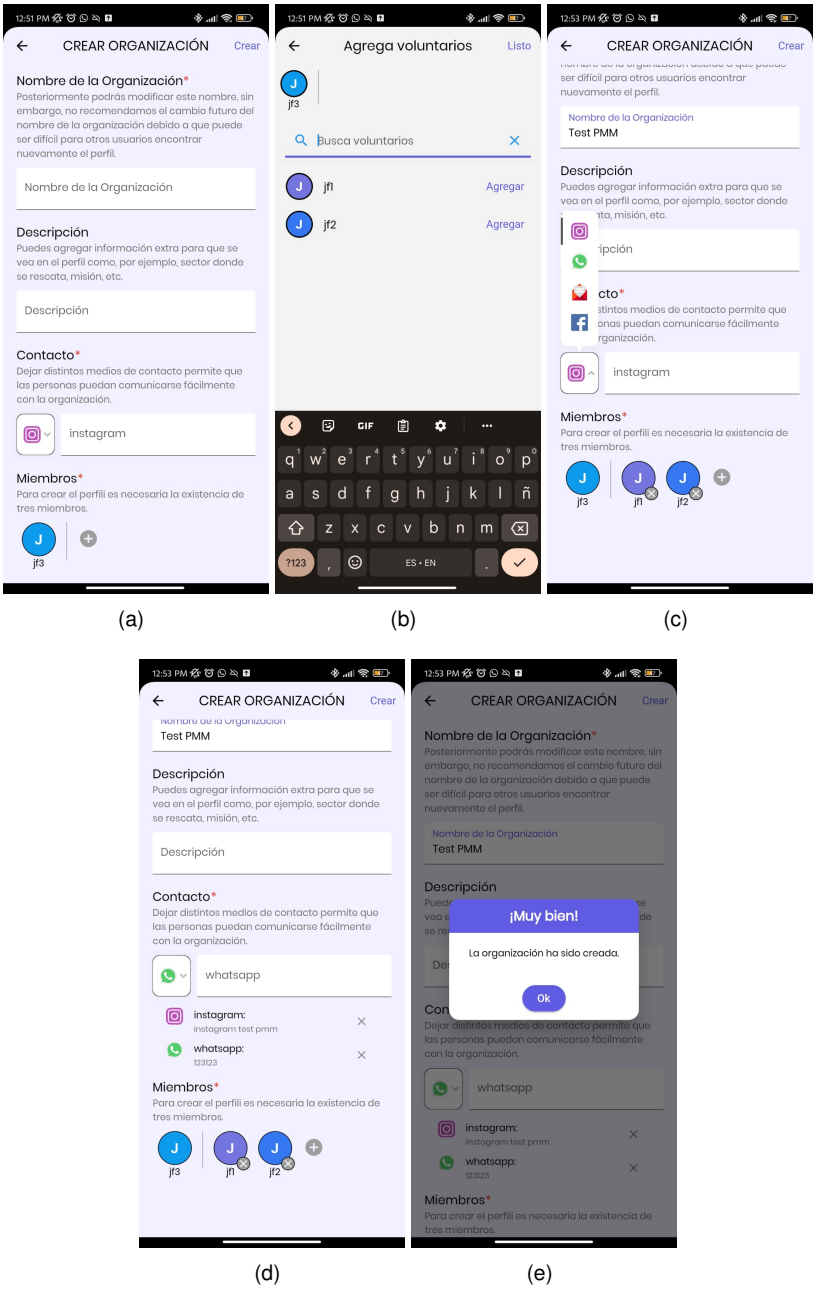
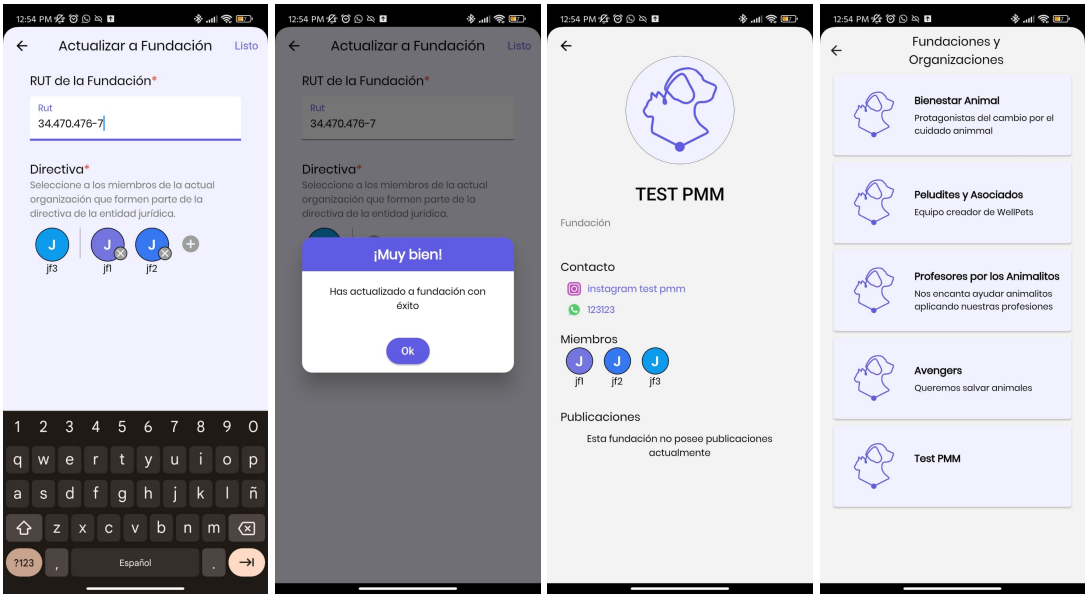


Figura 26: (a) Formulario inicial (b) Agregar voluntarios (c) Formulario con voluntarios (d) Redes sociales (d) Organización creada

- **Actualizar a Fundación:** Un usuario voluntario, perteneciente a una organización, puede actualizarla a fundación, esta debe estar constituida, por lo que se solicita ingresar el RUT de la persona jurídica de esta fig.27(a), una vez actualizada exitosamente esta aparecerá en la categoría especial de organizaciones y fundaciones y en su perfil se indicará que es una fundación fig.27(c)



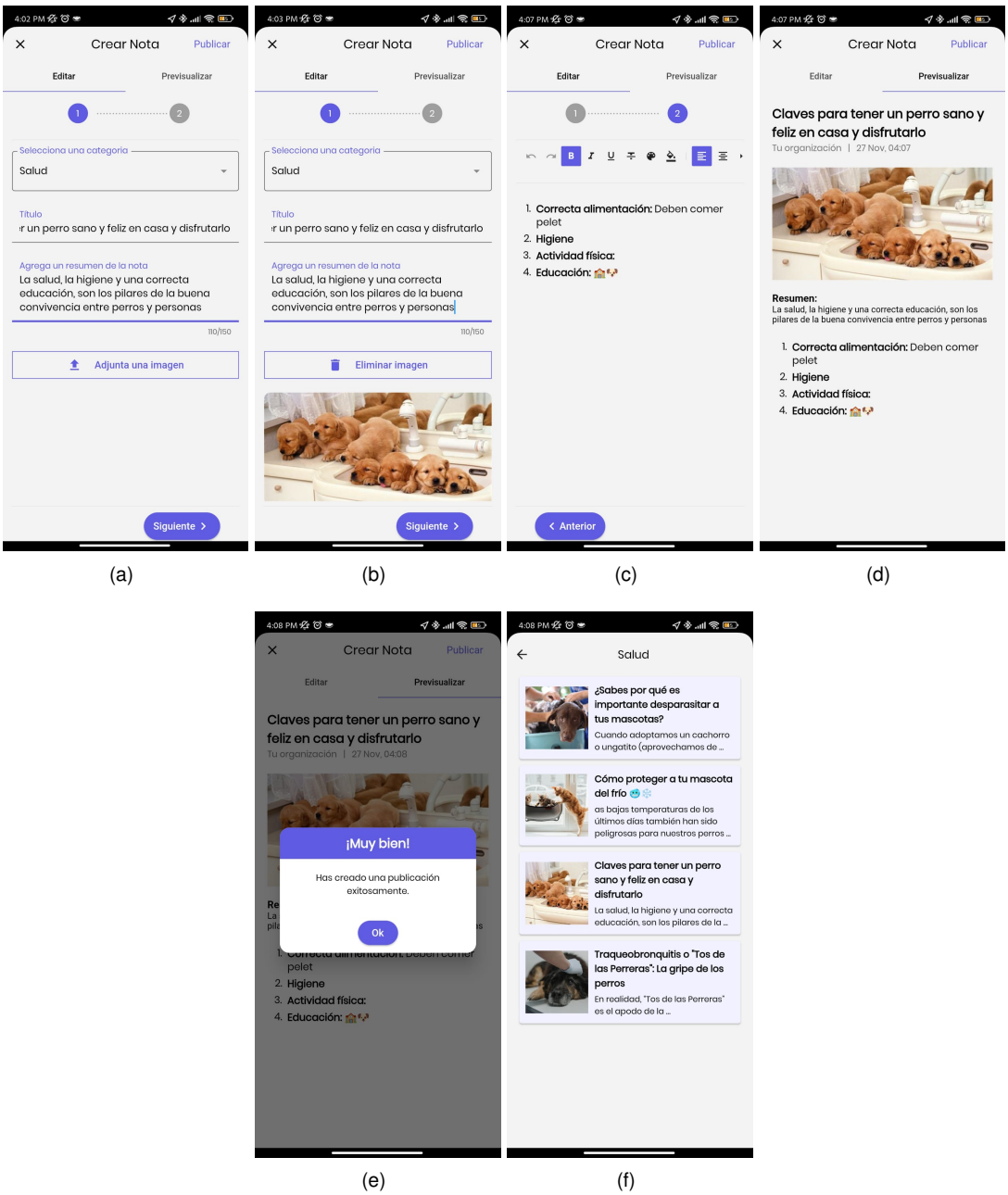
(a) (b) (c) (d)

Figura 27: (a) Solicitud de RUT (b) Actualizada con éxito (c) Perfil fundación (d) Categoría fundaciones y organizaciones

- **Crear publicación:** Un usuario voluntario, perteneciente a una fundación, puede crear publicaciones, para crear una se debe seleccionar la categoría, agregar un título, un resumen, un detalle fig.28(a) y tiene la opción de agregar una imagen fig.28(b).

