



UNIVERSIDAD DE GRANADA

Universidad de Granada

INGENIERÍA INFORMÁTICA

Trabajo fin de grado

DESARROLLO DE APLICACIÓN MÓVIL PARA ADOPCIÓN CANINA

Autora

Laura Vega Palacios

Tutor

Juan José Escobar Pérez

Junio de 2024

Resumen

A día de hoy, comenzar un proceso de adopción puede ser algo tedioso debido a lo descentralizada que está toda la información relacionada con las protectoras y refugios, una persona que quiera adoptar algún animal normalmente va a necesitar consultar diversas plataformas o redes para encontrar una protectora cercana o un animal que se adapte a su estilo de vida.

El desarrollo de este *Trabajo de Fin de Grado* tiene como finalidad la planificación y desarrollo de una aplicación móvil que permita la gestión de adopciones, la información de las protectoras y facilite la comunicación entre los adoptantes y las protectoras. En esta aplicación se podrán dar de alta protectoras o refugios, con sus correspondientes datos y ubicación. Las protectoras o refugios que estén verificadas por los administradores tendrán la capacidad de subir a la plataforma todos los caninos, que aparecerán en las listas de adopción o acogida dependiendo de lo que se haya marcado para cada uno. También se podrán dar de alta usuarios que busquen adoptar o acoger algún perro. Existirán formularios para dar de alta o editar la información del perro, con distintos apartados para cumplimentar todos los datos requeridos como la descripción, la raza, la edad, el peso, etc. Los usuarios también dispondrán de formularios para darse de alta y editar su información. La aplicación utilizará un sistema de geolocalización para mostrar a los usuarios resultados dentro de la aplicación y su distancia. Se podrán ver los resultados de la aplicación en un mapa insertado en la misma. También se incluirán barras de búsqueda y filtros rápidos para acotar resultados según los requisitos del usuario. Se incluirá dentro de la aplicación un sistema de mensajería para facilitar la comunicación, tanto entre usuarios y protectoras como con los administradores. El chat permitirá mensajes de texto e imágenes. Se añadirá, además, un sistema de favoritos y la posibilidad de compartir caninos de la aplicación de forma externa a través de un enlace. La aplicación constará de una página de información legal y de contacto a disposición del usuario.

Abstract

Currently, starting an adoption process can be quite tedious due to how decentralized all the information related to shelters and refuges is, a person who wants to adopt an animal usually will need to use various platforms or networks to find a nearby shelter or an animal that fits their lifestyle.

The purpose of this *Final Degree Project* is the planning and development of a mobile application that allows the management of adoptions, information on shelters, and facilitates communication between adopters and shelters. In this application, shelters or refuges can register, with their corresponding data and location. Shelters or refuges verified by administrators will have the ability to upload all the canines to the platform, which will appear in the adoption or foster lists depending on what has been marked for each canine. Users looking to adopt or foster a dog can also register. There will be forms to register or edit the dog's information, with different sections to complete all the required data such as description, breed, age, weight, etc. Users will also have forms to register and edit their information. The application will use a geolocation system to show users results within the application and their distance. The application results can be viewed on a map embedded in it. Search bars and quick filters will also be included to narrow results according to user requirements. A messaging system will be included in the application to facilitate communication, both between users and shelters, as well as with administrators. The chat will allow text messages and images. Additionally, a favorites system will be added, and it will be possible to share canines from the application externally via a link. The application will consist of a legal information and contact page available to the user.

Agradecimientos

A mis tíos, que me dieron la oportunidad de estudiar y seguir adelante.

Índice

1. Introducción	15
1.1. Motivación	15
1.2. Objetivos del proyecto	18
2. Diagrama de Gantt	22
3. Análisis	24
3.1. Definición y especificación de Requisitos	24
3.1.1. Requisitos funcionales	24
3.1.2. Requisitos no funcionales	33
3.2. Diagramas de casos de uso	35
3.3. Recursos	39
3.3.1. Recursos humanos	39
3.3.2. Hardware	39
3.3.3. Software	39
3.4. Costes	41
3.4.1. Recursos humanos	41
3.4.2. Hardware	41
3.4.3. Software	42
3.4.4. Otros	42
4. Diseño	43
4.1. Diseño de base de datos	43
4.1.1. Direcciones	43
4.1.2. Chats	44
4.1.3. Perros	45
4.1.4. Mensajes	47
4.1.5. Notificaciones	48
4.1.6. Usuarios	49
4.1.7. Imágenes	50
4.2. Diseño de interfaz de usuario	51
4.2.1. Tema	51
4.2.2. Nombre	52
4.2.3. Logo e iconos	53
4.2.4. Diseños	58
5. Implementación	64

5.1. Módulos y clases	64
5.2. Pantallas	78
6. Conclusiones	112
7. Bibliografía	114

Índice de cuadros

1. Objetivo 1	18
2. Objetivo 2	18
3. Objetivo 3	18
4. Objetivo 4	18
5. Objetivo 5	19
6. Objetivo 6	19
7. Objetivo 7	19
8. Objetivo 8	19
9. Objetivo 9	20
10. Objetivo 10	20
11. Objetivo 11	20
12. Objetivo 12	20
13. Objetivo 13	20
14. Objetivo 14	21
15. RF1: Registro de usuarios y protectoras.	24
16. RF2: Registro de caninos.	24
17. RF3: Mostrar listas de usuarios.	24
18. RF4: Mostrar listas de perros.	25
19. RF5: Capacidad de filtrado por propiedades en listas de perros.	25
20. RF6: Capacidad de filtrado por búsqueda en listas de perros.	25
21. RF7: Capacidad de filtrado por búsqueda en listas de usuarios..	25
22. RF8: Proporcionar botón de mapa de resultados en listas de usuarios.	26
23. RF9: Proporcionar botón de favoritos en perfiles de perros.	26
24. RF10: Mostrar sección de perros favoritos.	26
25. RF11: Mostrar adopciones recientes.	26
26. RF12: Proporcionar botón de cerrar sesión.	26
27. RF13: Proporcionar botón de contacto.	27
28. RF14: Botón de menu lateral en la app bar.	27
29. RF15: Mostrar datos personales en el menú lateral.	27
30. RF16: Botón de 'Inicio' en el menú lateral.	27
31. RF17: Botón de 'Perfil personal' en el menú lateral.	27

32.	RF18: Botón de 'Mensajes' en el menú lateral.	28
33.	RF19: Botón de 'Protectoras' en el menú lateral.	28
34.	RF20: Botón de 'Mis perros' en el menú lateral.	28
35.	RF21: Botón de 'Usuarios' en el menú lateral.	28
36.	RF22: Botón de 'Información legal' en el menú lateral.	28
37.	RF23: Edición de datos personales y de contacto.	29
38.	RF24: Edición de datos de perros.	29
39.	RF25: Recuperación de contraseña.	29
40.	RF26: Mostrar página de perfil de usuario/protectora.	29
41.	RF27: Mostrar página de perfil de canino.	30
42.	RF28: Compartir perros a través de enlace.	30
43.	RF29: Página del mapa.	30
44.	RF30: Página de chats.	30
45.	RF31: Enviar mensajes al chat.	31
46.	RF32: Enviar imágenes al chat.	31
47.	RF33: Abrir un chat nuevo.	31
48.	RF34: Borrar un chat.	31
49.	RF35: Marcar/desmarcar perro como adoptado.	32
50.	RF36: Página de contacto.	32
51.	RF37: Página de información legal.	32
52.	RF38: Proporcionar página de inicio según rol.	32
53.	RF39: Página de administrador.	32
54.	RF40: Búsqueda de direcciones.	33
55.	RF41: Cambiar a tema oscuro.	33
56.	RF42: Subir post al blog.	33
57.	RF43: Comentar post del blog.	33
58.	Costes del proyecto	42
59.	Tabla: addresses	43
60.	Tabla: chats	44
61.	Tabla: dogs	46
62.	Tabla: messages	47
63.	Tabla: notifications	48
64.	Tabla: requests	49
65.	Tabla: users	50

Índice de figuras

1.	Diagrama de Gannt	22
2.	Casos de uso: Usuario.	36
3.	Casos de uso: Protectora.	37

4.	Casos de uso: Administrador.	38
5.	Paleta	51
6.	Fuente Nunito Sans	52
7.	Logos	53
8.	Icono por defecto para usuario	54
9.	Icono por defecto para perro	54
10.	Icono para listas de adopción	54
11.	Icono para listas de acogida	54
12.	Icono usuario no verificado	55
13.	Icono usuario verificado	55
14.	Icono hueso	55
15.	Icono casa	55
16.	Iconos usuarios	56
17.	Iconos patas	57
18.	Pantalla de carga.	58
19.	Inicio de sesión.	58
20.	Registro/edición de usuario.	59
21.	Registro/edición de protectora.	59
22.	Inicio usuario.	59
23.	Inicio protectora.	59
24.	Lista de perros.	60
25.	Lista de usuarios.	60
26.	Perfil de usuario/administrador.	60
27.	Perfil de protectora.	60
28.	Perfil de perro.	61
29.	Compartir perro.	61
30.	Página de mapa y lista.	61
31.	Acción de redirigir a perfil.	61
32.	Registro/edición canino.	62
33.	Menú lateral.	62
34.	Inicio administrador.	62
35.	Pop-up verificación.	62
36.	Notificaciones.	63
37.	Página de mensajes.	63
38.	Recuperar contraseña.	63
39.	Módulos y sus dependencias.	64
40.	Módulo <i>appbar</i> .	65
41.	Módulo <i>chats</i> .	66
42.	Módulo <i>common</i> .	67
43.	Módulo <i>condition</i> .	68

44.	Módulo <i>contact</i>	68
45.	Módulo <i>db</i>	69
46.	Módulo <i>dog</i>	70
47.	Módulo <i>home</i>	71
48.	Módulo <i>init</i>	71
49.	Módulo <i>legal</i>	72
50.	Módulo <i>login</i>	72
51.	Módulo <i>message</i>	73
52.	Módulo <i>orderby</i>	73
53.	Módulo <i>register</i>	74
54.	Módulo <i>section</i>	75
55.	Módulo <i>services</i>	75
56.	Módulo <i>update</i>	76
57.	Módulo <i>user</i>	77
58.	Pantalla de carga	78
59.	Pantalla de inicio de sesión.	79
60.	Pantalla de recuperación de contraseña.	80
61.	Página de opciones de registro.	81
62.	Pantallas de registro.	82
63.	Pantallas de registro con errores.	83
64.	Barra de búsqueda de direcciones.	84
65.	Pantallas de inicio por rol.	85
66.	Menús laterales por rol.	86
67.	Pantalla de protectora no verificada.	87
68.	Lista de usuarios.	90
69.	Página de mapa.	91
70.	Lista de perros.	92
71.	Lista de perros con filtros.	93
72.	Filtros.	94
73.	Pantalla de perfil personal.	95
74.	Actualización de datos personales.	96
75.	Pantalla <i>Mis perros</i>	97
76.	Pantalla de registro de perro.	98
77.	Pantalla de perfil de perro.	99
78.	Pop-up de marcar/desmarcar como adoptado.	100
79.	Borrar perro.	101
80.	Compartir perro.	102
81.	Formulario de actualización.	103
82.	Notificación de actualización correcta.	104
83.	Pantalla de chats disponibles.	106

84.	Pantalla de información legal.	107
85.	Pantalla inicio de administrador.	110
86.	Verificar/desverificar usuario.	111

1. Introducción

1.1. Motivación

Numerosas familias, todos los años, se animan a incluir a una mascota en su círculo. Sin embargo, miles de mascotas son a su vez abandonadas por muchas de estas familias en todo el mundo. Existen [estudios \[1\]](#) que indican que casi 300.000 perros y gatos fueron recogidos durante 2022. Estos datos hacen saltar las alarmas de muchas de las asociaciones que luchan por el bienestar y los derechos de los animales. De todos los animales que son abandonados, muchos permanecen en las calles durante el resto de su vida; otros son recogidos por las autoridades pertinentes y terminan en protectoras o refugios, a la espera de encontrar otra familia. Existen organizaciones sin ánimo de lucro que se encargan de ayudar a muchas de las mascotas que están en las calles. Algunas de estas organizaciones son públicas y subvencionadas por el estado. Para garantizar el bienestar y la protección de los animales, en España se publicó una [ley \[2\]](#) en la que se tratan principalmente los siguientes puntos:

- **Principios generales**
 - Reconocimiento de los animales como seres dotados de sensibilidad.
 - Fomento de la adopción en lugar de la compra de animales.
 - Prohibición de prácticas que causen sufrimiento o estrés innecesario a los animales.
 - Concienciar acerca del bienestar y respeto animal.
- **Responsabilidad y tenencia responsable de animales de compañía**
 - Establecimiento de requisitos mínimos de bienestar, cuidado y alojamiento.
 - Obligación de identificación y registro de los animales de compañía.
 - Establecimiento de las obligaciones de los propietarios en cuanto a la tenencia responsable, incluyendo la alimentación, hogar y atención veterinaria.
 - Prohibición de mantener animales en condiciones inadecuadas o de privarles de cuidados esenciales.
- [noitemsep]
- **Cria y comercio de animales**
 - Regulación estricta de la cría y comercio de animales de compañía para evitar la explotación y el maltrato.
 - Obligación de los criadores y comerciantes de cumplir con requisitos específicos de bienestar animal.
 - Prohibición de la cría indiscriminada y la venta de animales en tiendas físicas, salvo excepciones debidamente justificadas.

- **Concienciación y medidas del estado contra abandonos y abusos**
 - Promoción de la educación y el respeto y protección de los animales.
 - Campañas de concienciación pública sobre el bienestar animal y la tenencia responsable.
 - Tipificación de conductas de maltrato y abandono como infracciones administrativas o penales, además del establecimiento de sanciones proporcionales a la gravedad de las infracciones.
 - Obligación de las administraciones públicas de establecer y mantener refugios y centros de acogida para animales abandonados.
 - Creación de un registro nacional de animales de compañía y establecimientos relacionados con ellos.

Esta ley ha sido un avance muy significativo en la legislación española en materia de protección animal, ayudando a crear un entorno más respetuoso y justo para los animales.

A pesar de que la compra de animales en España es legal, muchas organizaciones de bienestar animal y de los derechos de los animales recomiendan optar por la adopción en lugar de la compra. Esta recomendación viene de la mano de que es habitual encontrar criaderos donde los animales no disponen de las condiciones necesarias para vivir adecuadamente. Algunos de estos criaderos carecen de cuidados veterinarios, de una alimentación adecuada y de higiene. En algunos [artículos](#)[3] podemos encontrar más información sobre esta problemática. Si se opta finalmente por la compra, se recomienda visitar los criaderos a los que se va a comprar el animal para poder garantizar que no se están fomentando criaderos ilegales y que cumplen con la ley de bienestar animal.

Cuando se opta por la adopción, es necesario cumplir un proceso que puede variar según la organización a la que se acuda. Hay ciertos pasos que son comunes después de escoger una organización:

- **Elección del perro:** en este paso, se deberán tener en cuenta las necesidades de la mascota para determinar cuál se adapta mejor al estilo de vida del hogar donde va a residir. Lo habitual en esta etapa es interactuar con diferentes mascotas candidatas a ser adoptadas.
- **Solicitud de adopción:** es común que se tenga que cumplimentar un formulario y realizar una entrevista posterior para asegurar que el animal va a ir a un hogar adecuado.
- **Evaluación del hogar:** se organiza una visita al hogar donde va a ir el animal para garantizar que el entorno es seguro y adecuado. También se verifica que se disponga de todo lo necesario para recibir a la mascota, como comederos, juguetes, cama, etc.

- **Contrato de adopción:** es indispensable para garantizar la protección legal de las mascotas adoptadas. El nuevo dueño debe comprometerse legalmente a proporcionar atención veterinaria y a no abandonar al animal, entre otros puntos. En la mayoría de las organizaciones, es habitual vacunar y colocar un chip al animal antes de que vaya al hogar del nuevo dueño.
- **Recogida del animal:** el día de la recogida, se entrega todo el historial médico del animal (si se dispone de él) y la organización suele proporcionar consejos para los nuevos dueños.
- **Seguimiento:** se requiere un seguimiento posterior a la adopción para garantizar el bienestar del animal y brindar apoyo o asesoramiento al dueño.