  Durante este proceso el usuario puede previsualizar la publicación fig.28(d), para verificar como se verá una vez creada.

  Ya creada la publicación, ésta aparecerá en la categoría elegida fig.28(f) y todos los usuarios podrán acceder y leerla

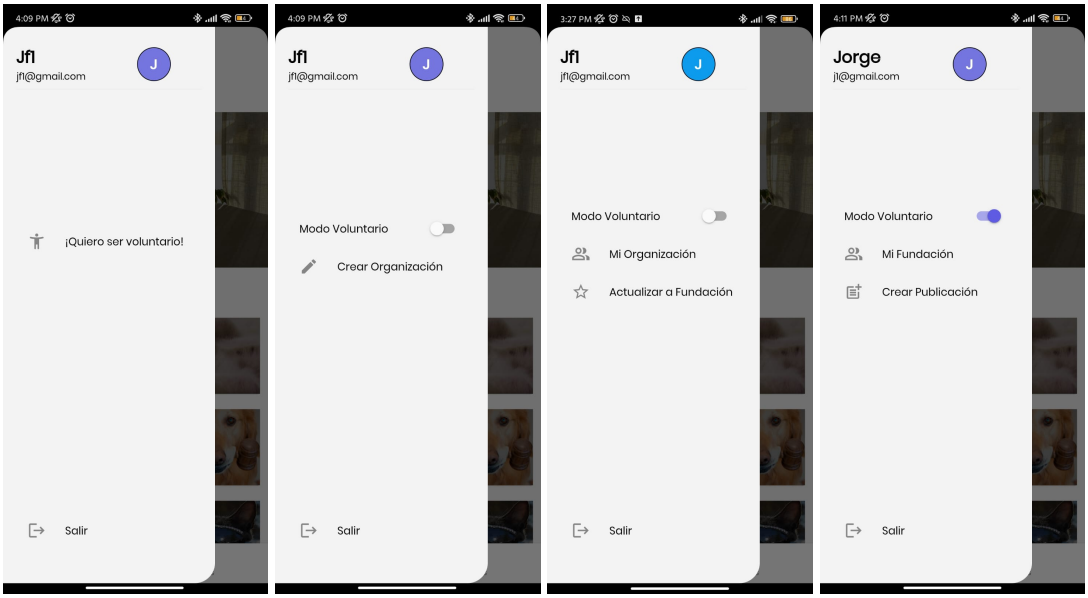


(a) (b) (c) (d)

(e) (f)

Figura 28: (a) Formulario inicial (b) Imagen agregada (c) Editor (d) Pre-visualización (e) Creada con éxito (f) Visible en su categoría

- **Menú lateral:** Según el tipo de usuario el menú lateral mostrará diferentes opciones, como se puede ver en la fig.29 todos los menú muestran la información del usuario en la parte superior y cuentan con un botón para cerrar sesión en la inferior.
  - **Menú usuario común:** tiene como única opción solicitar ser voluntario fig.29(a).
  - **Menú usuario voluntario:** permite activar y desactivar el modo voluntario, para elegir si aceptar que le lleguen notificaciones sobre alertas cercanas, además se da la opción de crear una organización fig.29(b).
    - **Menú voluntario con organización:** un voluntario con organización puede entrar al perfil de ésta y solicitar actualizarla a fundación fig.29(c).
    - **Menú voluntario con fundación:** un voluntario con fundación puede entrar al perfil de ésta y crear publicaciones fig.29(c).



(a) (b) (c) (d)

Figura 29: (a) Menú inicial (b) Menú voluntario (c) Menú organización (d) Menú fundación

8. Conclusión y Trabajos Futuros

La aplicación desarrollada alcanzó exitosamente el termino de Producto Mínimo Viable (MVP) durante el periodo de memorias, durante el cual se sometió a diversas verificaciones y pruebas que permitieron extraer conclusiones valiosas sobre su rendimiento y funcionalidad.

8.1. Conclusiones

Se demostró que la aplicación móvil cumplió satisfactoriamente con los requisitos planteados por la Fundación Bienestar Animal, proporcionando una herramienta efectiva para la comunicación e intercambio de información crucial en los problemas ligados al rescate animal entre usuarios y voluntarios.

Respecto a la escala de madurez de tecnologías en la que se encuentra el producto, este entra en la categoría de Modelo de sistema o demostración de prototipo en un entorno relevante (TRL6). Lo anterior es válido ya que la aplicación móvil desarrollada fue puesta a prueba de manera exitosa por múltiples usuarios y voluntarios en dispositivos móviles geolocalizados que realizan consultas e intercambio de mensajería utilizando servicios web e interfaces basadas en computación en la nube, demostrando que cumple con los requerimientos esenciales para su correcto funcionamiento.

Además de su funcionalidad, el diseño de la aplicación se sometió a múltiples evaluaciones y pruebas por distintos tipos de usuarios, con lo que se demostró su usabilidad y navegabilidad, constatando que la interfaz proporciona una experiencia de usuario intuitiva y fácilmente accesible.

No obstante, al reconocer estas fortalezas, es fundamental reconocer también sus principales limitaciones, las cuales están ligadas principalmente a poder discernir lo múltiples casos bordes de alertas falsas que podrían presentar un potencial riesgo para sus usuarios, un análisis crítico en este ámbito contribuirá a futuras mejoras, desarrollos y a la confianza de los usuarios en la plataforma.

Por otro lado, una fuerte amenaza como también un espacio a oportunidades es la capacidad de tener nuevos usuarios y retener estos dentro de la aplicación, para lo cual es necesario el desarrollo de funcionalidades que no fueron incluidas en el MVP, donde la principal limitante es el tiempo como recurso.

8.2. Trabajos futuros

Si bien la plataforma desarrollada cumple con los objetivos esperados por las distintas partes interesadas en el proyecto, esta aún tiene trabajo por delante para poder alcanzar la madurez de un producto en producción como tal. Bajo esto, se destacan tareas pendientes en distintas áreas, que aseguren una proyección exitosa.

8.2.1. Nuevas Secciones

Para dar mayor completitud, una mejor experiencia a los usuarios y poder retenerlos, se requiere también desarrollar funcionalidades y secciones que complementen lo existente, como se detalló previamente en el apartado de Definición de la Solución 6. Esto implica desarrollar:

- Un Foro interactivo para la comunidad que ya sea parte de la plataforma, el cual debe ser gestionado a su vez por miembros de fundaciones.
- Una sección que permita dar a conocer adopciones y operativos de estas, permitiendo a usuarios llenar un formulario para postular a una adopción
- Perfiles personalizables para los usuarios, ya que actualmente solo hay perfiles para las fundaciones y organizaciones.

- Funcionalidades que permitan mitigar cualquier intento de alerta falsa con fines maliciosos, como por ejemplo segmentando zonas geográficas del mapa donde se suelen reportar incidentes negativos para prevenir a los voluntarios que podrían acceder a estos sectores.
- Desarrollar la sección que permita implementar el modelo de negocios respectivo, para la sustentabilidad propia de la plataforma.

9. Anexos

9.1. Entrevista

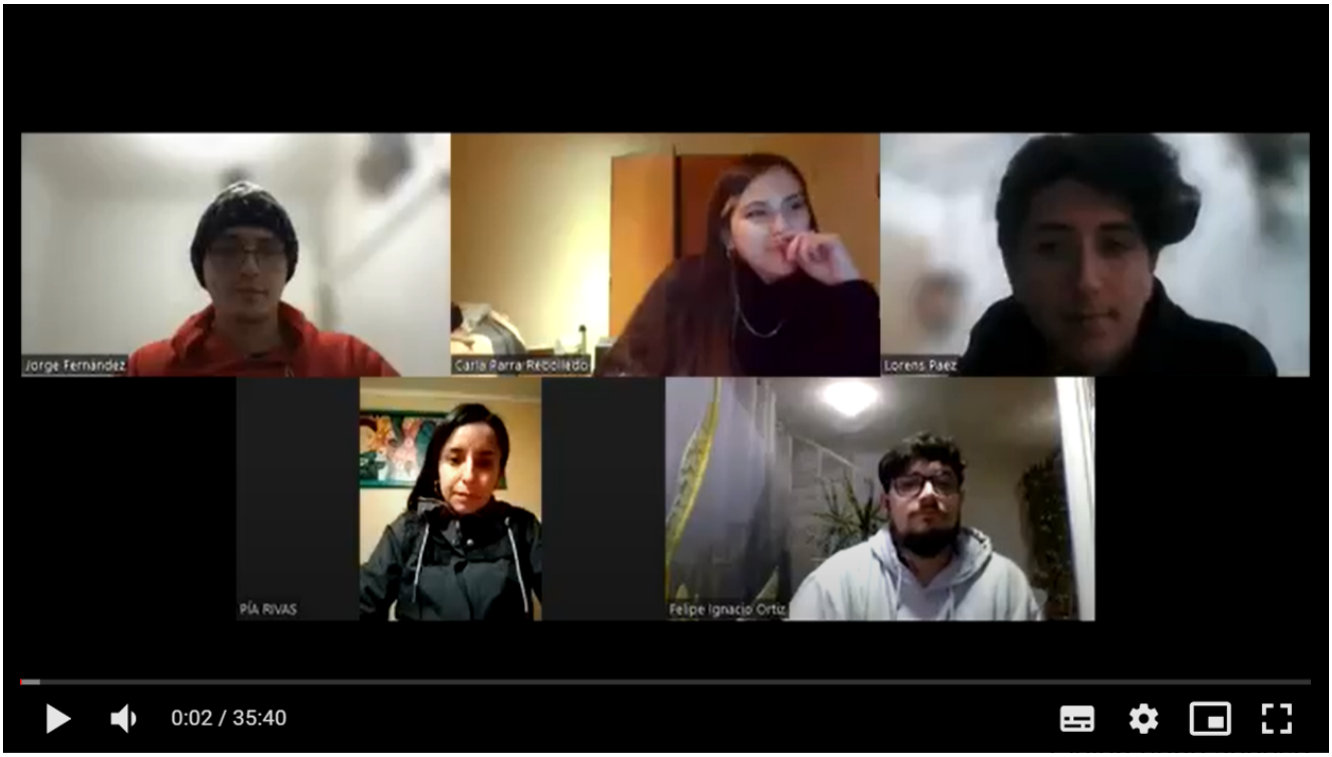


Figura 30: **LINK:** `https://drive.google.com/file/d/1FSxrNO_49Nk8ZKvAYqPrhFX_q4dorNfd/view?usp=sharing`