Todo este proceso es guiado por la organización correspondiente y es importante completarlo, aunque pueda resultar un poco tedioso. Para iniciar este proceso de adopción, la persona deberá ponerse en contacto previamente con las distintas protectoras o refugios en su área. Esta fase inicial puede presentar diferentes problemas a los adoptantes. En la mayoría de las protectoras, trabajan voluntarios y se sostienen de donaciones, por lo que es común que no cuenten con fondos para generar visibilidad. Muchas protectoras utilizan diversas redes sociales o sus propios sitios web para promocionarse, lo que implica que una persona interesada en adoptar podría tener que utilizar varias plataformas para ponerse en contacto con alguna. A veces, es difícil mantener una comunicación adecuada con diferentes protectoras incluso después de la adopción. Otro problema es que los futuros adoptantes pueden encontrar dificultades al elegir una mascota, ya que la información sobre las diferentes mascotas está dispersa y puede ser una tarea complicada buscar una que cumpla con todos los requisitos del adoptante.

Después de revisar los pasos y los posibles inconvenientes para solicitar un proceso de adopción, se propone una solución en forma de aplicación móvil. Con esta aplicación se pretende solventar la problemática de la fase inicial de búsqueda de protectoras, proporcionando a los usuarios listas de protectoras cercanas y de los perros disponibles, incluyendo filtros para acotar la búsqueda. También se propone como una forma de facilitar a las protectoras la gestión de sus perros y posibles adoptantes. Para poder facilitar la comunicación entre usuarios y protectoras se añadirá un chat en tiempo real dentro de la aplicación. La idea principal de esta aplicación es centralizar la información de las organizaciones y sus mascotas, además de concienciar e incentivar a los usuarios a adoptar.

1.2. Objetivos del proyecto

Para cubrir las necesidades del proyecto propuesto, se han definido una serie de objetivos divididos en obligatorios, que serán los requeridos para que la aplicación funcione como se espera, y optionales, para añadir funcionalidades extra o facilitar el uso de la aplicación.

Objetivo 1

Título	<i>Registro de usuarios con diferentes roles</i>
Tipo	Obligatorio
Descripción	La aplicación debe ofrecer la opción de registrarse tanto como usuario o como protectora, debe proporcionar los formularios correspondientes y almacenar los datos en la base de datos.

Objetivo 2

Título	<i>Registro de caninos</i>
Tipo	Obligatorio
Descripción	La aplicación debe ofrecer la posibilidad de registrar a varios perros con diferentes características por parte de los usuarios registrados como protectoras y almacenar los datos en la base de datos.

Objetivo 3

Título	<i>Listas de caninos con filtros y búsqueda</i>
Tipo	Obligatorio
Descripción	La aplicación debe mostrar diferentes listas de perros, permitiendo aplicar filtros rápidos y realizar búsquedas.

Objetivo 4

Título	<i>Listas de usuarios con búsqueda</i>
Tipo	Obligatorio
Descripción	La aplicación debe mostrar diversas listas de usuarios que permitan realizar búsquedas en ellas.

Objetivo 5

Título	<i>Sección de perros favoritos</i>
Tipo	Obligatorio
Descripción	Se debe incluir un botón de favoritos en los perfiles de los perros que permita a los usuarios añadir a su sección de favoritos aquellos perros que marquen como favoritos.

Objetivo 6

Título	<i>Sistema de mensajería entre usuarios</i>
Tipo	Obligatorio
Descripción	Se debe permitir a los usuarios abrir un chat con otros usuarios dentro de la aplicación. Además, los usuarios deberán recibir una notificación cuando reciban un mensaje.

Objetivo 7

Título	<i>Geolocalización y mapas con resultados</i>
Tipo	Obligatorio
Descripción	Se debe incluir el uso de la ubicación para ordenar los resultados de la aplicación según la distancia del usuario. Además, se debe agregar una página con un mapa que muestre la ubicación de los resultados junto con una lista de los mismos.

Objetivo 8

Título	<i>Actualizar datos de caninos</i>
Tipo	Obligatorio
Descripción	Se debe permitir actualizar los datos de los caninos, como el peso, edad, descripción, etc. Además, se debe proporcionar la opción de marcar si un perro está disponible para adoptar/acoger o si ya ha sido adoptado.

Objetivo 9

Título	<i>Actualizar datos de usuarios</i>
Tipo	Obligatorio
Descripción	Se debe permitir a un usuario actualizar sus datos personales, así como sus credenciales.

Objetivo 10

Título	<i>Barra de búsqueda de direcciones</i>
Tipo	Opcional
Descripción	Se debe permitir a un usuario buscar su dirección y autocompletar automáticamente los diferentes campos del formulario.

Objetivo 11

Título	<i>Tema oscuro en la aplicación</i>
Tipo	Opcional
Descripción	Se debe permitir al usuario cambiar al tema oscuro dentro de la aplicación.

Objetivo 12

Título	<i>Blog</i>
Tipo	Opcional
Descripción	Se debe permitir al usuario añadir publicaciones con imágenes y textos, además de añadir comentarios en las publicaciones para compartir ideas con otros usuarios.

Objetivo 13

Título	<i>Compartir perros a través de enlace</i>
Tipo	Opcional
Descripción	Se debe añadir un botón de compartir que permita al usuario generar un enlace al perro correspondiente para poder compartirlo de forma externa a la aplicación.

Objetivo 14

Título	<i>Mandar imágenes dentro del chat</i>
Tipo	Opcional
Descripción	Se debe permitir enviar imágenes dentro del chat.

2. Diagrama de Gannt

En esta sección se definen todas las etapas en las que se va a dividir este proyecto, condensadas en un Diagrama de Gannt. Este diagrama abarca desde enero de 2024 hasta junio del mismo año, estableciendo un marco temporal para cada fase del proyecto y asignando recursos de manera eficiente para alcanzar los objetivos establecidos.

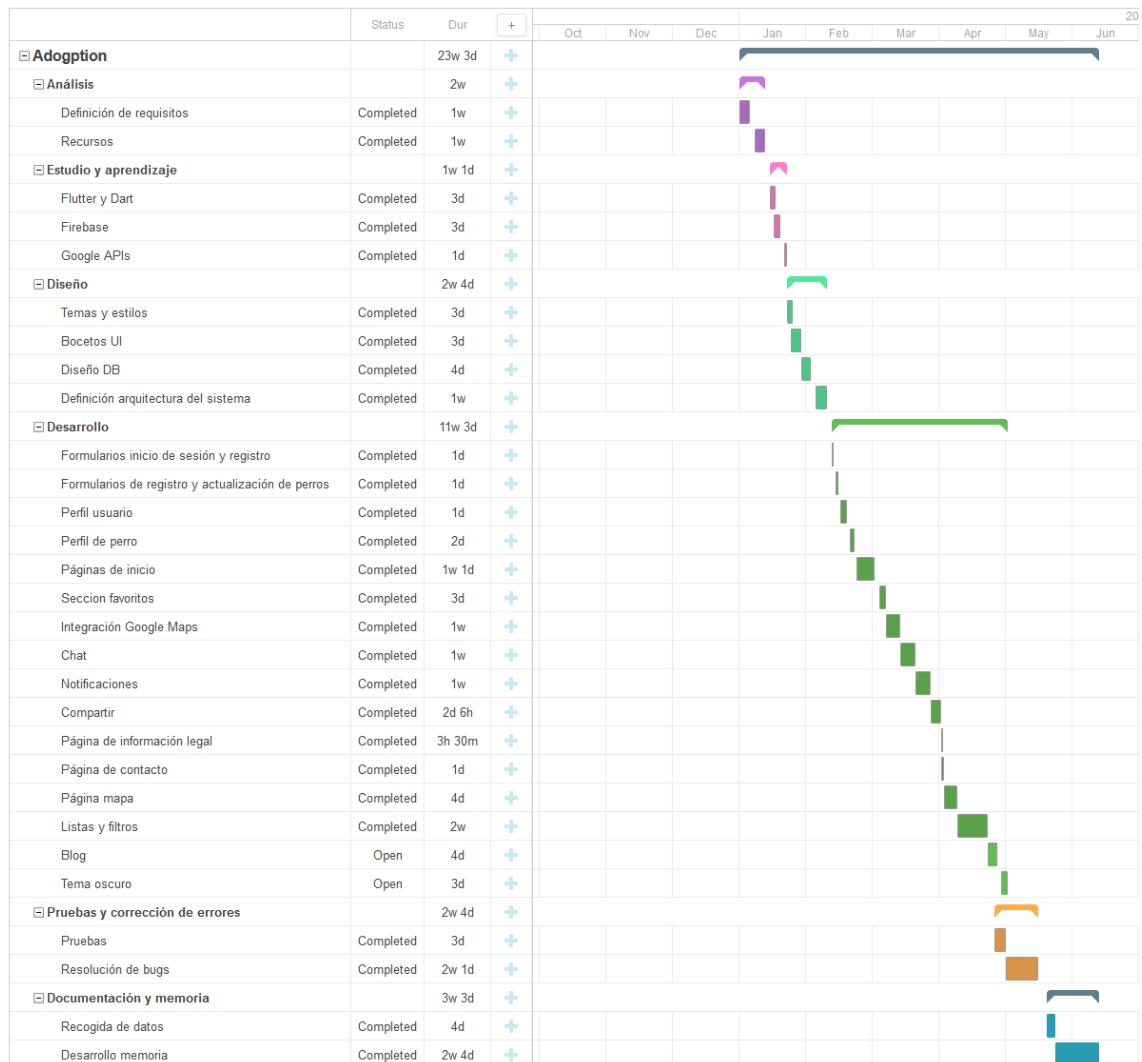


Figura 1: Diagrama de Gannt

Dentro del diagrama se pueden ver las diferentes etapas:

- **Análisis:** Esta etapa consta de unas dos semanas de tiempo, en esta se

definen todos los requisitos del sistema y se identifican todos los recursos humanos, de hardware y software necesarios.

- **Estudio y aprendizaje:** Se destina un breve período previo para aprender a utilizar los lenguajes y servicios que se emplean en el desarrollo de la aplicación.
- **Diseño:** Se reservan casi tres semanas para definir temas, bocetos, diseño de datos y la arquitectura del sistema.
- **Desarrollo:** La etapa más extensa, dividida en secciones para cada funcionalidad a desarrollar a las que se asigna un tiempo adecuado, considerando los servicios y componentes involucrados que definirán su dificultad.
- **Pruebas y corrección de errores:** Una vez completado el desarrollo, es necesario realizar pruebas exhaustivas para identificar errores y faltas de funcionalidades. Se reserva tiempo suficiente para corregir bugs o implementar mejoras de rendimiento si es necesario.
- **Documentación y memoria:** Se dedica un período para recopilar todos los datos y elaborar la memoria en Latex, que documentará el proceso y los resultados obtenidos en las etapas anteriores.

3. Análisis

3.1. Definición y especificación de Requisitos

En esta sección, se detallan los requisitos funcionales y no funcionales identificados para la aplicación que se va a desarrollar. Es esencial definir y comprender estos requisitos para garantizar buenos resultados en el desarrollo y diseño del sistema.

3.1.1. Requisitos funcionales

Los requisitos funcionales describen las acciones específicas que el sistema debe realizar, los servicios que debe proporcionar y cómo debe responder a diversas entradas. Estos requisitos se centran en el "qué" del sistema, delineando las funcionalidades clave que los usuarios esperan encontrar en la aplicación.

RF1: Registro de usuarios y protectoras.

Descripción	Ofrecer un formulario de registro para usuarios y protectoras.
Datos de Entrada	Datos ingresados por el usuario en el formulario de registro. Incluyendo datos de contacto, correo electrónico y dirección.
Datos de Salida	Datos del usuario almacenados en la base de datos.

RF2: Registro de caninos.

Descripción	Ofrecer un formulario de registro para añadir perros con características específicas.
Datos de Entrada	Datos ingresados por la protectora, tales como raza, peso, edad, color y descripción.
Datos de Salida	Datos del perro almacenados en la base de datos, junto al id de su protectora.

RF3: Mostrar listas de usuarios.

Descripción	Mostrar diferentes listas de usuarios en la aplicación, con filtros predeterminados.
Datos de Entrada	Ninguno.
Datos de Salida	Listas de usuarios de la aplicación que cumplen con los filtros.

RF4: Mostrar listas de perros.

Descripción	Mostrar diferentes listas de perros en la aplicación, con filtros predeterminados
Datos de Entrada	Ninguno.
Datos de Salida	Listas de perros de la aplicación que cumplen con los filtros.

RF5: Capacidad de filtrado por propiedades en listas de perros.

Descripción	Proporcionar diferentes filtros para las listas de perros, que corresponden con las diferentes propiedades que puede tener un perro en la aplicación.
Datos de Entrada	Valores de los filtros proporcionados por el usuario.
Datos de Salida	Lista de perros de la aplicación que cumplen con los filtros.

RF6: Capacidad de filtrado por búsqueda en listas de perros.

Descripción	Filtrar resultados de listas según el texto ingresado en una barra de búsqueda, la búsqueda se realiza en diferentes campos de los perros.
Datos de Entrada	Texto de búsqueda ingresado por el usuario.
Datos de Salida	Lista de perros de la aplicación que cumplen con el criterio de búsqueda en alguno de los campos.

RF7: Capacidad de filtrado por búsqueda en listas de usuarios..

Descripción	Filtrar resultados de listas según el texto ingresado en una barra de búsqueda, la búsqueda se realiza en diferentes campos de los usuarios.
Datos de Entrada	Texto de búsqueda ingresado por el usuario.
Datos de Salida	Lista de usuarios de la aplicación que cumplen con el criterio de búsqueda en alguno de los campos.

RF8: Proporcionar botón de mapa de resultados en listas de usuarios.

Descripción	Botón en listas de usuarios que redirige a una página con mapa con todos los resultados con ubicación.
Datos de Entrada	Ninguno
Datos de Salida	Botón con capacidad de redirección a la página del mapa.

RF9: Proporcionar botón de favoritos en perfiles de perros.

Descripción	Permitir a un usuario o protectora marcar como favorito a un perro.
Datos de Entrada	Click del usuario sobre el botón.
Datos de Salida	Datos del perro con la actualización de ids de usuarios que lo han marcado/desmarcado como favorito.

RF10: Mostrar sección de perros favoritos.

Descripción	Muestra una sección de perros marcados como favoritos en ese momento por el usuario en su página de inicio.
Datos de Entrada	Ninguno.
Datos de Salida	Lista horizontal de perros favoritos en la parte inferior de la página de inicio de los usuarios.

RF11: Mostrar adopciones recientes.

Descripción	Muestra un swiper con imágenes de perros marcados como adoptados recientemente.
Datos de Entrada	Ninguno.
Datos de Salida	Swiper con imágenes y nombres de perros adoptados en la aplicación.

RF12: Proporcionar botón de cerrar sesión.

Descripción	Mostrar botón de cerrar sesión en el menú de acciones en la app bar de la aplicación.
Datos de Entrada	Ninguno.
Datos de Salida	Mostrar página de inicio de sesión, después de haber eliminado todos los datos relacionados con el usuario iniciado.

RF13: Proporcionar botón de contacto.

Descripción	Mostrar botón de contacto en el menú de acciones en la app bar de la aplicación.
Datos de Entrada	Ninguno.
Datos de Salida	Mostrar página de contacto.

RF14: Botón de menu lateral en la app bar.

Descripción	Mostrar botón que abre menú lateral.
Datos de Entrada	Ninguno.
Datos de Salida	Menú lateral con las páginas disponibles para ese usuario.

RF15: Mostrar datos personales en el menú lateral.

Descripción	Muestra los datos del usuario iniciado en el menú lateral.
Datos de Entrada	Ninguno.
Datos de Salida	Datos del usuario almacenados en la base de datos.

RF16: Botón de 'Inicio' en el menú lateral.

Descripción	Proporcionar botón para redirigir al usuario a la página de inicio de la aplicación.
Datos de Entrada	Ninguno.
Datos de Salida	Página de inicio de la aplicación.

RF17: Botón de 'Perfil personal' en el menú lateral.

Descripción	Proporcionar botón para redirigir al usuario a la página de inicio de la aplicación.
Datos de Entrada	Ninguno.
Datos de Salida	Página de perfil personal.

RF18: Botón de 'Mensajes' en el menú lateral.

Descripción	Proporcionar botón para redirigir al usuario a la página de chats de la aplicación.
Datos de Entrada	Ninguno.
Datos de Salida	Página de chats.

RF19: Botón de 'Protectoras' en el menú lateral.

Descripción	Proporcionar botón para redirigir al usuario a la lista de protectoras de la aplicación.
Datos de Entrada	Ninguno.
Datos de Salida	Lista de protectoras ya verificadas en la aplicación.

RF20: Botón de 'Mis perros' en el menú lateral.