9.2. Encuesta

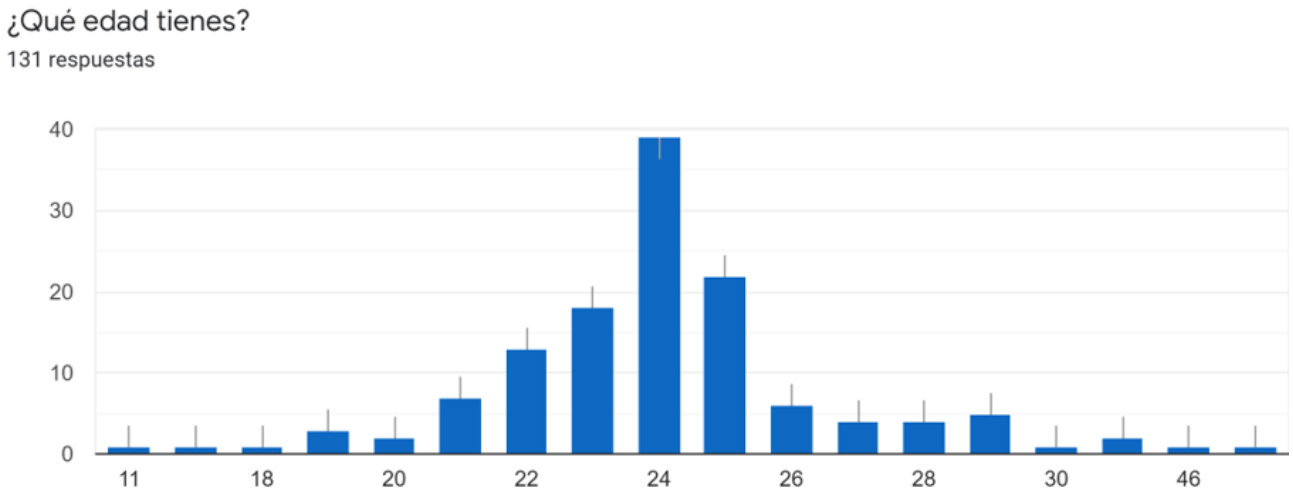


Figura 31: Fuente: Elaboración propia

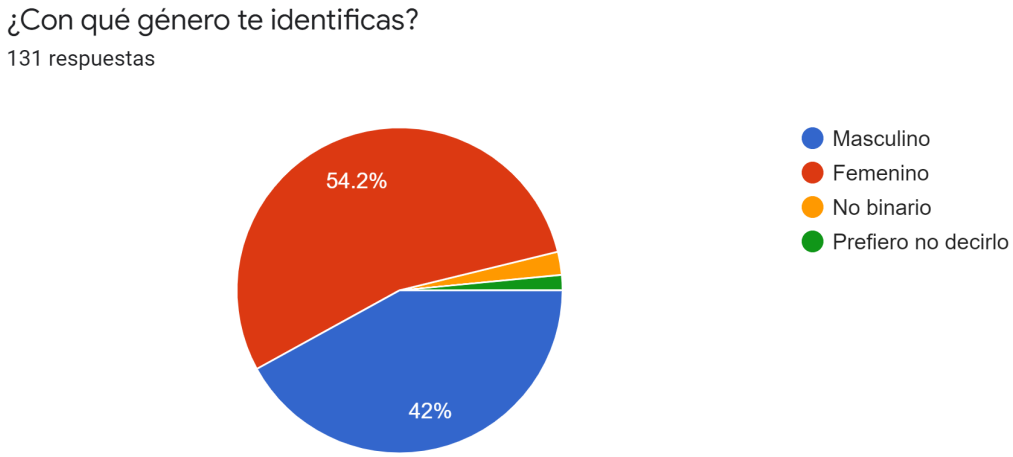


Figura 32: Fuente: Elaboración propia

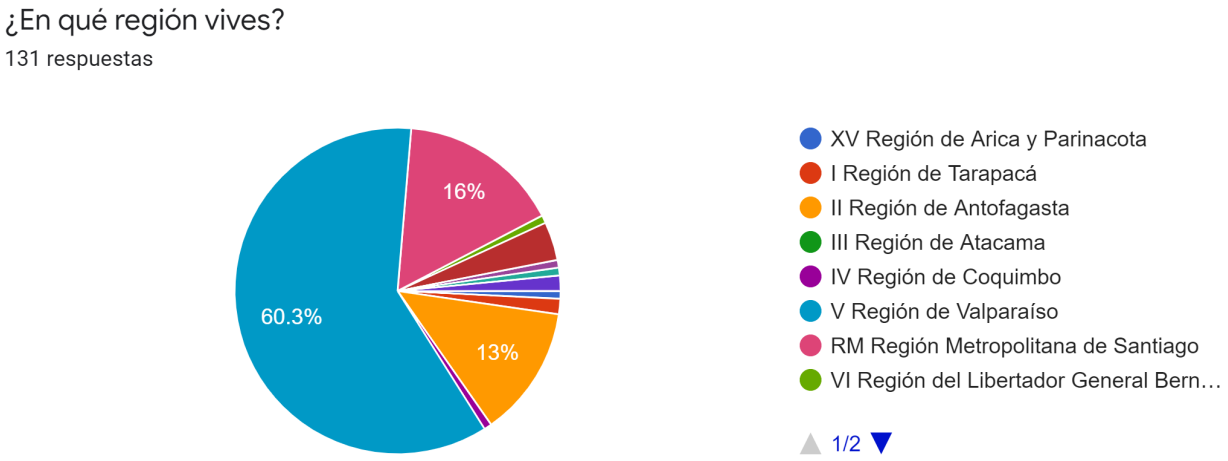


Figura 33: Fuente: Elaboración propia

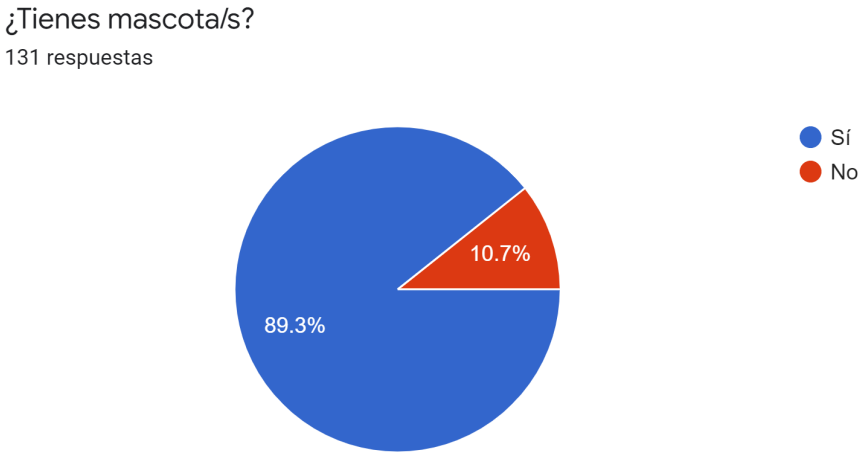


Figura 34: Fuente: Elaboración propia

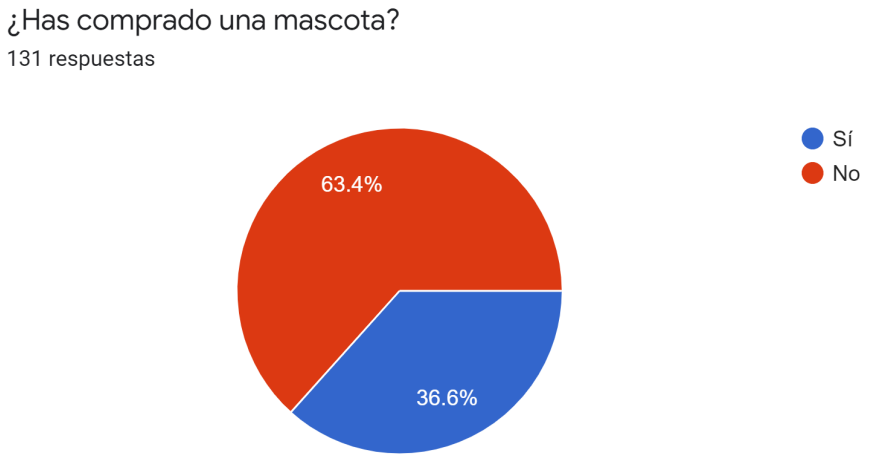


Figura 35: Fuente: Elaboración propia

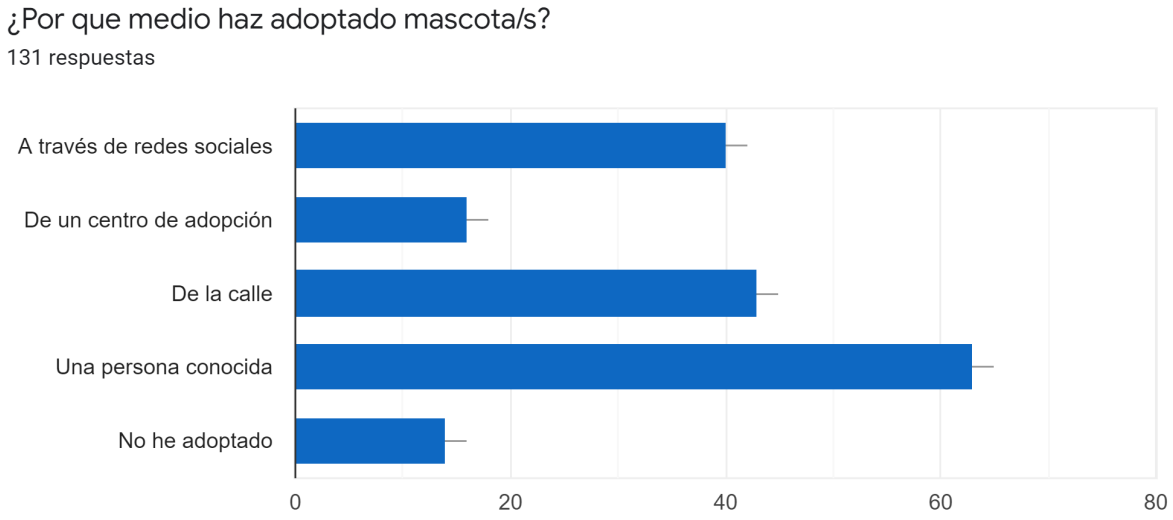


Figura 36: Fuente: Elaboración propia

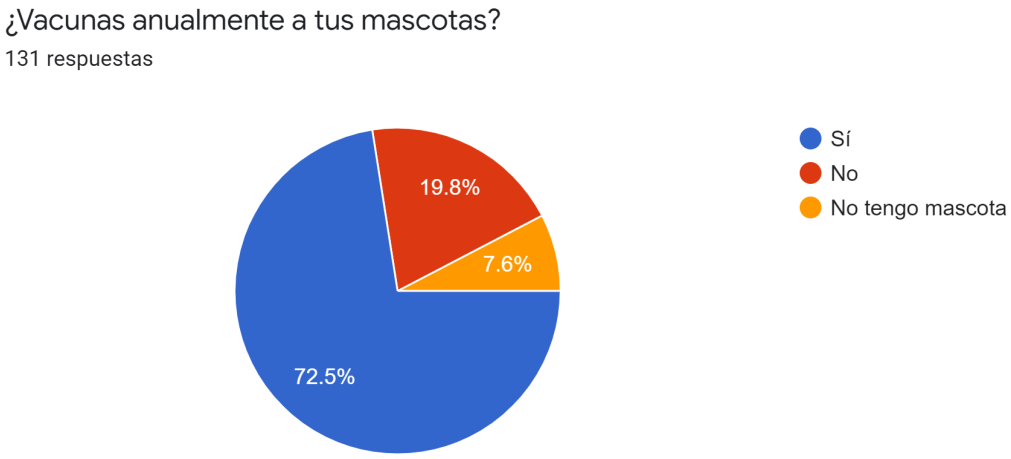


Figura 37: Fuente: Elaboración propia

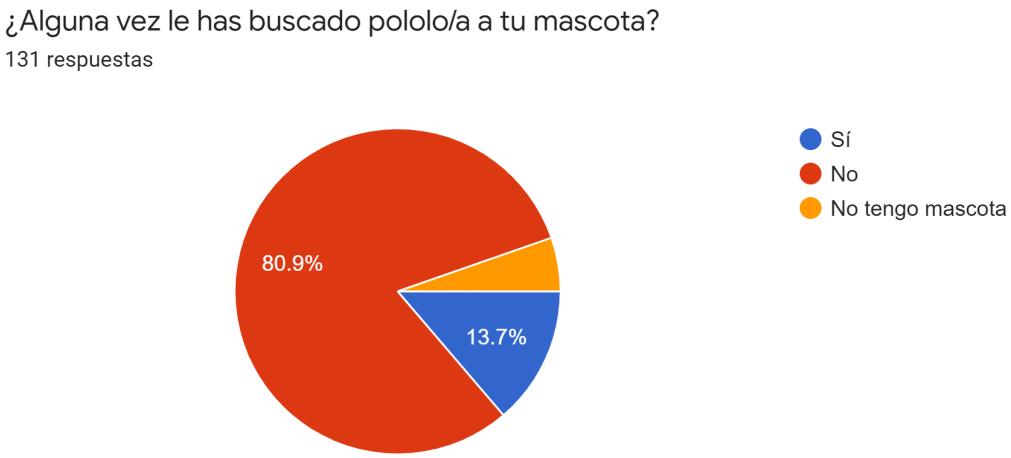


Figura 38: Fuente: Elaboración propia