Descripción	Proporcionar botón para redirigir al usuario a la lista de sus perros de la aplicación
Datos de Entrada	Ninguno.
Datos de Salida	Lista de perros que ha dado de alta el usuario en la aplicación.

RF21: Botón de 'Usuarios' en el menú lateral.

Descripción	Proporcionar botón para redirigir al usuario a la lista de usuarios de la aplicación.
Datos de Entrada	Ninguno.
Datos de Salida	Lista de usuarios que no son protectoras en la aplicación.

RF22: Botón de 'Información legal' en el menú lateral.

Descripción	Proporcionar botón para redirigir al usuario a la página de información legal de la aplicación.
Datos de Entrada	Ninguno.
Datos de Salida	Página de información legal de la aplicación.

RF23: Edición de datos personales y de contacto.

Descripción	Proporcionar formularios de edición para los datos del usuario.
Datos de Entrada	Datos ingresados por el usuario en el formulario. Puede haber datos sin actualizar.
Datos de Salida	Datos del usuario actualizados en la base de datos.

RF24: Edición de datos de perros.

Descripción	Proporcionar formularios de edición para los datos de los caninos.
Datos de Entrada	Datos ingresados por el usuario en el formulario. Puede haber datos sin actualizar.
Datos de Salida	Datos del canino actualizados en la base de datos.

RF25: Recuperación de contraseña.

Descripción	Proporcionar formularios de recuperación de contraseña para los usuarios y protectoras.
Datos de Entrada	Correo electrónico ingresado por el usuario.
Datos de Salida	Correo electrónico con las instrucciones de recuperación de contraseña.

RF26: Mostrar página de perfil de usuario/protectora.

Descripción	Proporcionar una interfaz que condense toda la información de un usuario en una sola página.
Datos de Entrada	Ninguno.
Datos de Salida	Datos del usuario correspondiente, incluyendo foto de perfil, correo y los perros (si los tiene). Además incluirá el botón de contacto para abrir chat.

RF27: Mostrar página de perfil de canino.

Descripción	Proporcionar una interfaz que condense toda la información relacionada con un perro en una sola página.
Datos de Entrada	Ninguno.
Datos de Salida	Datos del perro correspondiente, incluyendo foto de perfil, características del canino (edad, raza, peso, descripción...). Además de un mapa indicando su ubicación, el botón de favoritos y de compartir. También incluye el botón de abrir chat. Contendrá los botones de edición si el perro es del usuario que visita el perfil.

RF28: Compartir perros a través de enlace.

Descripción	Proporcionar un botón que genere un enlace para redirigir a un usuario al perfil de ese perro en concreto.
Datos de Entrada	Ninguno.
Datos de Salida	Url a la aplicación con los parámetros correspondientes para redirigir al perfil de perro.

RF29: Página del mapa.

Descripción	Proporcionar una interfaz que contenga un mapa con diferentes marcadores para cada una de las protectoras y un listado de las mismas debajo del mapa.
Datos de Entrada	Ninguno.
Datos de Salida	Mapa con marcadores y listado de protectoras, con elementos que pueden mover la cámara a los diferentes marcadores del mapa.

RF30: Página de chats.

Descripción	Proporcionar una interfaz que contenga un listado de chats abiertos dentro de la aplicación.
Datos de Entrada	Ninguno.
Datos de Salida	Listado de chats disponibles en la aplicación, incluyen una preview del último mensaje y datos del usuario correspondiente.

RF31: Enviar mensajes al chat.

Descripción	Permitir mandar mensajes a otro usuario dentro de la aplicación.
Datos de Entrada	Texto ingresado por el usuario.
Datos de Salida	Mensaje guardado en la base de datos Se actualizar la tabla de chats de la base de datos con el último mensaje enviado y generar el id del chat si es el primer mensaje. Notificación al usuario receptor.

RF32: Enviar imágenes al chat.

Descripción	Permitir mandar mensajes a otro usuario dentro de la aplicación.
Datos de Entrada	Imagen adjuntada por el usuario.
Datos de Salida	Imagen guardada además de su referencia en la tabla de mensaje. Se la tabla de chats de la base de datos con el último mensaje enviado y generar el id del chat si es el primer mensaje. Notificación al usuario receptor.

RF33: Abrir un chat nuevo.

Descripción	Proporcionar un botón de contacto para abrir un chat con algún usuario dentro de la aplicación.
Datos de Entrada	Ninguno.
Datos de Salida	Interfaz del chat, con la cabecera que incluye los datos del usuario receptor.

RF34: Borrar un chat.

Descripción	Deslizar un chat para eliminarlo de la lista de chats disponibles
Datos de Entrada	Ninguno.
Datos de Salida	Lista actualizada sin el chat que se acaba de eliminar

RF35: Marcar/desmarcar perro como adoptado.

Descripción	Proporcionar un botón dentro del perfil del perro que permite a una protectora marcar/desmarcar a un perro de la aplicación como adoptado.
Datos de Entrada	Estado nuevo según si se quiere marcar o desmarcar.
Datos de Salida	Perfil del canino actualizado y una etiqueta que indica si está adoptado.

RF36: Página de contacto.

Descripción	Proporcionar una interfaz que reúna datos de contacto de administradores.
Datos de Entrada	Ninguno.
Datos de Salida	Interfaz que incluye correo electrónico y un botón de contacto para abrir un chat con un administrador.

RF37: Página de información legal.

Descripción	Proporcionar una interfaz que reúna toda la información legal y advertencias.
Datos de Entrada	Ninguno.
Datos de Salida	Interfaz que incluye toda la información legal de la aplicación.

RF38: Proporcionar página de inicio según rol.

Descripción	Proporcionar una interfaz única según el rol después de iniciar sesión.
Datos de Entrada	Correo electrónico y contraseña para iniciar sesión.
Datos de Salida	Página de inicio que varía según el rol.

RF39: Página de administrador.

Descripción	Proporcionar una página para los administradores.
Datos de Entrada	Correo electrónico y contraseña de administrador para iniciar sesión.
Datos de Salida	Interfaz para los administradores, que incluye listas con diferentes filtros de usuarios y caninos para manejar usuarios y caninos.

RF40: Búsqueda de direcciones.

Descripción	Proporcionar una barra de búsqueda de direcciones para facilitar la introducción de datos.
Datos de Entrada	Una cadena de caracteres que puede contener el nombre de la calle, la ciudad, el código postal...
Datos de Salida	Un listado de direcciones que coinciden con la búsqueda, al seleccionar una, se autocompletan los datos del formulario.

RF41: Cambiar a tema oscuro.

Descripción	Proporcionar una barra de búsqueda de direcciones para facilitar la introducción de datos.
Datos de Entrada	Ninguno.
Datos de Salida	Interfaz con el tema oscuro.

RF42: Subir post al blog.

Descripción	Permitir a un usuario subir un post al blog.
Datos de Entrada	Contenido que puede ser texto o imágenes.
Datos de Salida	Post publicado en el blog de la aplicación.

RF43: Comentar post del blog.

Descripción	Permitir a un usuario comentar un post en el blog.
Datos de Entrada	Texto que puede incluir emoticonos.
Datos de Salida	Comentario publicado en el post.

3.1.2. Requisitos no funcionales

Los requisitos no funcionales son los encargados de ofrecer al usuario una experiencia robusta y óptima a lo largo de la aplicación. Para el desarrollo de la aplicación se han tenido en cuenta los siguientes puntos:

- **Rendimiento:** La aplicación debe ser capaz de responder rápidamente a las interacciones de los usuarios y ejecutar las operaciones en un breve período de tiempo.
 - Se propone una deadline de 3 segundos como tiempo óptimo para cargar una página completa.

- El tiempo de respuesta del servidor se espera que sea menor de 2 segundos.
 - Se espera que la aplicación sea capaz de manejar múltiples peticiones sin que se produzca una degradación perceptible del rendimiento. Se propone como deadline 10000 de solicitudes.
 - Se espera que la base de datos realice consultas simples y complejas en menos de un segundo.
 - Se propone que en el caso de búsquedas y filtrados el tiempo de procesamiento puede ser mayor de 3 segundos, pero menor de 6.
 - La aplicación debe poder manejar alrededor de 10000 usuarios de forma concurrente sin sufrir degradación de rendimiento.
- **Escalabilidad:** Es necesario que a la hora de implementar código, se mantenga un código en el que sea sencillo añadir o quitar funcionalidades sin comprometer el comportamiento de la aplicación.
- Los componentes de la aplicación deben estar claramente definidos y documentados.
 - Un componente base debe recoger todas las propiedades comunes que pueden tener ese tipo de componentes. Por ejemplo, un componente base para los botones, deberá recoger el tamaño de la fuente, la forma del botón, colores etc.
 - Las funcionalidades desarrolladas deben ser fácilmente deprecadas si es necesario, es decir, no deben depender unas de otras si no es estrictamente necesario.
 - Los nuevos componentes que se desarrolle no deben afectar el funcionamiento de la aplicación.
 - Se espera que si fuera necesario actualizar algún componente en la UI, actualizando el componente base se actualicen todos los componentes de ese tipo.
- **Usabilidad:** Para esta aplicación, que tiene un rango de usuarios muy amplio, se espera que un usuario poco experimentado sea capaz de usar la aplicación de forma sencilla. La aplicación debe proporcionar una interfaz intuitiva y atractiva.
- La interfaz debe ser limpia y organizada.
 - Los colores de la aplicación deben ser coherentes.
 - La navegación de la aplicación debe cumplir la norma de los tres clicks, "*cualquier persona que visite la aplicación debería alcanzar la información más crítica como máximo en tres clics*".
 - La aplicación deberá ser lo suficientemente intuitiva o proporcionar guía de uso si es necesario.
 - Se espera que la aplicación sea testeada por usuarios reales antes del

primer despliegue para posibles correcciones.

- **Confiabilidad:** Para la aplicación, se espera que haya un correcto manejo de errores.
 - Se espera que un usuario sea informado con un mensaje adecuado si ocurre un error crítico.
 - Durante el desarrollo es necesario testear para poder añadir mecanismos que minimicen los errores conocidos en la aplicación.
 - Las copias de seguridad son indispensables para garantizar que no se pierden los datos ante un error crítico.
 - Se propone añadir mecanismos de manejo de errores en todas las operaciones críticas (lectura y escritura) para evitar errores críticos.
- **Seguridad:** Para garantizar este requisito, la aplicación debe proteger los datos de los usuarios y prevenir accesos no autorizados. También, en este caso en concreto, es necesario garantizar en medida de lo posible, que toda la información que se dé de alta, sea real y fehaciente.
 - Las contraseñas o datos sensibles no pueden estar expuestos, se deben almacenar con cifrados.
 - La aplicación no va a permitir el acceso a usuarios que no tengan una cuenta.
 - La aplicación no va a permitir dar de alta a perros a las protectoras que no estén verificadas por administradores, para evitar casos fraudulentos.
 - Un administrador tiene acceso a diferentes listas en las que puede verificar información y tiene la libertad de eliminar lo que crea correspondiente.
 - Debe prevenirse con mecanismos para ataques comunes como pueden ser las inyecciones SQL, CSS, Y CSRF.
- **Documentación:** A lo largo de todas las etapas, se almacenará toda la información relevante, ya sea en el código, en documentos aparte o en la propia memoria.

3.2. Diagramas de casos de uso

A continuación se incluyen los diagramas de caso de uso definidos para cada uno de los roles.

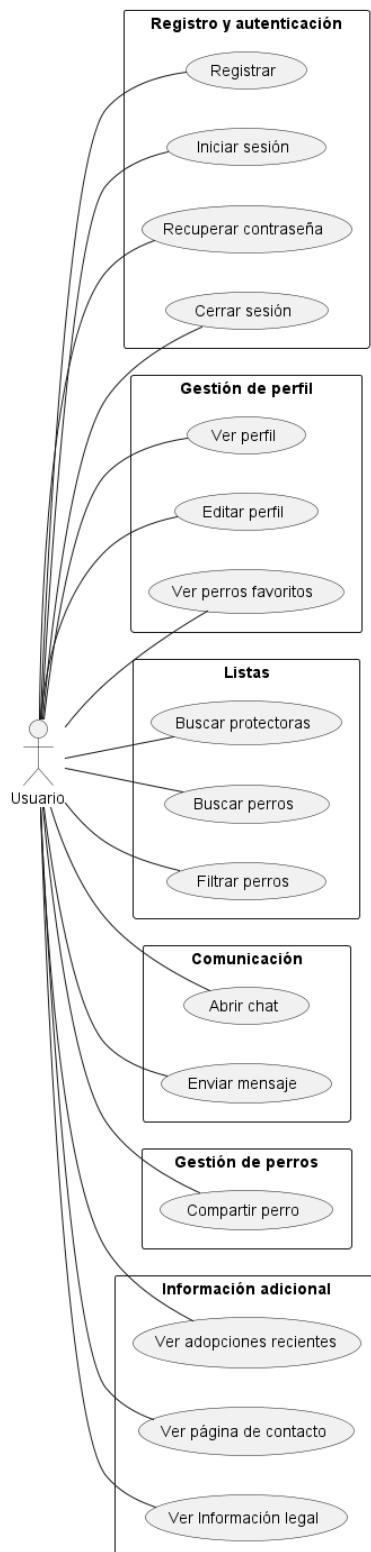


Figura 2: Casos de uso: Usuario.

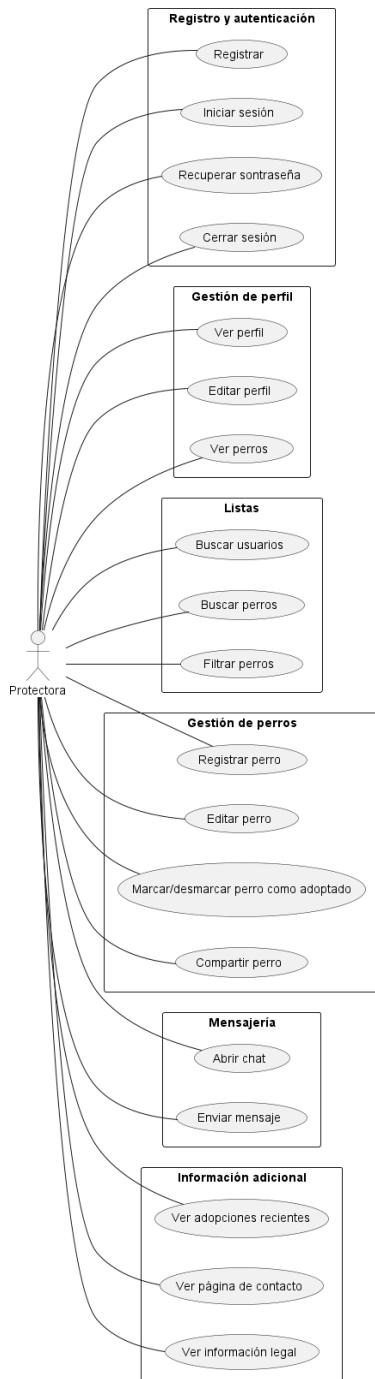


Figura 3: Casos de uso: Protectora.

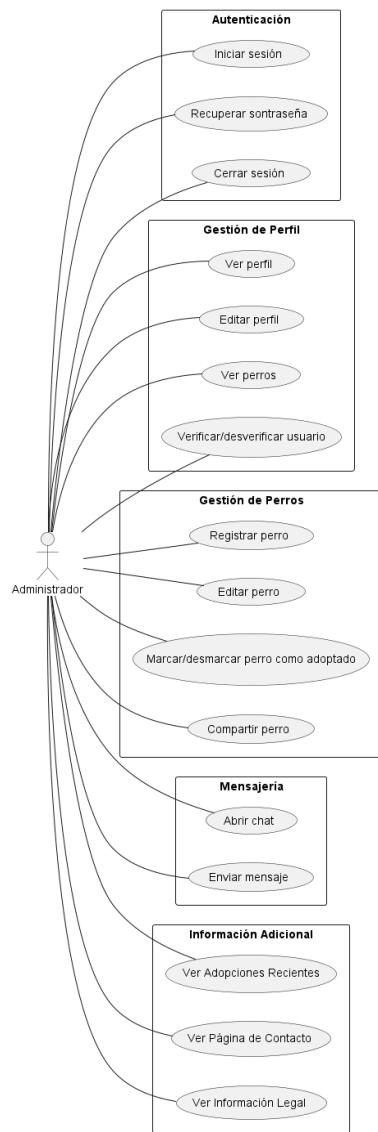


Figura 4: Casos de uso: Administrador.

3.3. Recursos

En esta sección se definen todos los recursos necesarios para el desarrollo de la aplicación.