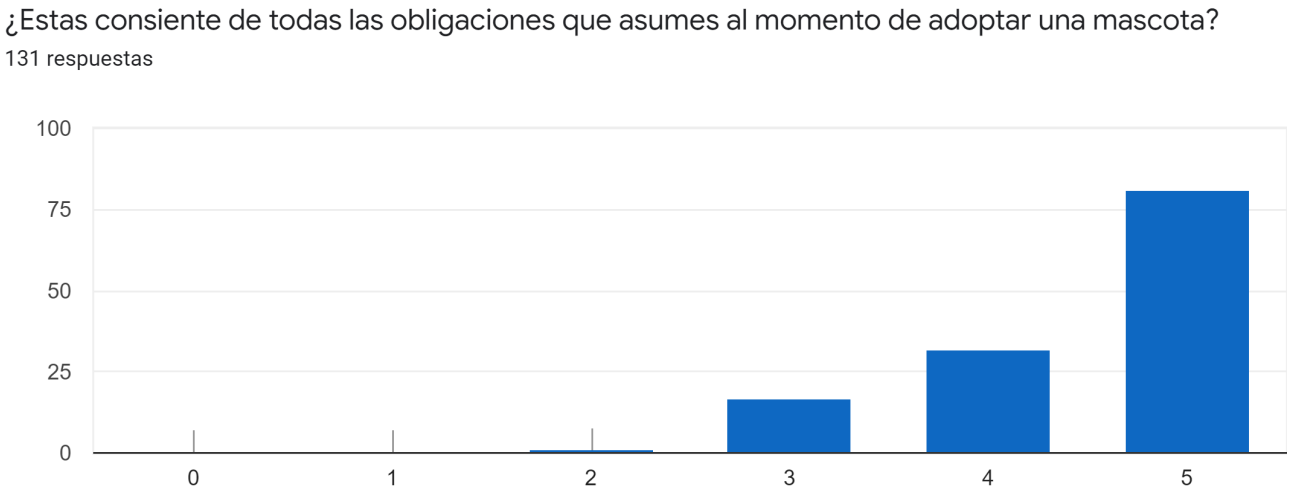


Figura 39: Fuente: Elaboración propia

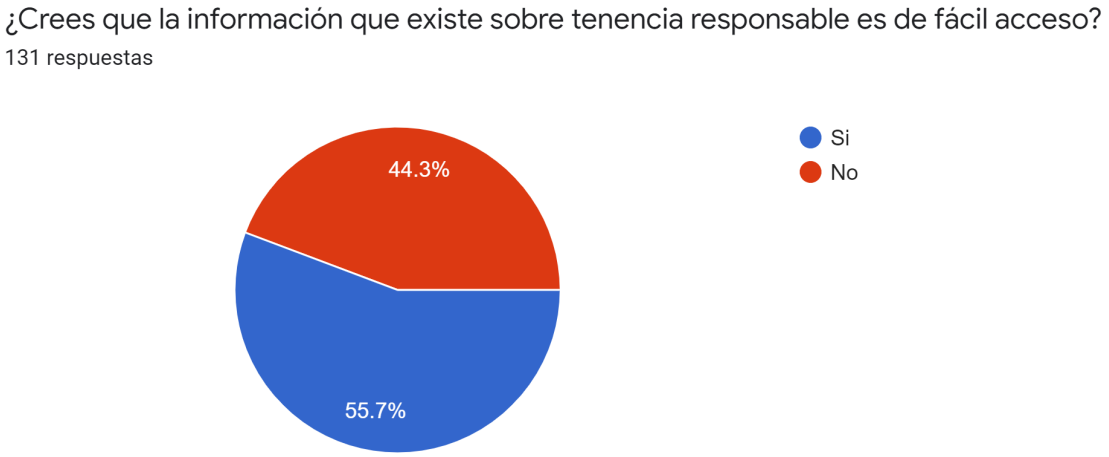


Figura 40: Fuente: Elaboración propia

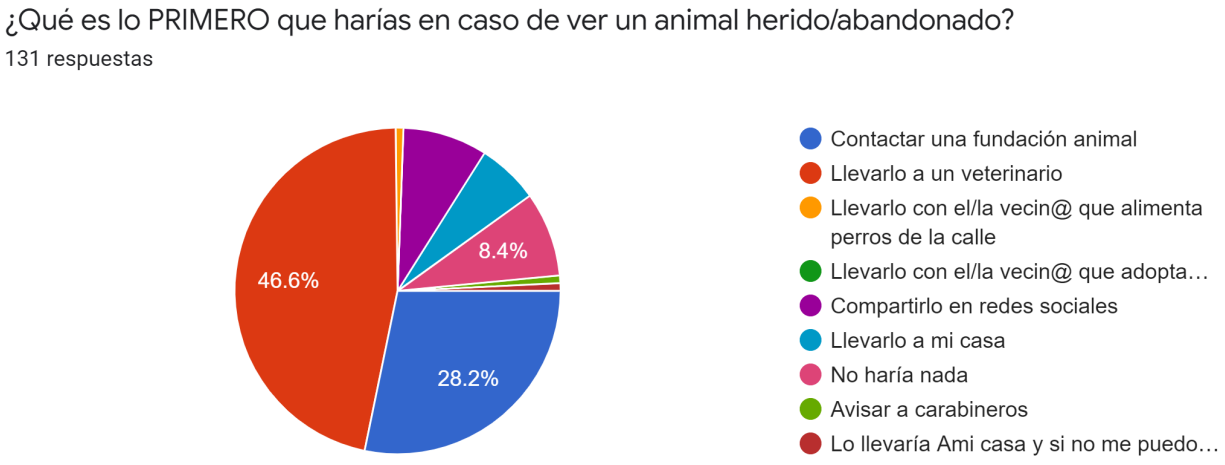


Figura 41: Fuente: Elaboración propia

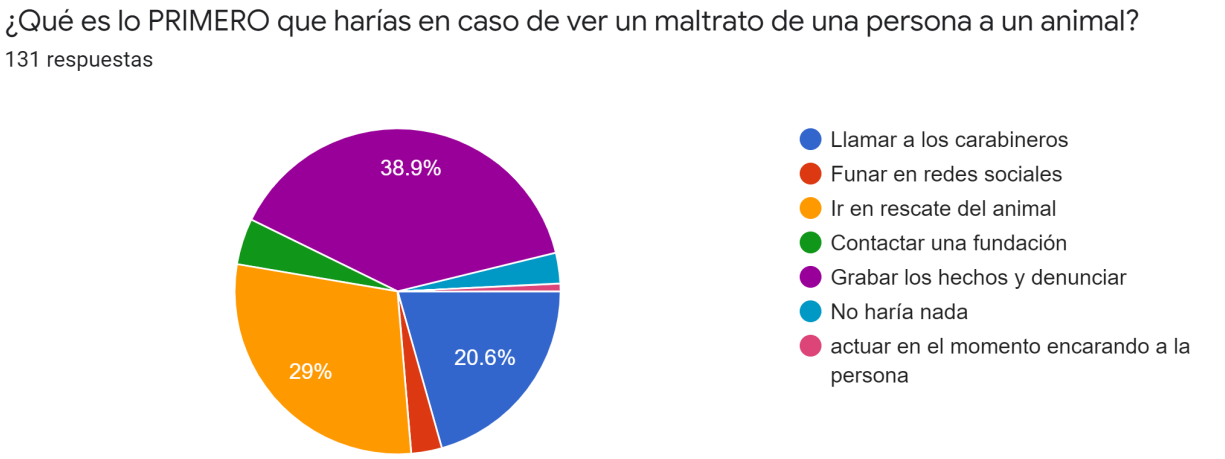


Figura 42: Fuente: Elaboración propia

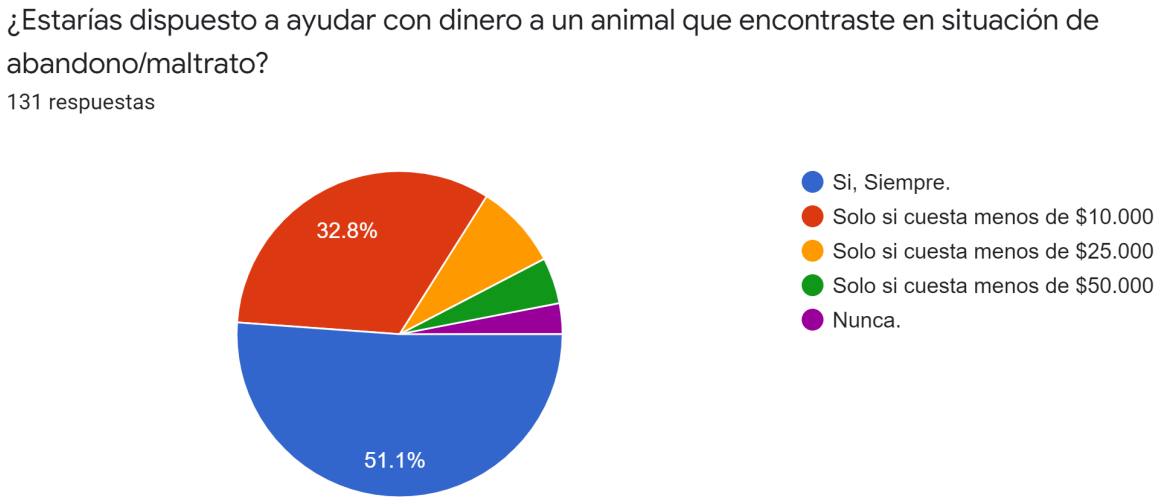


Figura 43: Fuente: Elaboración propia

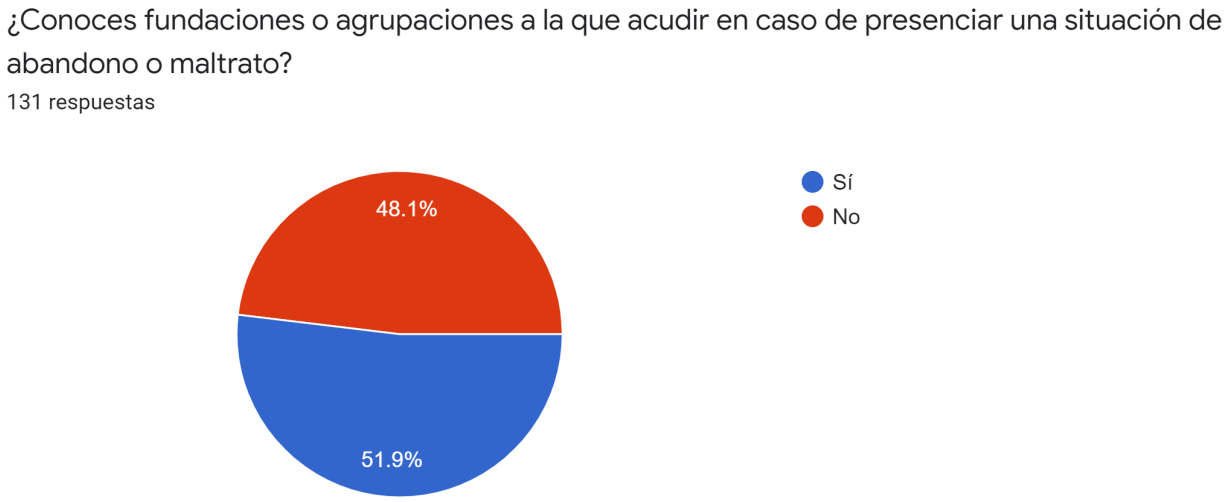


Figura 44: Fuente: Elaboración propia

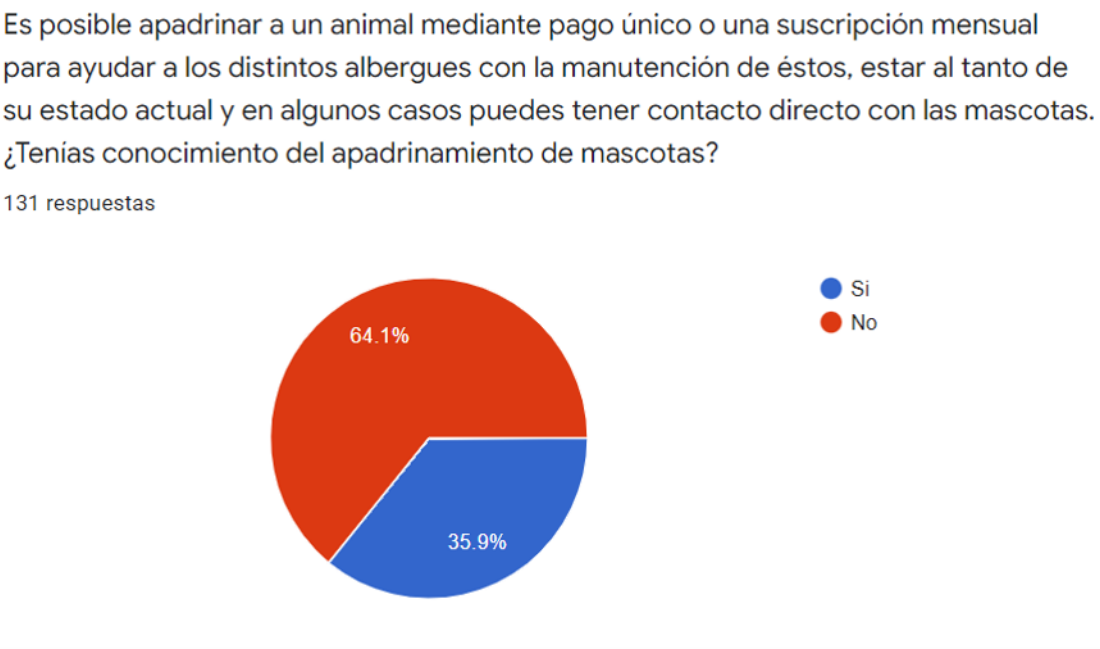


Figura 45: Fuente: Elaboración propia

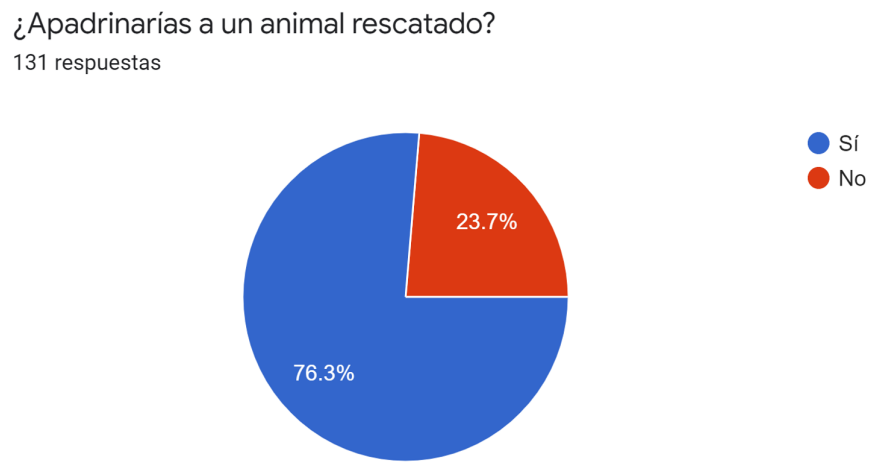


Figura 46: Fuente: Elaboración propia