3.3.1. Recursos humanos

En este proyecto, solo va a intervenir una persona, la autora de la memoria. Se trata de una Ingeniera de Software (semi-senior). Esta persona es encargada de cumplimentar todos los aspectos del desarrollo. El horario laboral será de Lunes a Viernes de media jornada.

3.3.2. Hardware

Para este proyecto, el hardware es un ordenador de sobremesa personal y una tablet android.

- Ordenador de sobremesa, para el desarrollo y las pruebas, con los siguientes componentes:
 - **Procesador:** AMD Ryzen 7 3700X 8-Core Processor - 3.60 GHz
 - **Memoria RAM:** DDR4 3000 2x16GB
 - **Tarjeta gráfica:** NVIDIA GeForce GTX 1660 SUPER
 - **Placa base:** ASUS TUF GAMING B550-PLUS WIFI II
 - **Disipador:** Noctua NH-D15
- Tablet android Hi9plus, para realizar pruebas en un dispositivo android físico.

3.3.3. Software

El sistema operativo utilizado es el que estaba instalado previamente en el ordenador personal.

Para escoger el framework que se va a utilizar en el proyecto partimos de la base de que vamos a realizar una aplicación android. Existen diferentes frameworks como *React Native*, *Flutter*, *Kotlin Multiplatform* etc. que ofrecen integración con android para facilitar el desarrollo. En este proyecto, se ha optado por desarrollar con Flutter, debido a que se ha trabajado previamente con él. Este framework, además, permite realizar un desarrollo multiplataforma y ofrece un alto rendimiento.

El lenguaje utilizado es *Dart*, este lenguaje es muy similar a Java, se trata de un lenguaje orientado a objetos y se utiliza principalmente para el desarrollo de aplicaciones del lado del cliente. Es un lenguaje bastante popularizado, uno de los motivos para optar por este lenguaje es el mantenimiento que recibe a día de hoy

y por la simple que resulta para programar. Para añadir funcionalidades extra, se han escogido varias bibliotecas que ofrecen algunos microservicios o widgets.

Por último, para manejar la base de datos, se ha optado por usar *Firebase*. Se trata de una plataforma de desarrollo de aplicaciones web muy popularizada, también desarrollada por Google. Ofrece diferentes servicios para la autenticación, el hosting de la aplicación y bases de datos en tiempo real, eso lo convertía en el candidato adecuado para manejar el chat, la funcionalidad de compartir y otras cosas funcionalidades que nuestra aplicación requiere.

- **Sistema Operativo:** Windows 11 x64
- **IDE:** Android Studio Hedgehog — 2023.1.1 Patch 2
- **Framework:** Flutter 3.19.0 - [Flutter DEV](#) [4]
- **Lenguaje:** Dart 3.3.0 - [Dart DEV](#) [5]
- **Paquetes:**
 - `cupertino_icons: ^1.0.2`
 - `firebase_core: ^2.24.2`
 - `firebase_database: ^10.3.8`
 - `firebase_storage:`
 - `cloud_firestore: ^4.13.6`
 - `firebase_auth: ^4.15.3`
 - `responsive_sizer: ^3.3.0+1`
 - `flutter_login: ^5.0.0`
 - `google_fonts: ^4.0.4`
 - `resize: ^1.0.0`
 - `awesone_notifications: ^0.9.2`
 - `csc_picker: ^0.2.7`
 - `map_address_picker: ^0.3.5`
 - `geocoding: ^3.0.0`
 - `search_map_location: 0.0.6`
 - `image_picker: ^1.0.7`
 - `image_cropper: ^4.0.1`
 - `path_provider: ^2.1.2`
 - `flutter_image_compress: ^2.1.0`
 - `address_form: ^0.0.2`
 - `address: ^0.1.0+2`
 - `google_maps_webservice: ^0.0.20-nullsafety.5`
 - `meta_validator: ^0.0.2`
 - `geolocator: ^9.0.2`
 - `dropdown_search: ^5.0.6`
 - `card_swiper: ^3.0.1`

- filter_list: ^1.0.2
 - flutter_filter_dialog: ^1.2.0
 - choice: ^2.3.2
 - animate_gradient: ^0.0.2+1
 - firebase_messaging: ^14.9.1
 - flutter_chat_bubble: ^2.0.2
 - chat_bubbles: ^1.6.0
 - share_plus: ^9.0.0
 - app_links: ^6.0.1
 - url_launcher: ^6.2.6
 - firebase_dynamic_links: ^5.5.4
 - go_router: ^14.0.2
 - get: ^4.6.6
 - linkwell: ^2.0.6
 - toastification: ^1.0.0
- **Base de datos:** Firebase database - [Firebase console](#) [6]

3.4. Costes

Habiendo escogido todos los recursos necesarios para el desarrollo del producto, se pueden definir un presupuesto que englobe todos los apartados.

3.4.1. Recursos humanos

Para calcular el costo relacionado con el personal, se toma como referencia el sueldo medio de un programador junior, que ronda los *24.000€* anuales, lo que equivale a *12.000€* a media jornada. Haciendo cálculos, al mes corresponden unos *1.000€*. Dado que la duración del proyecto es de 5 meses y medio, el costo total en salarios para el empleado será de *5.500€*.

3.4.2. Hardware

Debido a que el equipo utilizado no es nuevo, se consideran los gastos del equipo teniendo en cuenta que un equipo informático se puede amortizar hasta 8 años. El precio aproximado del ordenador es de unos *1000€*. Un año tiene 12 meses, lo que resulta en 96 meses en total de amortización, lo que implica que cuesta unos *11€* al mes. Dado que la duración del proyecto es de 5 meses y medio, el costo del ordenador sería de *60,50€* en total. El precio de la tablet es de unos *100€*, así que utilizando los mismos cálculos, el costo de la tablet es de *5,72€* en total.

3.4.3. Software

Para el alcance de este proyecto y los números a los que se aspira, al menos durante la primera fase, se ha optado por utilizar las versiones gratuitas del software escogido, lo que resulta en un coste total de 0€ en software.

- Firebase - Plan Spark - [Guía de planes \[7\]](#)
- Google APIs - Ofrecen créditos gratuitos de hasta 200€ para cubrir los gastos que se generen por el uso de la API. - [Precios\[8\]](#)
- IDE + Framework - Gratuitos

3.4.4. Otros

Debido a que el desarrollo del proyecto se realiza en la casa de la persona empleada, se añaden a los costes la electricidad usada. Usando como referencia el gasto mensual de electricidad, que ronda los 30€.

Recurso	Coste (€)
Personal	5.500
Hardware	Ordenador - 60,50 Tablet - 5,72
Software	0
Electricidad	180
Total	5.746,22€

Costes del proyecto

4. Diseño

En esta parte se incluye toda la estructura de la aplicación, tanto como la interfaz de usuario, el backend y la base de datos.

4.1. Diseño de base de datos

Esta sección es primordial para generar unas bases sólidas para la aplicación. Se deben almacenar todas las entidades y todas sus propiedades, definiendo claramente el tipo de cada una.

4.1.1. Direcciones

Para todas las protectoras, es necesario almacenar la dirección en la que están ubicadas. Esta información es crucial para mostrar a los usuarios la ubicación de cada protectora y generar marcadores en el mapa correspondiente. Por un lado, se almacenan todas las partes de la dirección, como la ciudad, la calle, el código postal, etc. Por otro lado, cuando un usuario introduzca su dirección, se generarán automáticamente las coordenadas de esa dirección para almacenarlas también, evitando tener que calcularlas cada vez que se carguen los mapas en la interfaz. Asimismo, cuando un usuario actualice su dirección, se actualizarán también sus coordenadas correspondientes en la base de datos.

Para todas las direcciones se genera una entrada en el que su ID corresponde con el del usuario al que pertenece.

Tabla: addresses

Campo	Tipo	Dato	Obligatorio
city	cadena	Ciudad de la dirección	Si
country	cadena	País de la dirección (España siempre)	Si
province	cadena	Provincia de la dirección	Si
street	cadena	Calle de la dirección	Si
street_number	cadena	Número de la calle	No
zipcode	cadena	Código postal	Si
lat	número	Latitud	Si
lng	número	Longitud	Si

Aquí se muestra un ejemplo real de una dirección almacenada en la base de datos.

```
{
  "city": "Murcia",
  "country": "España",
  "lat": 37.9879153,
  "lng": -1.1315578,
  "province": "Murcia",
  "street": "Calle Maestro Alonso",
  "street_number": "4",
  "zipcode": "30005"
}
```

4.1.2. Chats

Para la tabla de chats, se ha decidido almacenar los IDs de los usuarios implicados, aparte se almacena un array de IDs que indica para quien está disponible el chat en ese momento. Al igual que en otras aplicaciones de mensajería, cuando se elimina un chat, éste simplemente desaparece para el usuario que lo borra esto hace que la propiedad de quién puede acceder al chat en ese momento sea modificada. En esta primera iteración del proyecto, los mensajes no se borran. Por ejemplo, si una persona A borra su chat con la persona B y luego vuelve a abrir un chat con esa misma persona, los mensajes anteriores seguirán estando disponibles. En la fase de implementación, se profundizará más en el comportamiento definido para este aspecto.

Almacenar aquí los IDs de los usuarios implicados, permite que si se proponen chats de grupo, sea más sencillo escalarlo, ya que en esta forma la base de datos ya lo está soportando. Lo mismo ocurre con la propiedad que hace referencia a que usuarios pueden ver el chat.

Para todos los chats se genera un ID automático, al que se hace referencia desde la tabla de usuarios.

Tabla: chats

Campo	Tipo	Dato	Obligatorio
availableTo	array	IDs de Usuarios que tienen el chat visible en su página	Si
lastMessage	cadena	ID del último mensaje mandado en el chat	Si
users	array	IDs de Usuarios que intervienen en el chat	Si

Aquí se muestra un ejemplo real de un chat almacenado en la base de datos.

```
{  
  "availableTo": [  
    "kuaKjc6XoiVot1rMeGqBySA5pE13",  
    "bu04Vkazs302oVKetugxrvTfRw12"  
,  
  "lastMessage": "BaKmuuo8debUTdSNsvw0",  
  "users": [  
    "kuaKjc6XoiVot1rMeGqBySA5pE13",  
    "bu04Vkazs302oVKetugxrvTfRw12"  
,  
  ]  
}
```

4.1.3. Perros

Esta entidad es una de las más importantes de la aplicación, ya que es la que se principal afectada por la mayoría de funcionalidades que se proponen. Principalmente esta tabla contendrá todas las propiedades de los perros que se van a manejar dentro de la app, aunque hay algunas consideraciones importantes para algunas de ellas.

Para el género se han propuesto dos valores fijos que luego se transformaran en la cadena correspondiente cuando se carguen en la UI. Los valores son **male** y **female**. Se propone esta solución para cuando se decida añadir traducciones en la aplicación y no trabajar con palabras en español en la base de datos.

Para almacenar la edad, se ha optado por almacenarla junto a las unidades, ya que así se le da la posibilidad al usuario a especificar mejor la edad del perro (el caso de uso es muy concreto, pero añade valor). Los valores de las unidades serán fijos, pudiendo ser:

- Meses
- Años

Lo mismo ocurre con el peso, se opta por almacenar el peso junto con las unidades para facilitar al usuario especificar el peso. A pesar de que se pueda especificar las unidades del peso, el peso en la base de datos siempre será tratado como KG, para facilitar el uso de los filtros, cuando se traiga a la UI, se actualizará al valor real que corresponda. Las unidades disponibles serán

- Gramos
- KG

En una primera instancia se propuso almacenar los perros favoritos en la tabla de usuarios, pero tras hacer algunas pruebas se optó por este modelo finalmente. Almacenar los usuarios que han marcado al perro como favorito, facilita las actualizaciones en tiempo real de los resultados de perros.

Para todos los perros se genera un ID automático.

Tabla: dogs

Campo	Tipo	Dato	Obligatorio
adopted	booleano	Indica si el perro ha sido adoptado.	Si
age	número	Edad	Si
ageUds	cadena	Unidades de la edad	Si
breed	cadena	Raza	Si
castrated	booleano	Indica si está castrado	Si
color	cadena	Color	Si
created	marca de tiempo	Fecha y hora de cuando se ha dado de alta	Si
description	cadena	Descripción	No
favoriteBy	array	IDs de usuarios que lo han marcado como favorito	No
forAdoption	booleano	Indica si está disponible para adopción	No
forFoster	booleano	Indica si está disponible para acogida	No
gender	cadena	Género	Si
name	cadena	Nombre	Si
ownerId	cadena	ID del propietario	Si
profilePic	cadena	URL de la imagen de perfil de la mascota	No
weight	número	Peso	Si
weightUds	cadena	Unidades de peso	Si

Aquí se muestra un ejemplo real de un perro almacenado en la base de datos.

```
{
  "adopted": false,
  "age": 1,
  "ageUds": "Años",
  "breed": "Golden Retriever",
  "castrated": false,
  "color": "Golden",
  "created": "2023-01-01T12:00:00Z",
  "description": "Un perro amable y sociable.",
  "favoriteBy": [
    "user1",
    "user2"
  ],
  "forAdoption": true,
  "forFoster": false,
  "gender": "Male",
  "name": "Luna",
  "ownerId": "user1",
  "profilePic": "https://example.com/pic/luna.jpg",
  "weight": 10,
  "weightUds": "kg"
}
```

```

    "breed": "Breed 2",
    "castrated": true,
    "color": "Color 1",
    "created": "23 de abril de 2024, 10:56:48 a.m. UTC+2",
    "description": "",
    "favoriteBy": [],
    "forAdoption": true,
    "forFoster": true,
    "gender": "female",
    "name": "Katrina",
    "ownerId": "S4wXwckahiSOLI0Q3exXDVRh2y12",
    "profilePic": "",
    "weight": 1,
    "weightUds": "KG"
}

```

4.1.4. Mensajes

A parte de almacenar toda la información de los chats, hay que almacenar los mensajes que se mandan en los chats. Aparte de guardar el contenido del mensaje, se propone hacer una diferenciación por tipos, habiendo en esta primera iteración solo dos tipos, que son: **default** e **image**, indicando que un mensaje es texto o imagen respectivamente. Esta definición de tipos ayudará a saber como cargar el contenido en la interfaz.

Para todos los mensajes se genera un ID automático que se puede referenciar desde la tabla de chats.

Tabla: messages

Campo	Tipo	Dato	Obligatorio
chatId	cadena	ID del chat	Si
from	cadena	ID del emisor del mensaje	Si
messageContent	cadena	Contenido del mensaje	Si
read	booleano	Indica si el receptor a leído el mensaje	Si
sent	marca de tiempo	Fecha y hora en la que se mandó el mensaje	Si
to	cadena	ID del receptor del mensaje	Si
type	cadena	Tipo del mensaje	Si

Aquí se muestra un ejemplo real de una dirección almacenada en la base de datos.

```
{  
  "chatId": "Sm1Ji0chb9uhofa20VW5",  
  "from": "kuaKjc6XoiVot1rMeGqBySA5pE13",  
  "messageContent": "Hola, ¿estás?",  
  "read": true,  
  "sent": "6 de mayo de 2024, 1:57:30 a.m. UTC+2",  
  "to": "bu04Vkaza302oVKetugxrvTfRw12",  
  "type": "default"  
}
```

4.1.5. Notificaciones

Cuando un usuario envíe un mensaje, el usuario receptor debe recibir una notificación. Para gestionar las notificaciones de los mensajes, se ha decidido crear una tabla que contenga las solicitudes de notificación para un usuario específico. Al iniciar sesión, se agrega un listener que escucha todos los cambios en esta tabla y cuando llega una nueva solicitud, se envía una notificación a través del canal correspondiente. Cuando el usuario receptor recibe la notificación, esta solicitud se marca como vista.

Para cada una de las entradas su ID corresponde con el del usuario al que pertenece.

Tabla: notifications

Campo	Tipo	Dato	Obligatorio
requests	collection	Colección de peticiones	Si

Para cada petición se genera un ID y se genera en un formato en concreto.

Tabla: requests

Campo	Tipo	Dato	Obligatorio
chats	array	IDs de los chats que pertenecen al usuario	Si
email	cadena	ID del chat desde el que se manda la notificación	Si
profilePic	cadena	URL de la imagen de foto de perfil	Si
rol	cadena	Rol del usuario	Si
username	cadena	Nombre de usuario	Si
verified	cadena	Indica si ha sido verificado por un administrador	Si, solo si el rol es company

Aquí se muestra una petición real almacenada en la base de datos.

```
{
  "body": "Hola, ¿cómo estás?",
  "chatId": "Sm1Ji0chb9uhofa20VW5",
  "new": false,
  "summary": "Hola, ¿cómo estás?",
  "title": "Laura Vega Palacios",
  "userId": "kuaKjc6XoiVot1rMeGqBySA5pE13"
}
```

4.1.6. Usuarios

Los usuarios son otra entidad de gran relevancia dentro de la aplicación. Aunque un usuario tiene bastantes datos dentro de la aplicación, muchos de estos datos se almacenan en forma de referencia con su ID correspondiente en otras tablas. Además, para manejar los registros de usuarios se utilizará el servicio de Autenticación que proporciona Firebase, que se encarga de facilitar las herramientas para registrar usuarios y permite almacenar tanto el correo electrónico como la contraseña, entre otras cosas.

Tabla: users

Campo	Tipo	Dato	Obligatorio
body	cadena	Contenido de la notificación	Si
chatId	cadena	ID del chat desde el que se manda la notificación	Si
new	booleano	Indica si la notificación se ha lanzado o no	Si
summary	cadena	Preview de la notificación	Si
title	cadena	Titulo de la notificacion	Si
userId	cadena	Usuario que manda la notificación	Si

Aqui se muestra un ejemplo de usuario guardado en la base de datos.

```
{
  "chats": [
    "wfmRPtHiGvUI9vi9Fs7k"
  ],
  "email": "protectora@test.com",
  "profilePic": "",
  "rol": "company",
  "username": "Protectora super chula",
  "verified": true
}
```

4.1.7. Imágenes

En cuanto al almacenamiento de imágenes, se utilizará el servicio que proporciona Firebase llamado *Storage*. Este servicio permite almacenar imágenes, vídeos o audios y viene integrado de manera nativa con varios servicios de Firebase como el servicio de *Authentication* mencionado anteriormente. Dentro del almacenamiento reservado para la app se propone organizarlo en los siguientes directorios:

- *images*
 - *chats*
 - *profilePics*

Las imágenes de perfil de usuarios o perros, se almacenarán en el directorio de *profilePics*, por otra parte, cuando se mande una imagen por chat, esta se almacenará en el directorio de *chats*. Para trabajar con las imágenes dentro de la app, se pueden obtener los enlaces que se generan cuando se van almacenando

dentro de *Storage*. Estas URLs que se obtienen se irán almacenadas en las tablas definidas anteriormente. Si una de las imágenes es borrada, se tiene que borrar su referencia en la tabla de la base de datos y la imagen almacenada en *Storage*.

4.2. Diseño de interfaz de usuario

El diseño de la interfaz de usuario (UI) se centra en crear una experiencia de usuario (UX) intuitiva y atractiva. Inicialmente, se define un estilo y tema que se debe mantener a lo largo de todos los bocetos o prototipos que se vayan generando a continuación. Es importante tener en cuenta que el diseño sea responsive para que se adapte adecuadamente a todos los dispositivos móviles.

4.2.1. Tema

Para los colores principales de la aplicación, se opta por escoger el color morado. El color morado dentro de UX, puede evocar diferentes sensaciones al usuario. Muchas marcas de lujo utilizan este color para sus logos, aunque también puede evocar sentimientos de calma o relajación según las tonalidades que se utilicen. Teniendo el morado como color principal, se añade a la paleta su color análogo, el azul. Para finalizar los colores principales, a la paleta se añade el color complementario del azul, que sería el naranja. Estos tres colores conformaran los elementos de la aplicación, el morado se utilizará para la mayoría de elementos, indicando que son elementos importantes. El azul, se mezclará con el morado en algunos elementos para hacer más dinámicos los fondos y utilizarlo en elementos no tan relevantes. Por último, el naranja, será un color que se utilice como acentos o llamar la atención del usuario, junto con el naranja se podrán utilizar algunos de sus colores análogos como el amarillo o el rojo. Para los fondos de la aplicación se añade también un blanco roto a la paleta de colores que se propone.

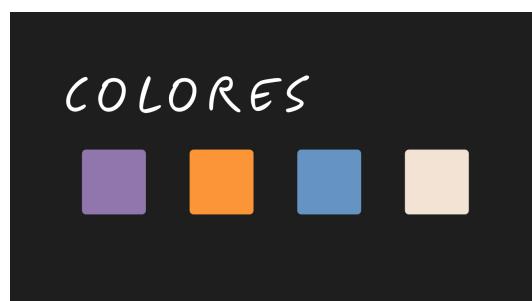


Figura 5: Paleta

Como fuente, se opta por escoger la fuente *Nunito Sans*, que proporciona

Google. Se escoge porque es una fuente bastante estándar y legible para cualquier tipo de usuario.

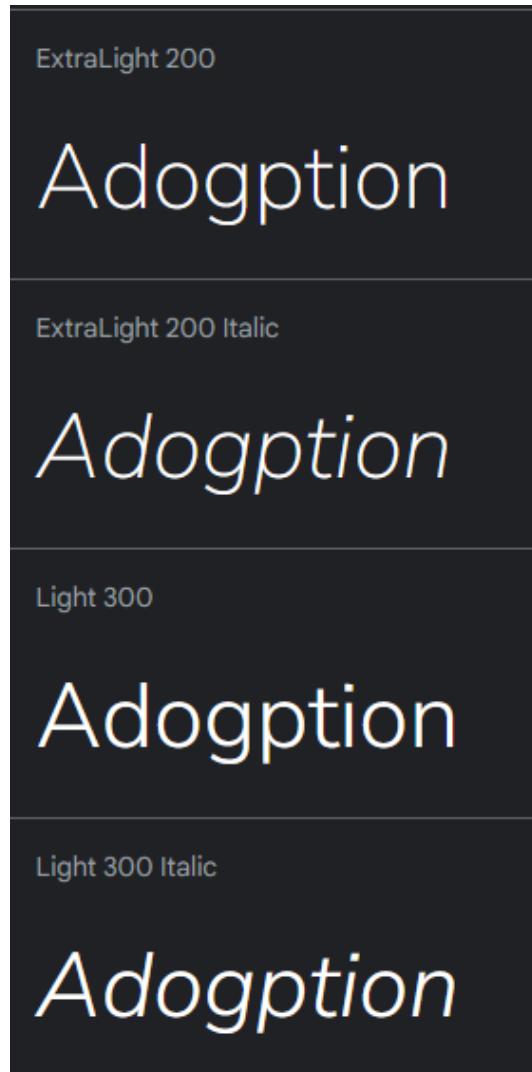


Figura 6: Fuente Nunito Sans

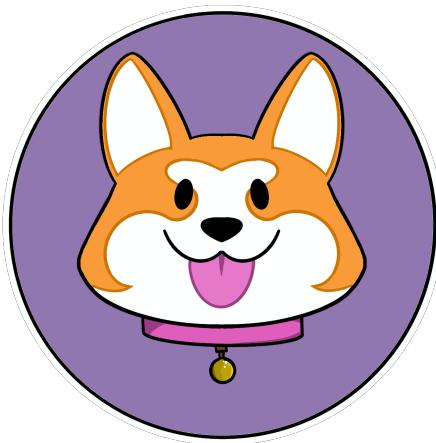
4.2.2. Nombre

Como nombre de la aplicación, se tiene en cuenta que inicialmente la aplicación está dirigida a la adopción de perros, así que se propone un juego de palabras. Uniendo las palabras *Adoption* y *Dog*, que son la traducción de las palabras adopción y perro, conformando el nombre: ***Adogption***.

4.2.3. Logo e iconos

La última parte relacionada con el branding en este proyecto es la de generación de logos e iconos. En esta sección se incluye tanto el logo de la aplicación como los iconos que se han hecho para incluir dentro de la aplicación. Algunos de los iconos podrían finalmente no utilizarse.

La idea del logo es que se perciba claramente que la aplicación va sobre perros. Los corgis son unas de las razas más populares en el mundo, destacan por que es una raza muy amigable y muy querida dentro de la cultura pop, lo que la convierte en la cara perfecta para llamar la atención. Además del logo principal, se ha diseñado un logo animado para incluir en pantallas de carga u otros lugar. Por último, se han hecho algunos iconos por defecto, que serán utilizados en perfiles o listas.



(a) Logo



(b) Logo animado

Figura 7: Logos

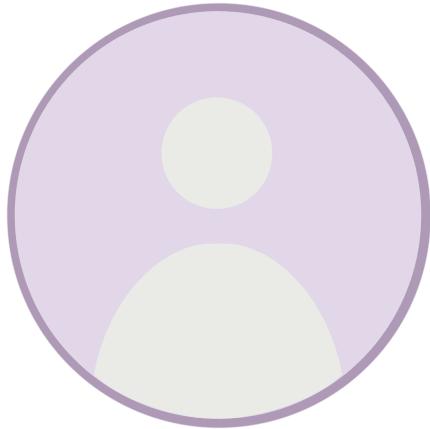


Figura 8: Icono por defecto para usuario



Figura 9: Icono por defecto para perro



Figura 10: Icono para listas de adopción



Figura 11: Icono para listas de acogida

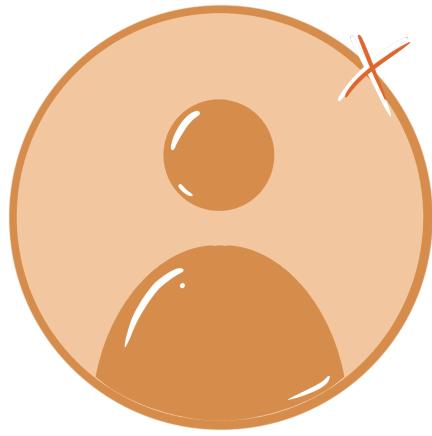


Figura 12: Icono usuario no verificado



Figura 13: Icono usuario verificado

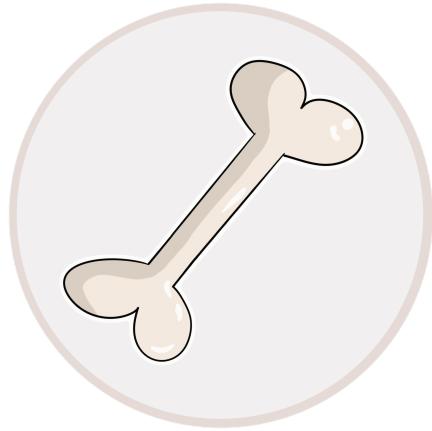
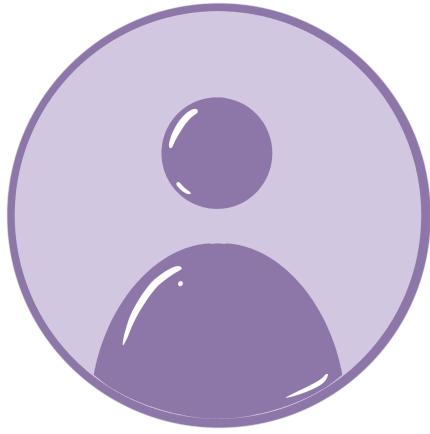


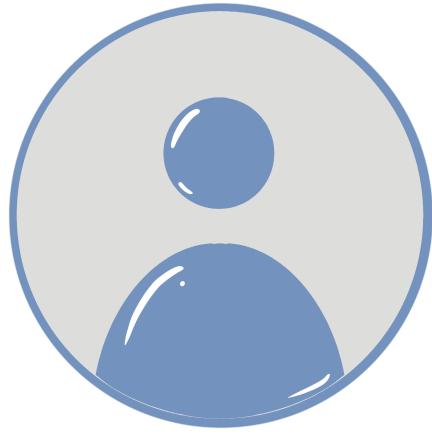
Figura 14: Icono hueso



Figura 15: Icono casa



(a) Icono en morado



(b) Icono en azul

Figura 16: Iconos usuarios



(a) Icono en naranja



(b) Icono en morado



(c) Icono en azul

Figura 17: Iconos patas

4.2.4. Diseños

En esta sección se incluyen todos los diseños realizados en *Miró* para las pantallas de la aplicación. Entre los diseños se encuentran todas las páginas principales de la aplicación y componentes relevantes. En la sección de implementación se especificará la función de cada una de las pantallas.



Iniciar sesión

Correo electrónico

Contraseña

Iniciar Sesión

Registrarse

[¿Has olvidado la contraseña?](#)

Figura 18: Pantalla de carga.

Figura 19: Inicio de sesión.

Datos personales

Nombre completo
Correo electrónico
Confirma el correo electrónico
Contraseña
Confirma contraseña

Confirmar

Figura 20: Registro/edición de usuario.

Dirección

Search
País
España CP
Provincia
Ciudad
Calle
Información adicional

Confirmar

Figura 21: Registro/edición de protectora.



Figura 22: Inicio usuario.



Figura 23: Inicio protectora.

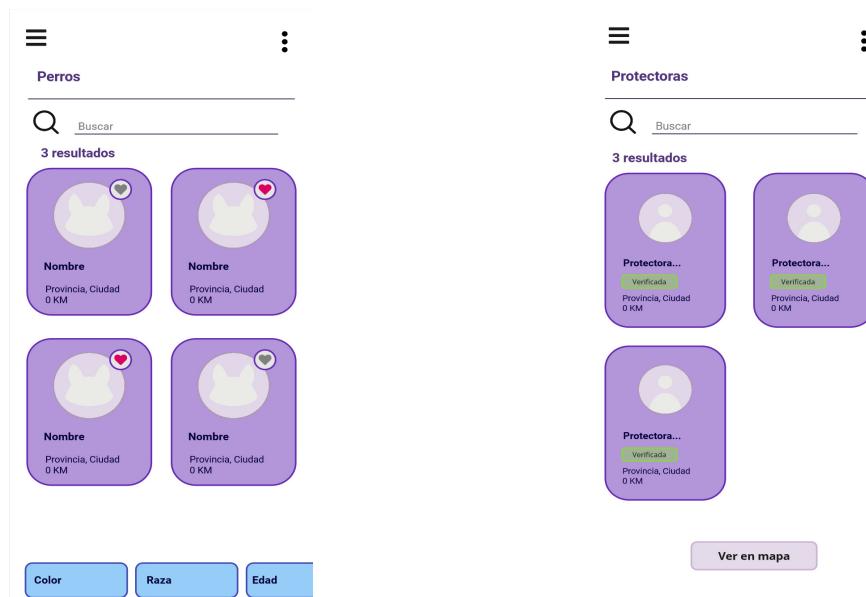


Figura 24: Lista de perros.

Figura 25: Lista de usuarios.

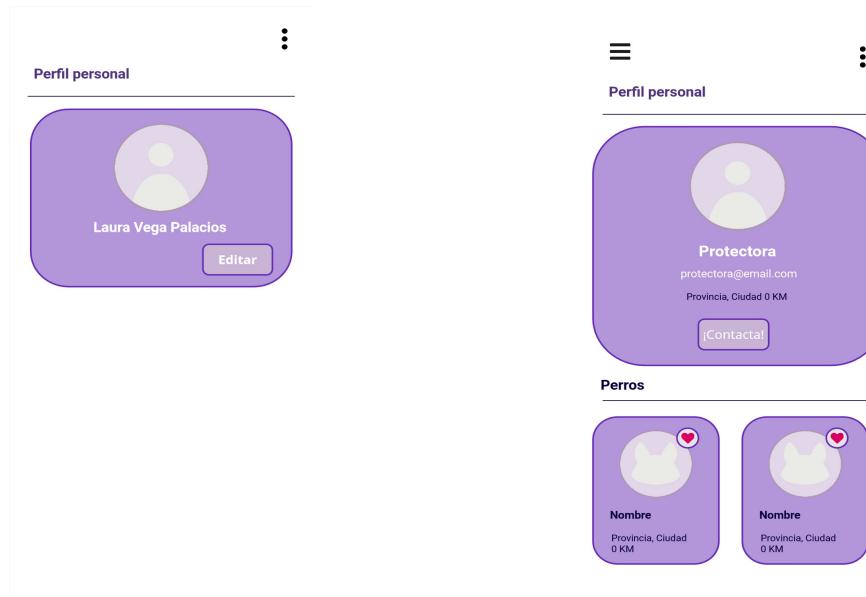


Figura 26: Perfil de usuario/administrador.

Figura 27: Perfil de protectora.