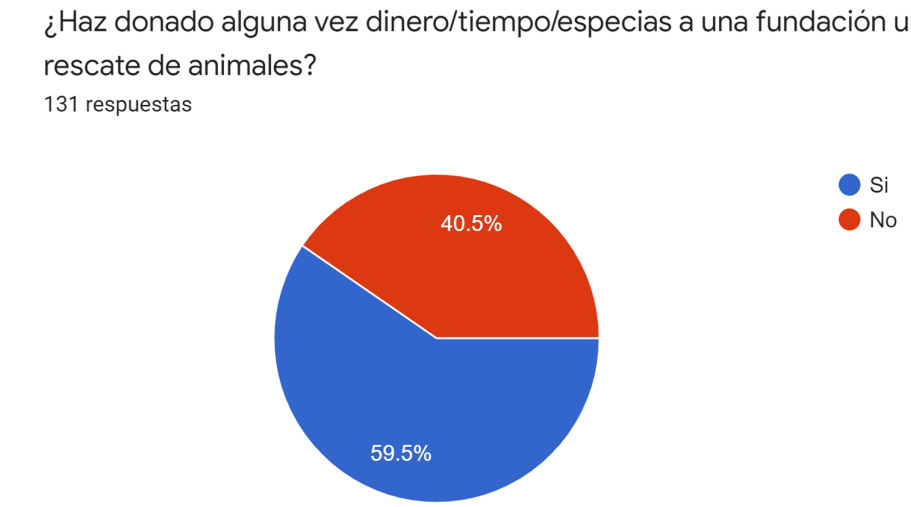


Figura 47: Fuente: Elaboración propia

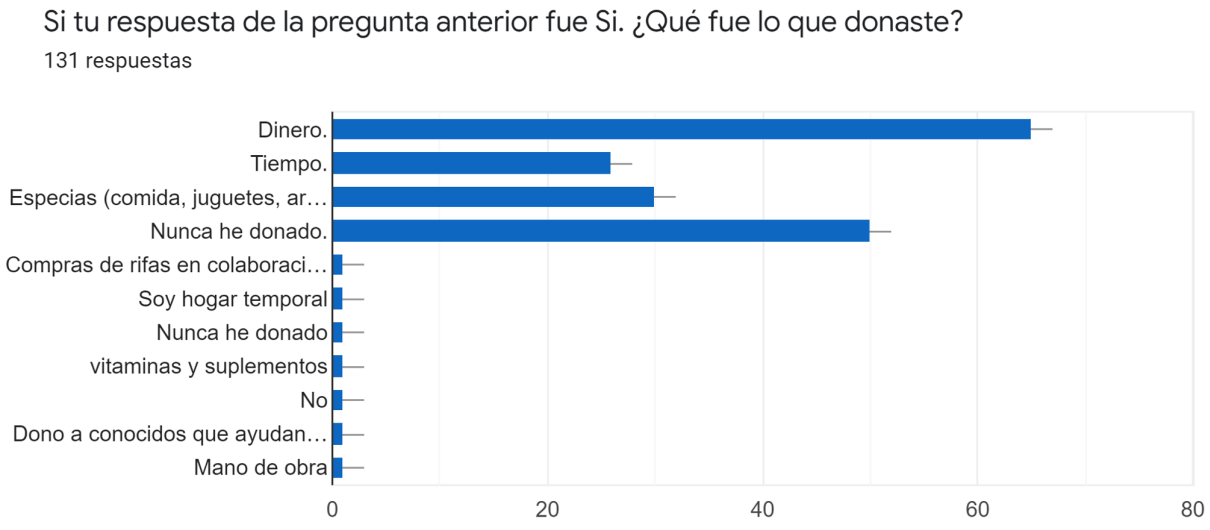


Figura 48: Fuente: Elaboración propia

9.3. Validaciones secciones de la plataforma

9.3.1. Voluntaria de una fundación

`https://drive.google.com/drive/folders/1MwTV5OpqBaFr_EMFH46MA-MAPzrLpPmL`

9.3.2. Voluntaria de una organización

https://drive.google.com/drive/folders/1MwTV5OpqBaFr_EMFH46MA-MAPzrLpPmL

9.3.3. Personas comunes con leve relación con el mundo del rescate animal

https://drive.google.com/drive/folders/1MwTV5OpqBaFr_EMFH46MA-MAPzrLpPmL

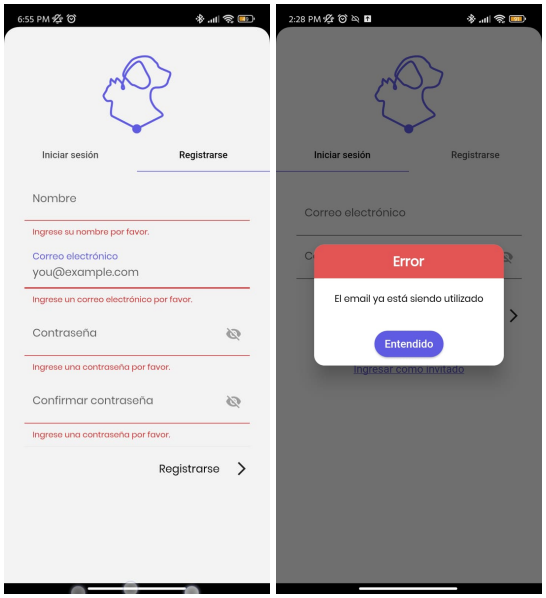
9.3.4. Personas comunes sin relación con el mundo del rescate animal