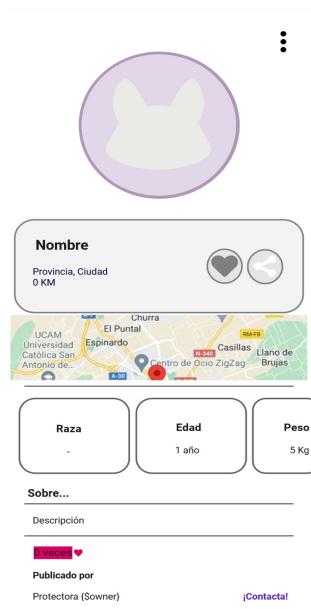


Figura 28: Perfil de perro.

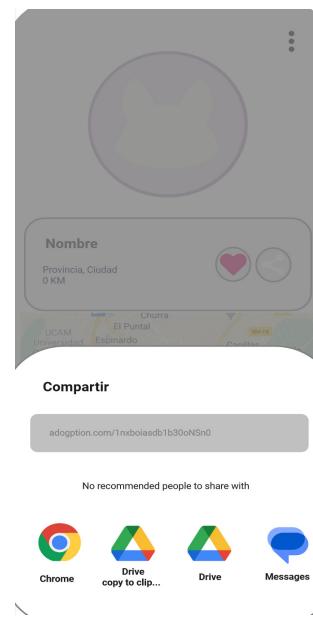


Figura 29: Compartir perro.



Figura 30: Página de mapa y lista.

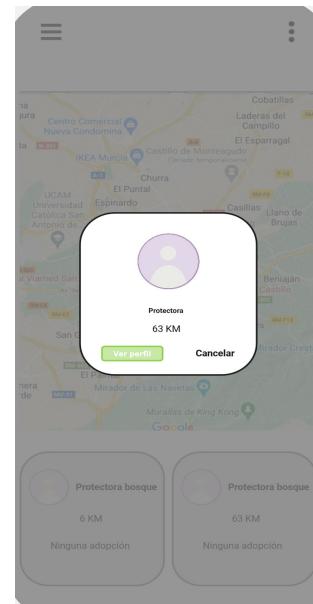


Figura 31: Acción de redirigir a perfil.



Figura 32: Registro/edición canino.



Figura 33: Menú lateral.



Figura 34: Inicio administrador.



Figura 35: Pop-up verificación.

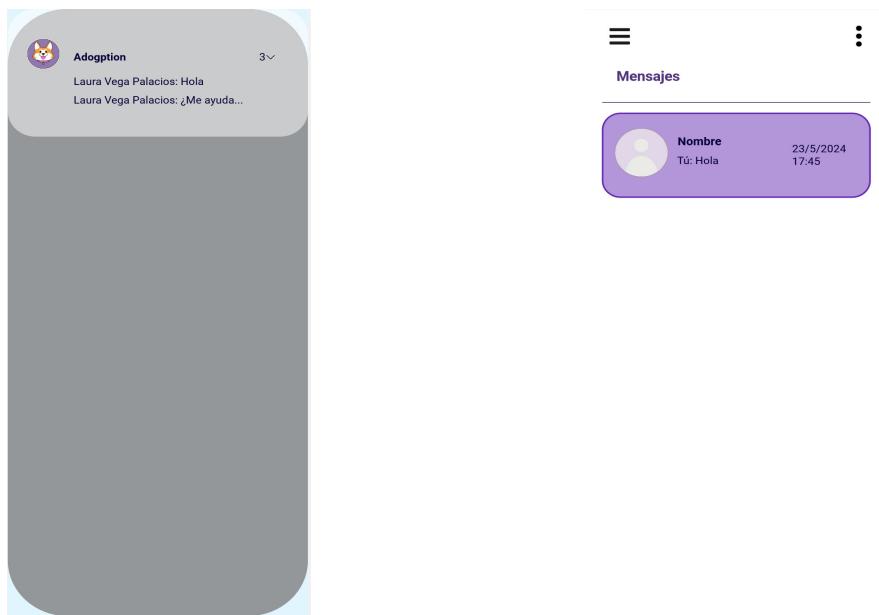


Figura 36: Notificaciones.

Figura 37: Página de mensajes.



Figura 38: Recuperar contraseña.

5. Implementación

En esta sección se incluyen las clases y pantallas finales, que son resultado del desarrollo y de la posterior corrección de errores.

5.1. Módulos y clases

Durante el desarrollo se han implementado diferentes módulos, cada uno con una finalidad específica. Mantener los componentes granulados en módulos y submódulos asegura una mejor escalabilidad y mantenibilidad de la aplicación y facilita tareas como refactorizaciones y cambios de estilos, además de evitar muchas repeticiones de código.

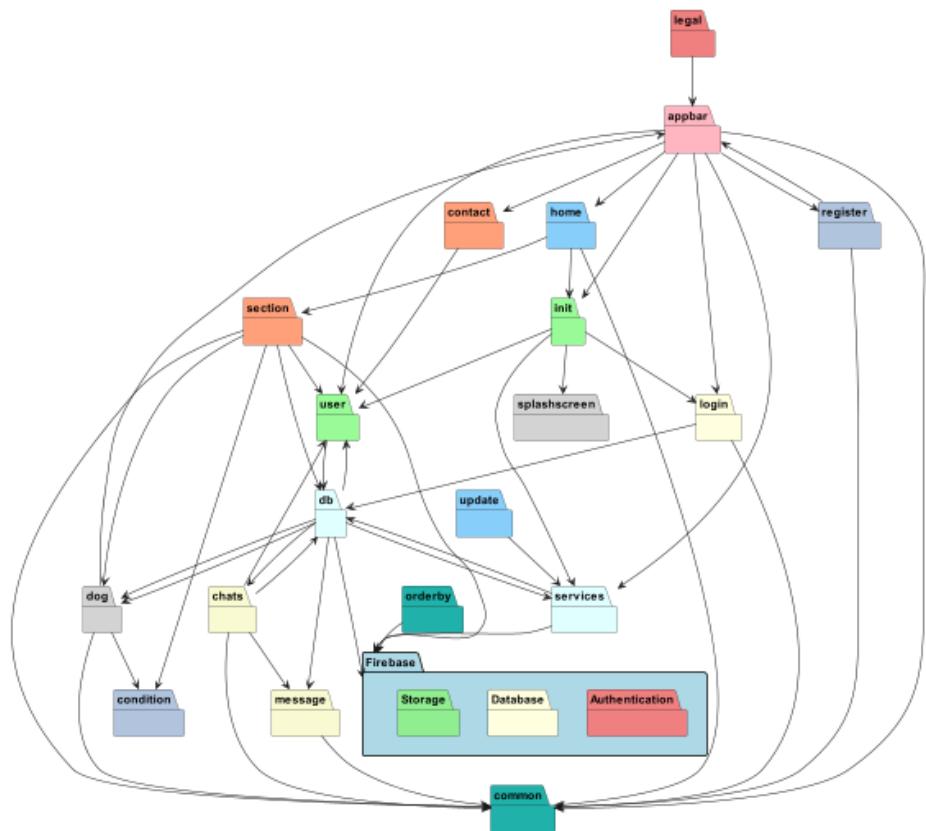


Figura 39: Módulos y sus dependencias.

appbar

Este módulo se utiliza para generar las diferentes *AppBar* que se utilizan en las pantallas de la aplicación. Una *AppBar* es el componente que se coloca en la parte superior de la pantalla y puede contener botones u otros componentes tales como barras de búsqueda o imágenes. La única clase que contiene el módulo se encarga de instanciar el componente en las diferentes pantallas según el tipo de barra que se necesite.

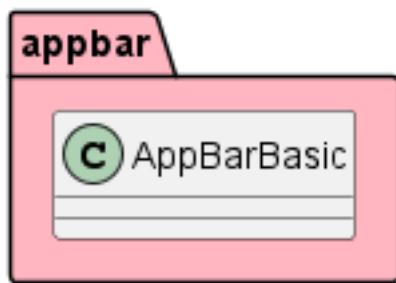


Figura 40: Módulo *appbar*.

chats

Aquí se incluyen todas las clases que necesita un chat para funcionar correctamente.

- **ChatInput:** Componente que permite a los usuarios escribir mensajes y adjuntar imágenes a los chats.
- **ChatRoomPage:** Interfaz del chat.
- **ChatsPage:** Es la página principal que contiene todos los chats, contiene todos los chats que un usuario tiene en ese momento, son una lista de *ChatWidget*.
- **ChatWidget:** Este componente muestra una previsualización del chat, que incluye la información del usuario, el último mensaje y la fecha y hora en la que se envió. Además, incluye el número de mensajes sin leer del chat.
- **ChatModel:** Clase que se encarga de generar y manejar modelos de chat que se generan a partir de toda la información de la base de datos relacionada con los chats.

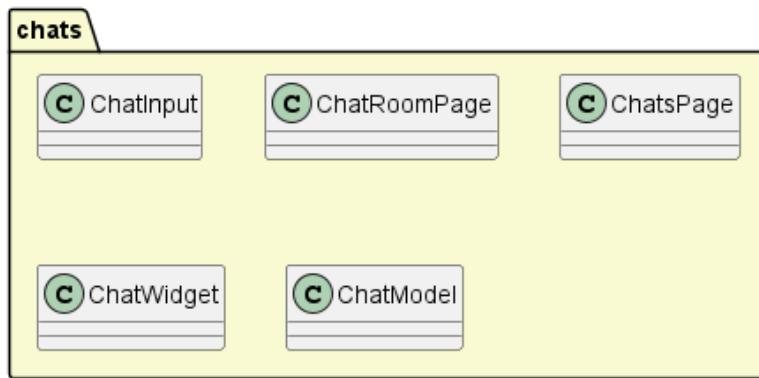


Figura 41: Módulo *chats*.

common

En este módulo se definen todos los componentes básicos que incluyen las diferentes pantallas generadas. Además, se definen diferentes submódulos para cada uno de los componentes creados. Todos los submódulos contienen una clase base en la que se definen los estilos y funcionalidades básicas, y el resto de las clases extienden estas clases base. Es importante destacar que, aunque se hayan creado clases propias para los componentes, todas ellas instancian componentes de las bibliotecas por defecto de Flutter, como la biblioteca *material*.

- **button:** Define todos los botones que se utilizan en diferentes componentes y pantallas. Existen botones que tienen diferentes funcionalidades, como redirigir a otras pantallas, actualizar datos o mostrar pop-ups.
- **card:** Una tarjeta es un componente muy parecido a un contenedor pero con un visual más atractivo. Por ello se ha generado un componente base que se reutiliza en diferentes widgets y que contiene este módulo.
- **checkbox:** Recoge los diferentes checkboxes que se han desarrollado, además de su comportamiento.
- **container:** En este módulo se define un contenedor básico con un estilo concreto para poder reutilizarlo en diferentes sitios.
- **dialog:** Recoge todos los pop-ups que se han desarrollado y todo su comportamiento.
- **drawer:** Este módulo contiene todos los menús laterales y componentes de estos. Los componentes de los menús incluyen imágenes de perfil y botones que redirigen a otras pantallas.
- **dropdown:** Incluye los diferentes dropdowns que se han utilizado en la app. Algunos de ellos incluyen la funcionalidad de búsqueda de opciones.
- **form:** Incluye todos los formularios que se utilizan en la aplicación.

- **icon:** Recoge iconos básicos que tienen funcionalidad. Por ahora, el único que se incluye es el ícono de favoritos.
- **images:** Este módulo incluye clases muy diferentes. Por una parte, incluye clases que proveen funcionalidades para seleccionar imágenes del dispositivo y, por otra parte, incluye componentes que facilitan la inserción de imágenes en la app.
- **input:** Recoge todos los componentes que permiten a un usuario introducir información de algún tipo. Pueden ser de texto, numéricos o incluso componentes compuestos que contienen dropdowns o checkboxes.
- **maps:** Incluye toda la funcionalidad relacionada con los mapas, tanto la página de mapas como los mapas que se pueden insertar en diferentes partes de la aplicación.
- **padding:** En este submódulo se define un padding que se reutiliza a lo largo de la aplicación.
- **picker:** Incluye componentes relacionados con seleccionar direcciones en mapas o similares.
- **searchbar:** Define las barras de búsqueda implementadas en la app, además de su funcionalidad.
- **swiper:** En este submódulo, por ahora, solo se ha definido el swiper para mostrar las adopciones recientes.
- **text:** Contiene clases que definen textos que se usan a lo largo de la aplicación. Los textos pueden ser títulos, subtítulos, avisos, etc.
- **widget:** Incluye componentes que son comunes pero están compuestos por otros componentes comunes.



Figura 42: Módulo *common*.

condition

Este módulo el encargado de manejar los filtros de las listas de una forma más dinámica.

- **Basic:** Define el componente básico para añadir filtros a las listas, se encarga de abrir el pop-up con las opciones del filtro.
- **Gender:** Define el filtro para el género.
- **Weight:** Define el filtro para el peso.
- **ConditionModel:** Clase que se encarga de generar y manejar modelos de condiciones a partir de la información proporcionada por la aplicación o el

usuario. Esta clase además proporciona un método que se encarga de añadir todos los filtros que haya aplicados en ese momento a la query.

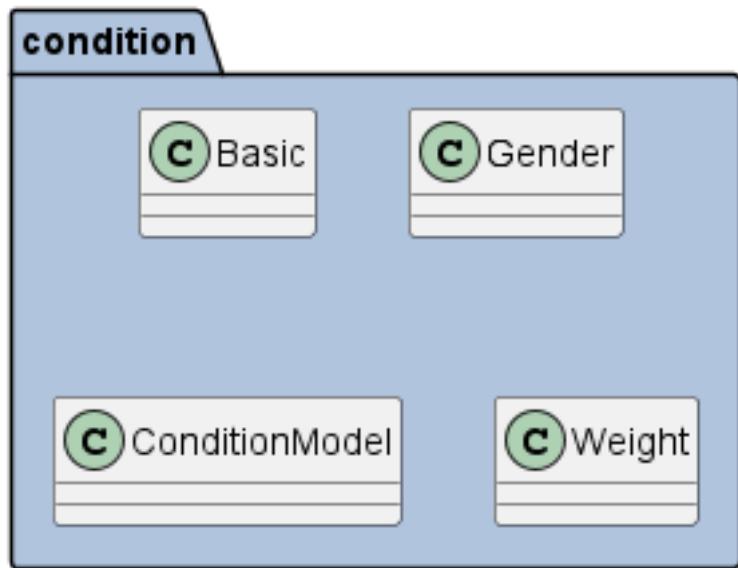


Figura 43: Módulo *condition*.

contact

A pesar de que este módulo solo contiene una clase que se encarga de mostrar toda la información de contacto con administradores, se definió aparte para añadir más clases en futuras iteraciones.

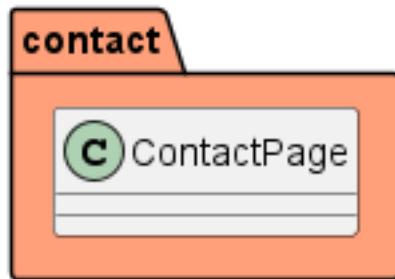


Figura 44: Módulo *contact*.

db

Este módulo es el encargado de comunicarse directamente con la base de datos de Firebase. Este contiene diferentes clases para las diferentes entidades que existen y para el almacenamiento.

- **DBBase:** Clase abstracta que contiene los métodos básicos que deben incluir el resto de clases.
- **DBAddresses:** Clase que maneja las direcciones y coordenadas.
- **DBChats:** Clase que maneja toda la información relacionada con chats.
- **DBDogs:** Clase que maneja toda la información relacionada con perros y su información.
- **DBFavorites:** Clase que se encarga exclusivamente de actualizar la información de los favoritos dentro de la tabla de perros.
- **DBMessages:** Clase que maneja toda la información relacionada con mensajes.
- **DBNotifications:** Clase que maneja toda la información relacionada con notificaciones
- **DBStorage:** Clase que se encarga de insertar, actualizar y eliminar toda las imágenes relacionadas con alguna de las entidades.
- **DBUsers:** Clase que maneja toda la información relacionada con los usuarios.

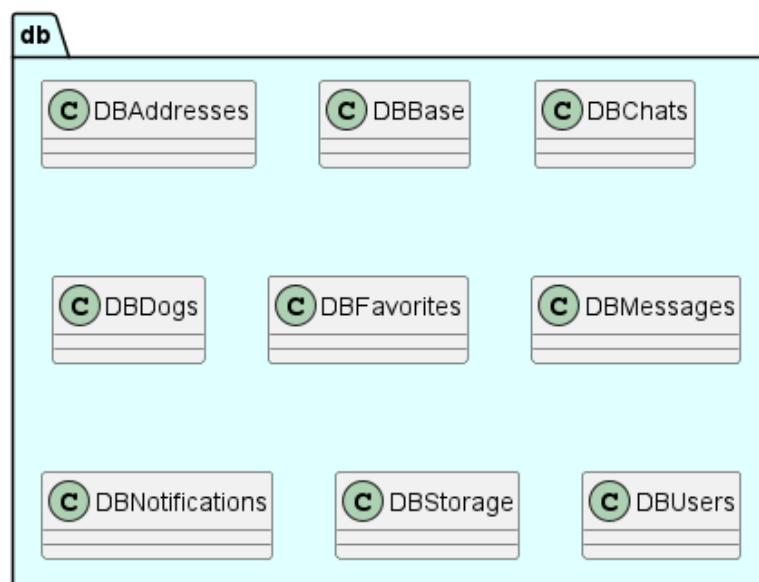


Figura 45: Módulo *db*.

dog

Módulo que contiene las clases principales relacionadas con los perros.

- **AvailableDogsPage:** Pantalla de lista de perros que contiene los filtros y una barra de búsqueda. Se encarga de manejar las consultas y mostrar los resultados.
- **DogFeature:** Widget que se instancia con la información de alguno de los atributos del perro para mostrarlo en el perfil.
- **DogProfile:** Pantalla que contiene toda la información de un perro, incluyendo el botón de favoritos y compartir.
- **DogModel:** Clase que se encarga de generar y manejar modelos de perros a partir de la información en la base de datos.

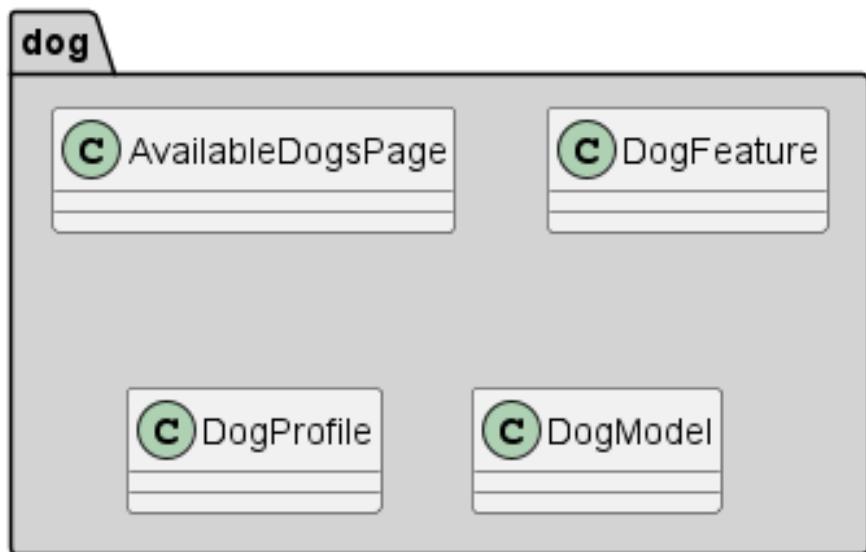


Figura 46: Módulo *dog*.

home

Se incluyen todas las pantallas de inicio definidas para los diferentes roles.

- **AdminHome:** Pantalla de inicio para administradores.
- **MyHome:** Pantalla de inicio para usuarios.
- **MyHomePageCompany:** Pantalla de inicio para protectoras.

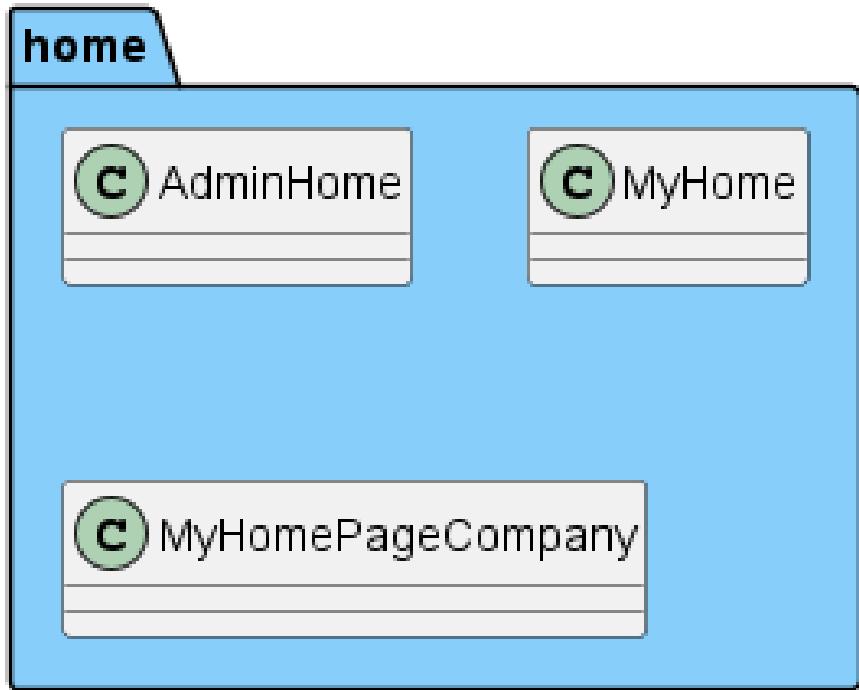


Figura 47: Módulo *home*.

init

La clase contenida en este módulo se encarga de manejar la pantalla que se debe mostrar cuando se abre la aplicación, además de inicializar todos los servicios que lo requieran. La pantalla mostrará la *Splashscreen* y dependiendo si hay un usuario iniciado se redirigirá a la pantalla de inicio del rol correspondiente o a la pantalla de inicio de sesión.

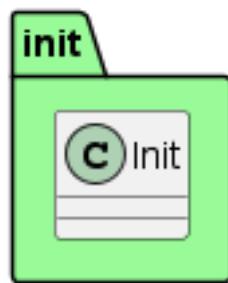


Figura 48: Módulo *init*.

legal

Este módulo solo contiene una pantalla que contiene toda la información legal relacionada con las adopciones y los usos de la aplicación.



Figura 49: Módulo *legal*.

login

Contiene la página para iniciar sesión y la de recuperación de contraseña.

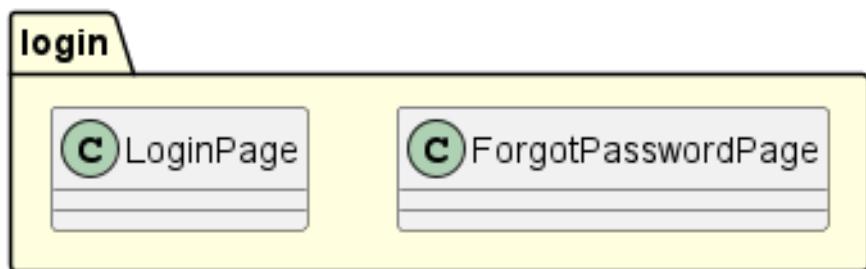


Figura 50: Módulo *login*.

message

En este módulo se definen los widgets relacionados con los mensajes que se utilizan dentro de las salas de chat.

- **DateMessageDivide:** Este widget se inserta entre mensajes consecutivos que son de días diferentes para indicar la fecha de los mensajes posteriores.
- **MessageWidget:** Widget que muestra el contenido del mensaje que puede ser texto o imagen, además contiene un ícono que indica si un mensaje ha sido leído o no.
- **MessageModel:** Clase que se encarga de generar y manejar modelos de mensajes a partir de la información en la base de datos.

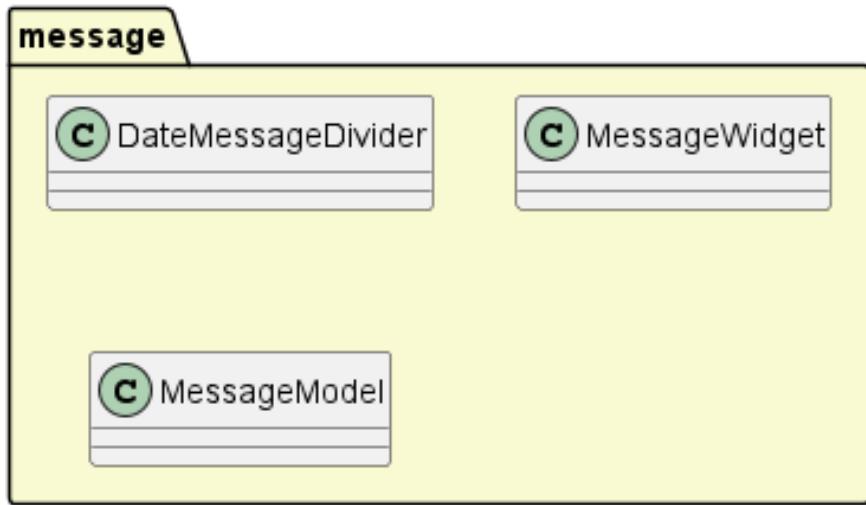


Figura 51: Módulo *message*.

orderby

Este módulo contiene el modelo para añadir la cláusula de orden a las consultas.



Figura 52: Módulo *orderby*.

register

Contiene todas las páginas relacionadas con los registros dentro de la app. Cada una de las pantallas incluye un formulario específico que se encarga de todas las validaciones.

- **RegisterAsCompanyPage**: Pantalla de registro de protectora.
- **RegisterAsUserPage**: Pantalla de registro de usuario.
- **RegisterDog**: Pantalla de registro de perro.

- **RegisterPage**: Pantalla que ofrece las opciones para registrarse como protectora o usuario.

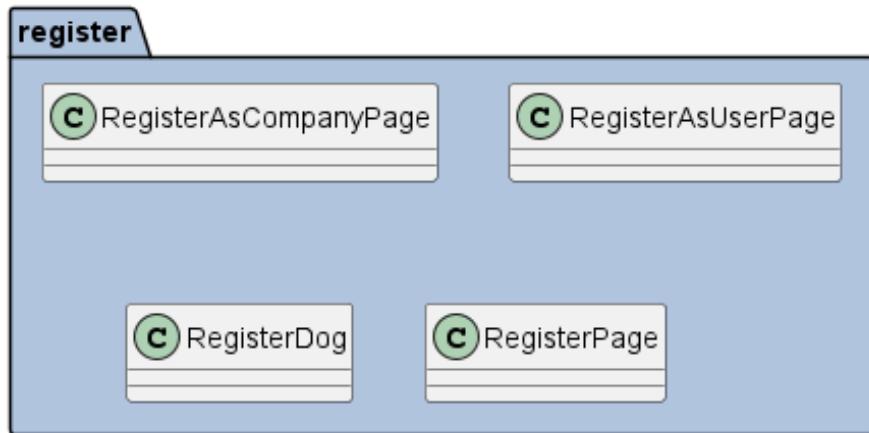


Figura 53: Módulo *register*.

section

Contiene diferentes widgets que definen secciones que se reutilizan en algunas de las pantallas.

- **AdminCategorySection**: Incluye la sección principal de la página de inicio de administrador.
- **CompanyCategorySection**: Incluye la sección principal de la página de inicio de protectora.
- **RecentAdoptionsSection**: Define la sección de adopciones recientes que se usa en diferentes páginas de inicio.
- **UserCategorySection**: Incluye la sección principal de la página de inicio de usuario.
- **UserFavoritesSection**: Define la sección de favoritos de un usuario específico.
- **UserOwnedDogsSection**: Define la sección de perros que son de un usuario específico.

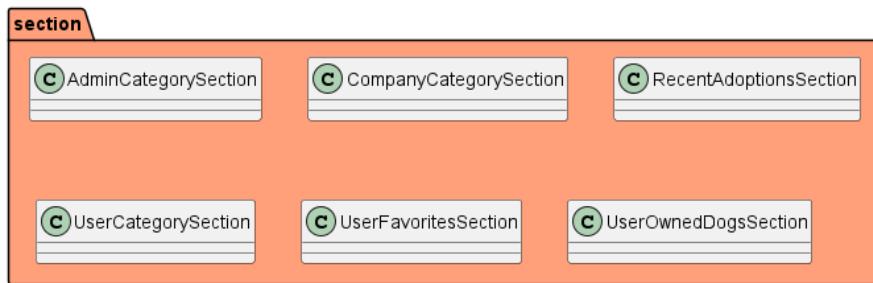


Figura 54: Módulo *section*.

services

Se incluyen clases que manejan los servicios que proporciona Flutter o Firebase además de otras clases que son servicios creados específicamente para la aplicación.

- **AddressService**: Servicio para manejar los modelos de las direcciones y las coordenadas.
- **AwesomeNotificationService**: Clase que maneja el servicio de notificaciones.
- **GeocodingService**: Clase que maneja el servicio de geocoding que proporciona Google dentro de la app.
- **RoutingService**: Servicio para manejar las rutas de la aplicación.
- **Auth - AuthUsers**: Clase que maneja el servicio de autenticación que proporciona Firebase.

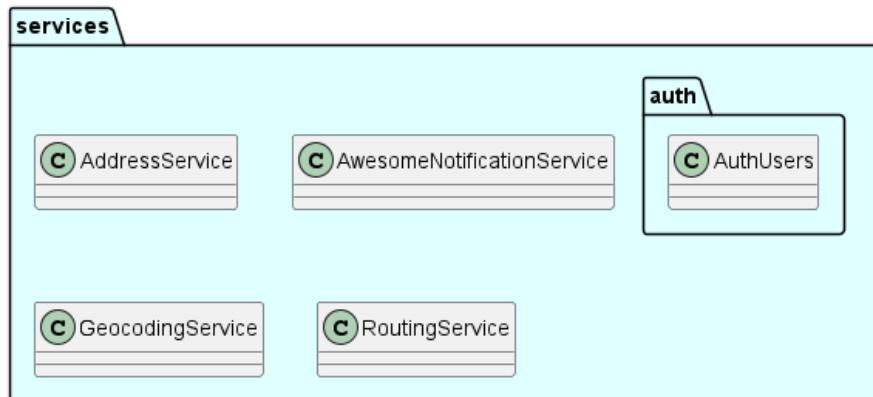


Figura 55: Módulo *services*.

update

Contiene las páginas que incluyen formularios para la actualización de datos de usuarios y de perros.



Figura 56: Módulo *update*.

user

Módulo que contiene todas las clases relacionadas con los usuarios y protegidas.

- **CurrentUser:** Clase singleton que se encarga de manejar todos los datos relacionados con el usuario iniciado a lo largo de la aplicación.
- **Logged UserModel:** Clase para manejar el modelo del usuario iniciado.
- **User Model:** Clase que se encarga de generar y manejar modelos de usuarios a partir de la información en la base de datos.
- **Profile Page:** Pantalla que contiene toda la información personal del usuario. Puede contener un listado de perros y un botón para abrir una página de mapa dependiendo del rol de usuario.
- **User Map Widget:** Elemento que se instancia en las listas usadas en los mapas con toda la información del usuario.
- **Users Page:** Pantalla de la lista de usuarios que contiene además la barra de búsqueda. Se encarga de manejar las consultas y mostrar los resultados.

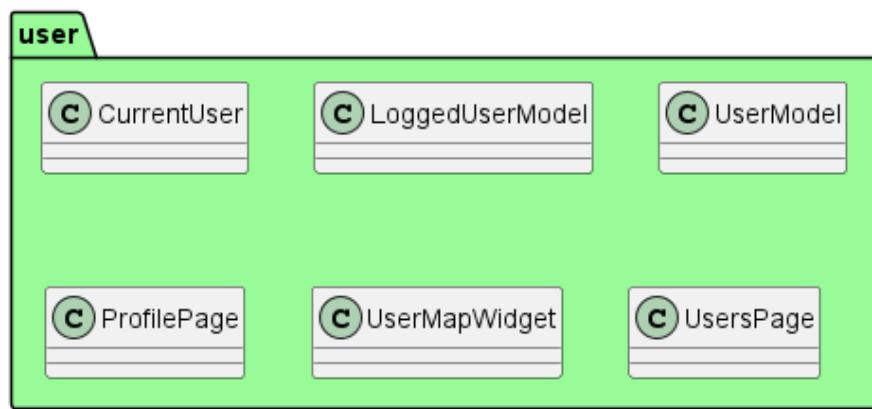


Figura 57: Módulo *user*.

5.2. Pantallas

En esta sección se exponen todas las pantallas resultantes del desarrollo.

Pantalla de carga

La pantalla de carga es una pantalla de transición que se utiliza justo después de iniciar sesión, mientras se carga toda la información del usuario.