https://drive.google.com/drive/folders/1MwTV5OpqBaFr_EMFH46MA-MAPzrLpPmL

https://drive.google.com/drive/folders/1MwTV5OpqBaFr_EMFH46MA-MAPzrLpPmL

9.4. Manejo de errores Frontend

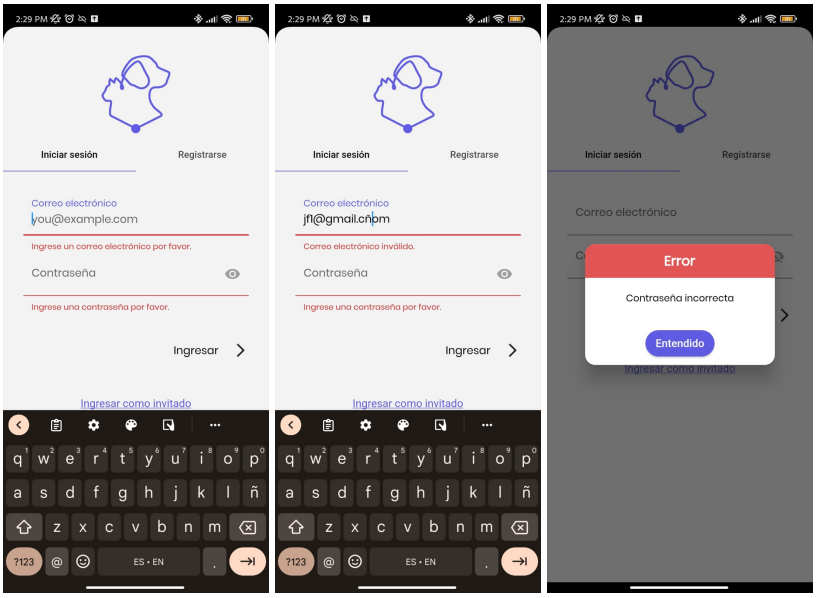
9.4.1. Registro de Usuarios



(a) (b)

Figura 49: (a) Campos vacíos (b) Correo ya registrado

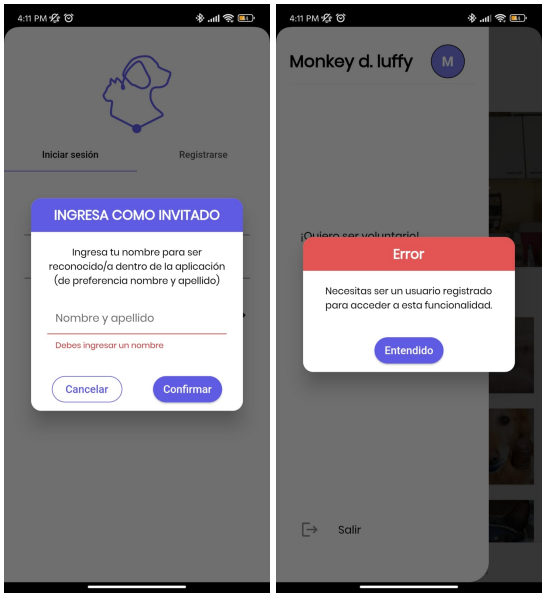
9.4.2. Ingreso de Usuarios



(a) (b) (c)

Figura 50: (a) Campos vacíos (c) Correo inválido (d) Contraseña incorrecta

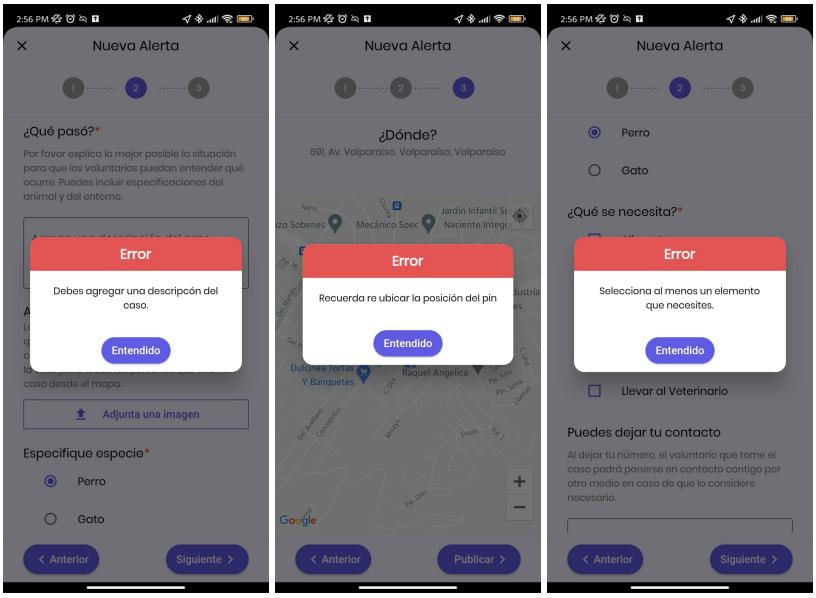
9.4.3. Ingreso de invitados



(a) (b)

Figura 51: (a) Campos vacíos (b) Necesitas ser un usuario registrado

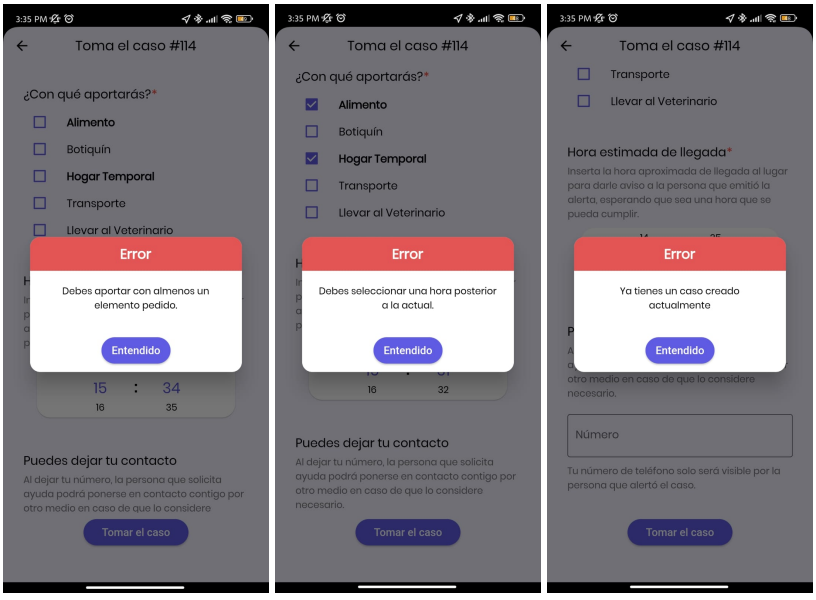
9.4.4. Crear alerta



(a) (b) (c)

Figura 52: (a) Campo Descripción vacío (b) Posición (c) Sin elementos para el caso

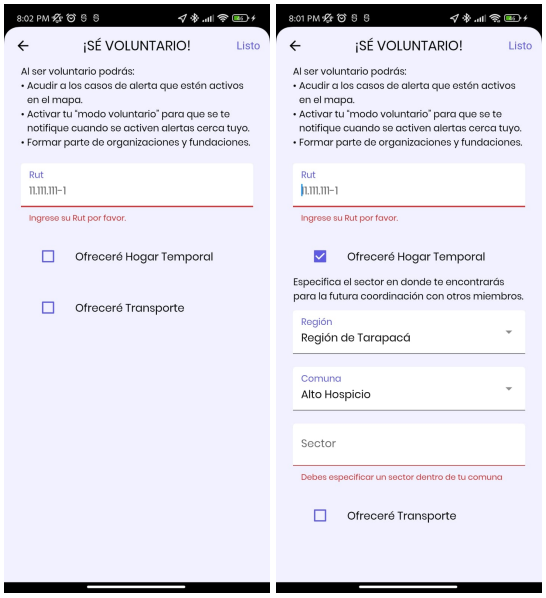
9.4.5. Tomar alerta



(a) (b) (c)

Figura 53: (a) Aporte (b) Hora (c) Ya tienes un caso

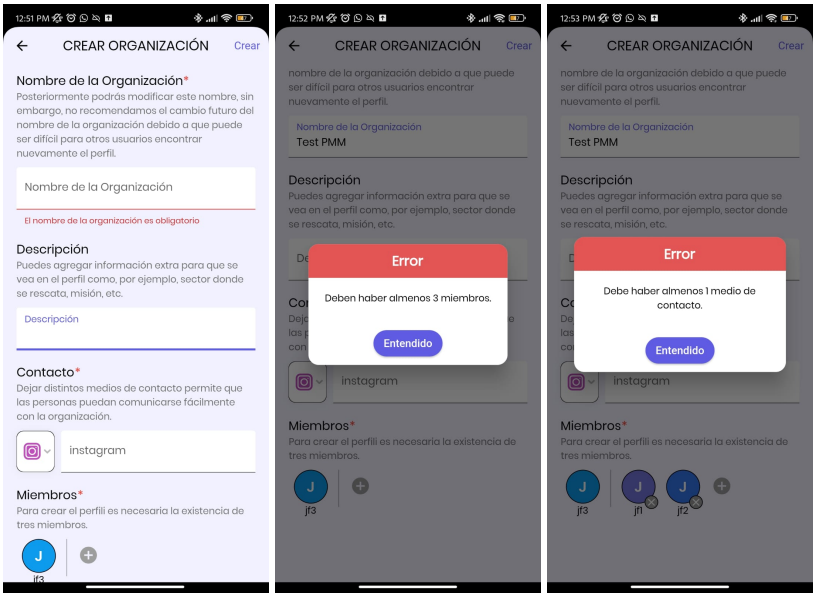
9.4.6. Quiero ser voluntario



(a) (b)

Figura 54: (a) Campo de RUT vacío (b) Sector vacío

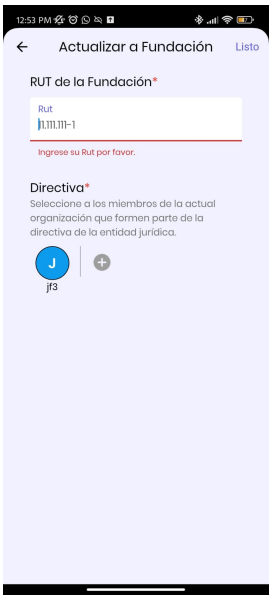
9.4.7. Crear organización



(a) (b) (c)

Figura 55: (a) Campo de nombre vacío (b) Miembros (c) Medios de contacto

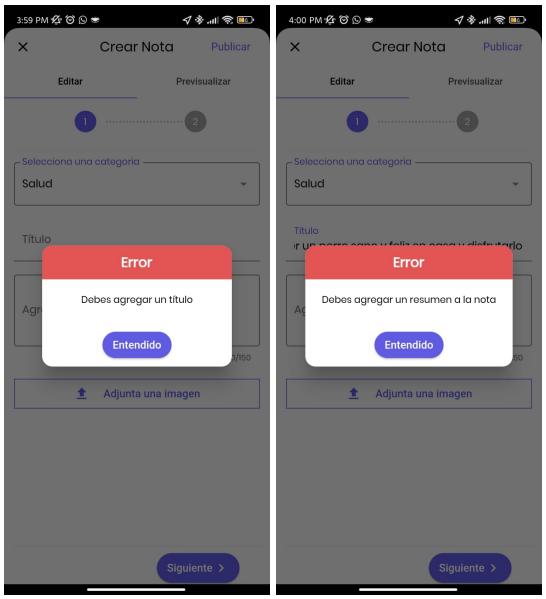
9.4.8. Actualizar a fundación



(a)

Figura 56: (a) Campo de RUT vacío

9.4.9. Crear publicación



(a) (b)

Figura 57: (a) Campo de título vacío (b) Campo de resumen vacío

10. Referencias

[1] El Chile Que Viene. (2021, junio 8). Sitio Web Cadem. https://cadem.cl/chile-que-viene/

[2] ChileAtiende. (s. f.). Ley de Tenencia Responsable de Mascotas y Animales de Compañía (Ley Cholito). Gob.cl. Recuperado 3 de diciembre de 2022, de https://www.chileatiende.gob.cl/fichas/51436-ley-de-tenencia-responsable-de-mascotas-y-animales-de-compania-ley-cholito

[3] Fernandez, N. (2022). Preguntas frecuentes. Bookmate. https://www.wuf.pe/preguntas-frecuentes

[4] SOSAFE - Hacer del mundo un lugar más seguro. (s. f.). Sosafeapp.com. Recuperado 3 de diciembre de 2022, de https://es.sosafeapp.com/

[5] Voy Contigo - Seguridad y Comunidad para Mujeres. (2021, marzo 13). Voy Contigo Aportes. https://voycontigo.app/

[6] INT'L homepage. (2021, noviembre 9). Life360. https://www.life360.com/intl/

[7] PetPins - Everything about pets near you. (s. f.). PetPins. Recuperado 3 de diciembre de 2022, de https://www.petpins.net/

[8] NestJS - A progressive Node.js framework. (s. f.). NestJS - A progressive Node.js framework. Recuperado 3 de diciembre de 2022, de https://nestjs.com/

[9] Cloud application platform. (s. f.). Heroku.com. Recuperado 3 de diciembre de 2022, de https://www.heroku.com/

[10] Node.Js. (s. f.). Node.Js. Recuperado 3 de diciembre de 2022, de https://nodejs.org/en/

[11] Express - Infraestructura de aplicaciones web Node.js. (s. f.). Expressjs.com. Recuperado 3 de diciembre de 2022, de https://expressjs.com/es/

[12] Fastify, Fast and low overhead web framework, for Node.js. (s. f.). Fastify.io. Recuperado 3 de diciembre de 2022, de https://www.fastify.io/

[13] PostgreSQL. (2022, diciembre 3). PostgreSQL. https://www.postgresql.org/

[14] MySQL. (s. f.). Mysql.com. Recuperado 3 de diciembre de 2022, de https://www.mysql.com/

[15] AngularJS — superheroic JavaScript MVW framework. (s. f.). Angularjs.org. Recuperado 3 de diciembre de 2022, de https://angularjs.org/

[16] Data Transfer Object. (s. f.). Martinfowler.com. Recuperado 3 de diciembre de 2022, de https://martinfowler.com/eaaCatalog/dataTransferObject.html

[17] JSON. (s. f.). Json.org. Recuperado 3 de diciembre de 2022, de https://www.json.org/json-es.html

[18] Abba, I. V. (2022, octubre 21). What is an ORM – the meaning of object Relational Mapping database tools. Freecodecamp.org. https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/

[19] Prisma. (s. f.). Prisma. Recuperado 3 de diciembre de 2022, de https://www.prisma.io/

[20] Prisma CLI. (s. f.). Prisma. Recuperado 3 de diciembre de 2022, de https://www.prisma.io/docs/concepts/components/prisma-cli

[21] JSON web tokens - jwt.Io. (s. f.). Jwt.Io; Auth0. Recuperado 3 de diciembre de 2022, de https://jwt.io/

[22] Jones, M., Bradley, J., Sakimura, N. (2015). JSON Web Token (JWT). RFC Editor.

[23] Passport-jwt. (s. f.). Passport.Js. Recuperado 3 de diciembre de 2022, de http://www.passportjs.org/packages/passport-jwt/

[24] Documentation. (s. f.). Documentation — NestJS - A Progressive Node.Js Framework. Recuperado 3 de diciembre de 2022, de https://docs.nestjs.com/security/authentication

[25] Npm: Argon2. (s. f.). Npm. Recuperado 3 de diciembre de 2022, de https://www.npmjs.com/package/argon2

[26] Cross platform implementation. (s. f.). Reactnative.dev. Recuperado 4 de diciembre de 2022, de https://reactnative.dev/architecture/xplat-implementation

[27] Build apps for any screen. (s. f.). Flutter.dev. Recuperado 3 de diciembre de 2022, de https://flutter.dev/?gclid=CjwKCAiAhKycBhAQEiwAgf19eozIuCKBeMkovojZZCrLcrLUEyoSIna6JldUMUyT6w1qT6PqxyfiLBoCedEQAvD_BwE&gclsrc=aw.ds

[28] Bloc state management library. (s/f-a). Bloclibrary.dev. Recuperado el 26 de noviembre de 2022, de https://bloclibrary.dev/#/architecture

[29] Developers, G. [@GoogleDevelopers]. (2018, mayo 10). Build reactive mobile apps with Flutter (Google I/O '18). Youtube. https://www.youtube.com/watch?v=RS36gBEp8OI

[30] List of state management approaches. (s/f). Flutter.dev. Recuperado el 26 de noviembre de 2022, de https://docs.flutter.dev/development/data-and-backend/state-mgmt/options

[31] Bloc state management library. (s/f-b). Bloclibrary.dev. Recuperado el 26 de noviembre de 2022, de https://bloclibrary.dev/#/coreconcepts?id=cubit

[32] Apple developer documentation. (s/f). Apple.com. Recuperado el 26 de noviembre de 2022, de https://developer.apple.com/documentation/security/keychain_services#//apple_ref/doc/uid/TP30000897-CH203-TP1

[33] Android developers. (s/f). Android Developers. Recuperado el 26 de noviembre de 2022, de https://developer.android.com/topic/security/data

[34] Cloud Storage for. (s/f). Firebase. Recuperado el 26 de noviembre de 2022, de https://firebase.google.com/docs/storage

[35] Set up a Firebase Cloud Messaging client app on Flutter. (n.d.). Firebase. Retrieved November 26, 2022, from https://firebase.google.com/docs/cloud-messaging/flutter/client

[36] Http. (s/f). Dart Packages. Recuperado el 26 de noviembre de 2022, de https://pub.dev/packages/http

[37] Connectivity Plus. (s/f). Dart Packages. Recuperado el 26 de noviembre de 2022, de https://pub.dev/packages/connectivity_plus

[38] Flutter Local Notifications. (s/f). Dart Packages. Recuperado el 26 de noviembre de 2022, de https://pub.dev/packages/flutter_local_notifications

[39] Flutter Secure Storage. (s/f). Dart Packages. Recuperado el 26 de noviembre de 2022, de https://pub.dev/packages/flutter_secure_storage

[40] Shared Preferences. (s/f). Dart Packages. Recuperado el 26 de noviembre de 2022, de https://pub.dev/packages/shared_preferences

[41] The Heroku CLI. (s. f.). Heroku.com. Recuperado 3 de diciembre de 2022, de https://devcenter.heroku.com/articles/heroku-cli

[42] Bienestar Animal Postman API Workspace. Recuperado 3 de diciembre de 2022, de https://documenter.getpostman.com/view/8939439/2s7YYu5hoi