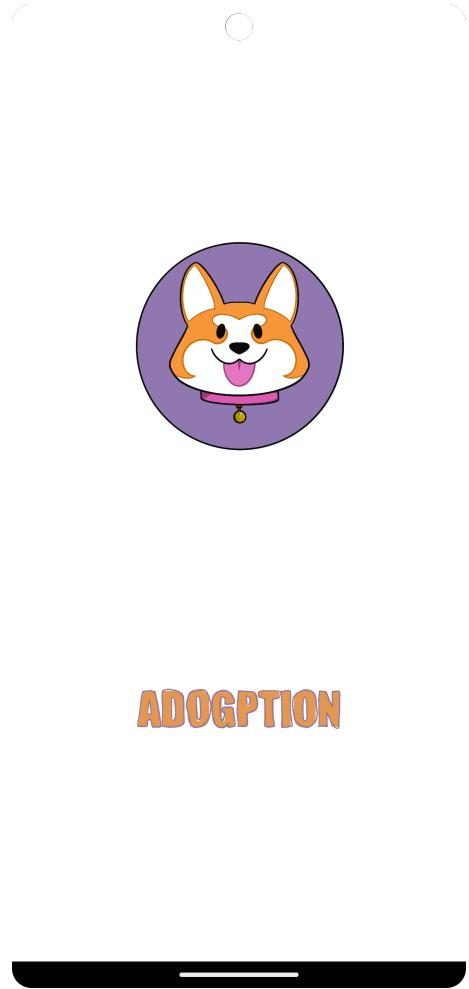


Figura 58: Pantalla de carga

Inicio de sesión

La pantalla de inicio de sesión muestra un formulario básico para introducir el correo electrónico y la contraseña. Este formulario indicará si hay errores durante el intento de iniciar sesión. El botón de inicio de sesión maneja toda la autenticación mediante el servicio de autenticación de Firebase.

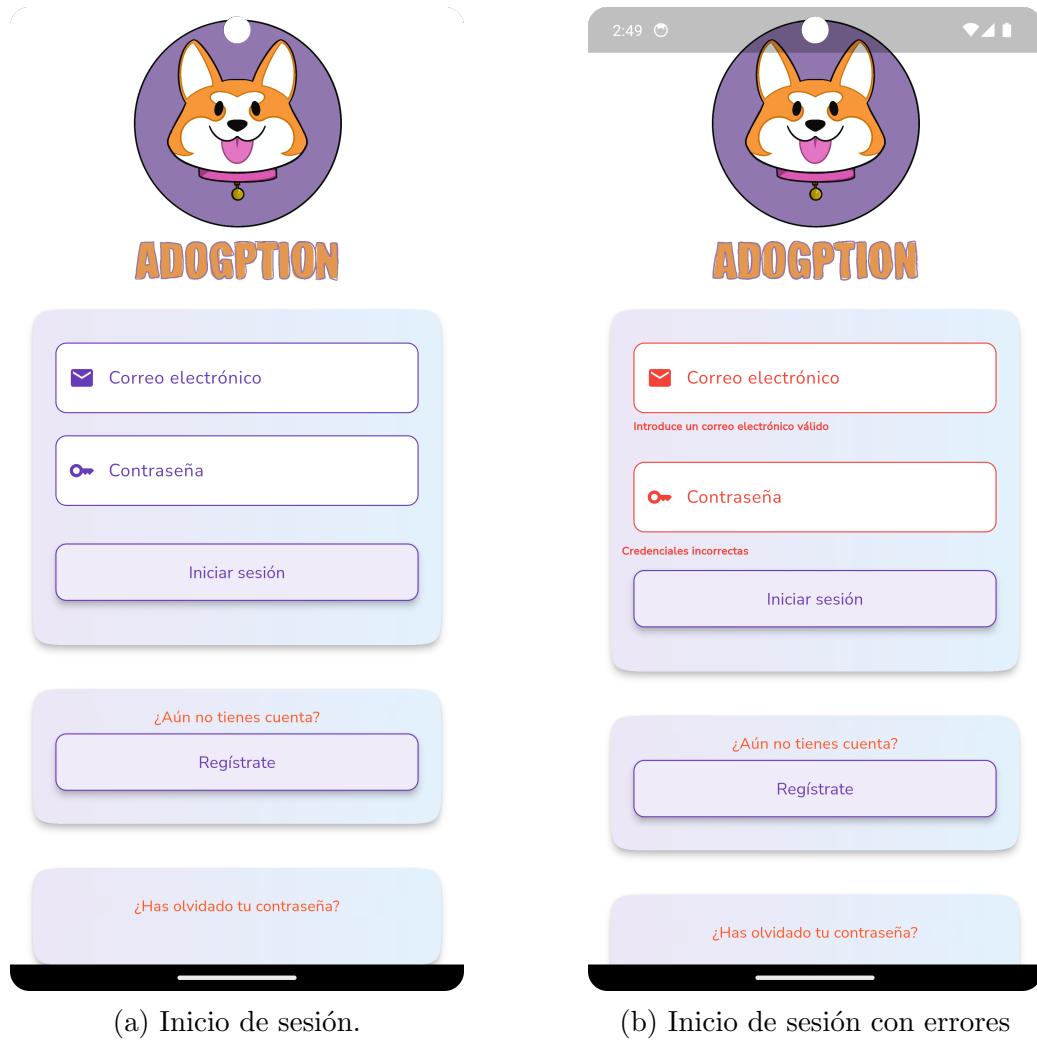


Figura 59: Pantalla de inicio de sesión.

Recuperación de contraseña

Esta pantalla contiene un formulario que solo incluye un campo para introducir el correo electrónico. Si el correo electrónico está registrado, el servicio de autenticación de Firebase se encargará de enviar un correo electrónico con las instrucciones para cambiar la contraseña.



Figura 60: Pantalla de recuperación de contraseña.

Registro

Si se pulsa el botón *Regístrate*, que se incluye en la pantalla de inicio de sesión mostrada en la Figura 59, se redirige a una pantalla que muestran dos botones que llevan a los formularios para registrarse como usuario normal o como protectora respectivamente. Ambos formularios de registro piden los mismos datos a excepción de la dirección, que solo se requiere en el caso de las protectoras.



Figura 61: Página de opciones de registro.

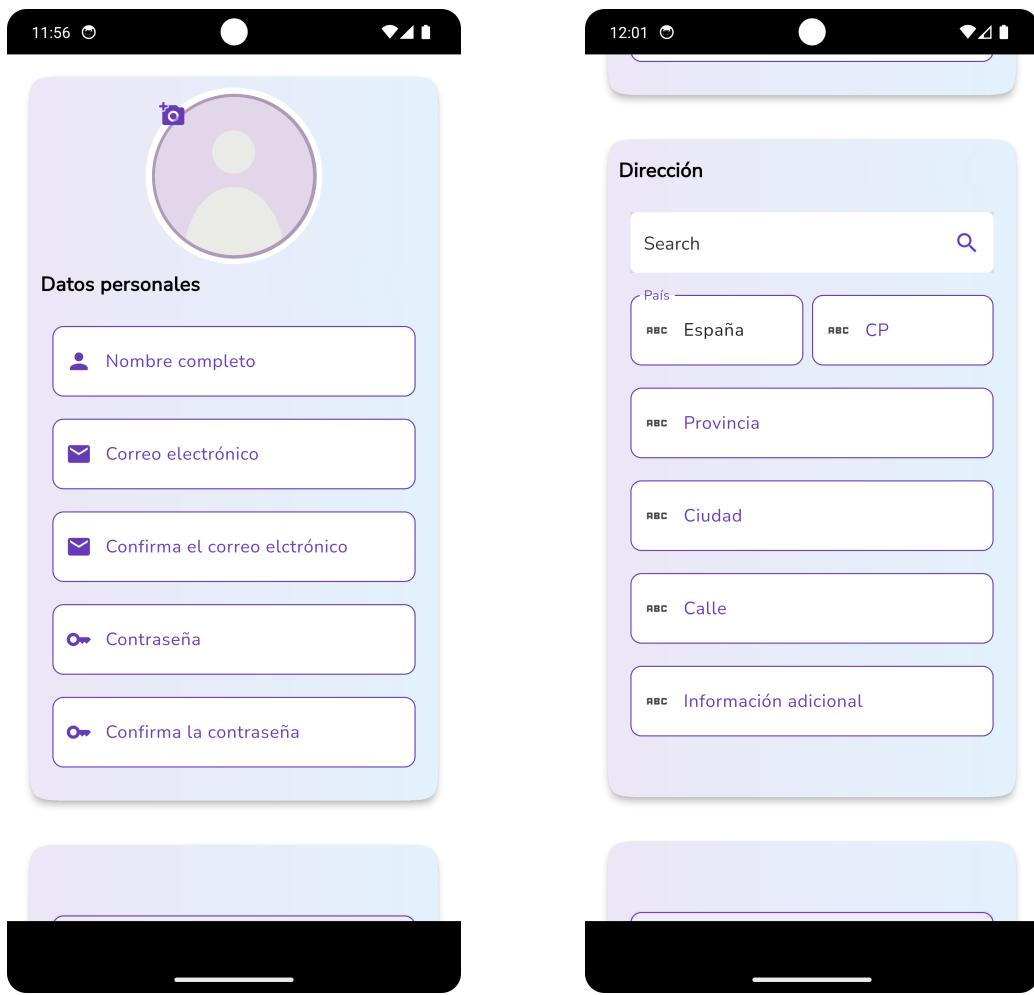


Figura 62: Pantallas de registro.

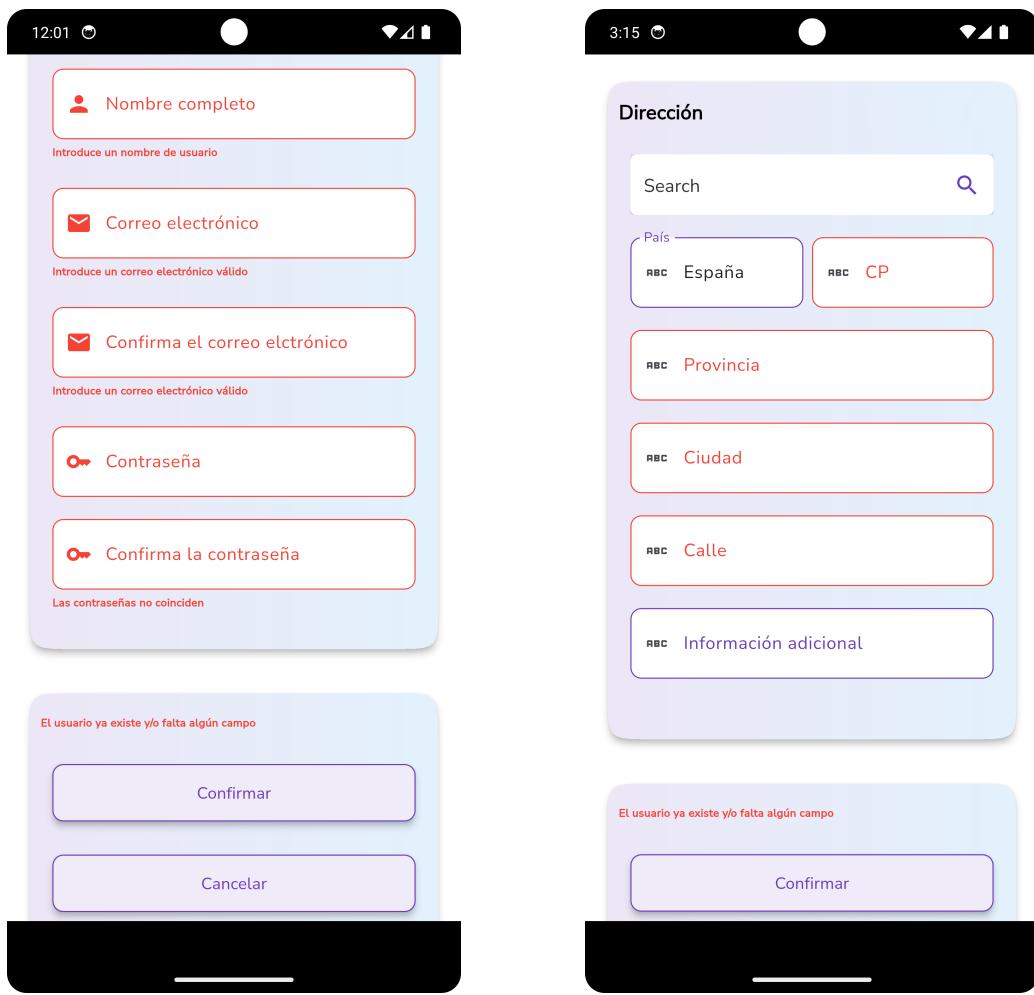
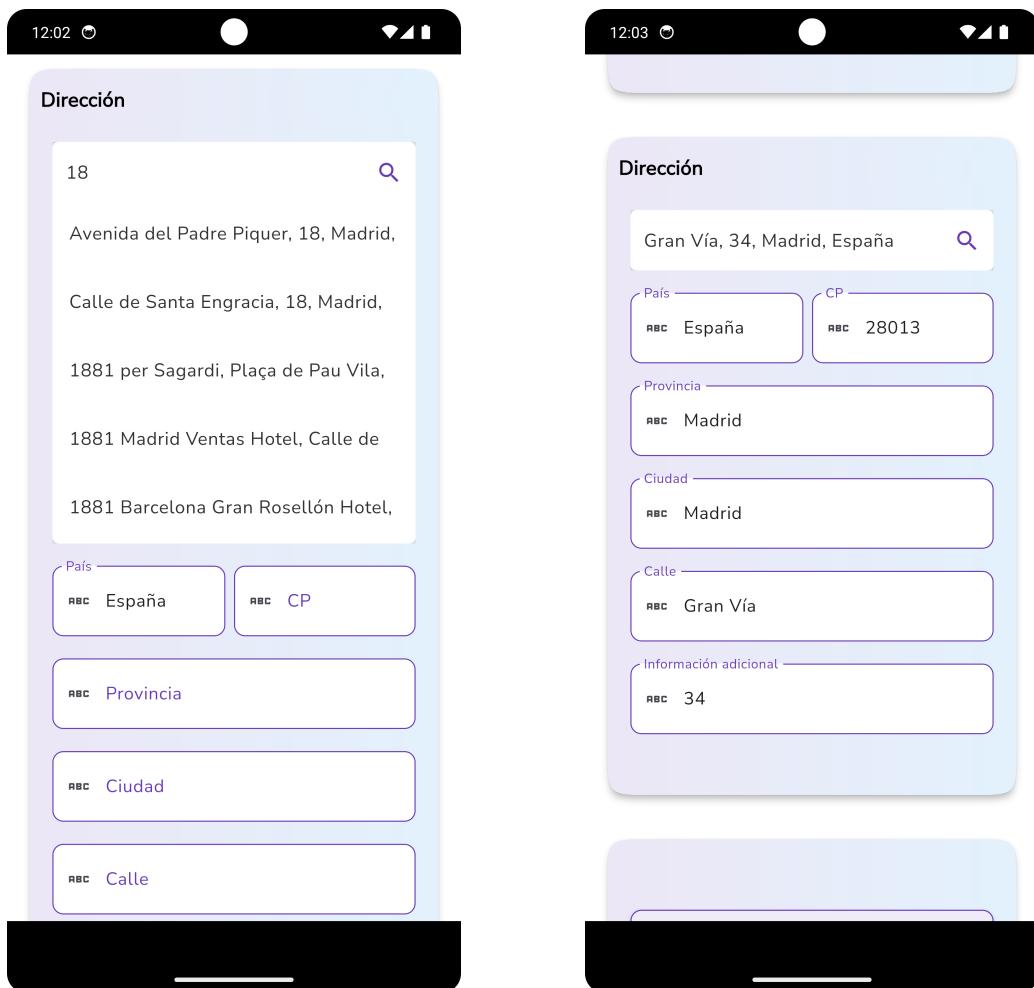


Figura 63: Pantallas de registro con errores.

El componente más complejo de estos formularios es la barra de búsqueda de direcciones. Este componente se ha desarrollado en base a un componente proporcionado por el paquete `search_map_location` [9], que proporciona la funcionalidad básica para buscar direcciones con la API de Google Maps. A este componente se le ha añadido el comportamiento de autocompletar los campos del formulario una vez se selecciona una dirección, además de ajustar todos los estilos según los de la aplicación.



(a) Búsqueda de direcciones.

(b) Autocompletar campos de direcciones.

Figura 64: Barra de búsqueda de direcciones.

Pantallas de inicio

Para cada uno de los roles definidos, se ha creado una pantalla de inicio específica. Ambas pantallas incluyen bastante contenido en común, como la sección de adopciones recientes o la sección de favoritos. Se incluyen listas para mostrar los perros marcados para adoptar o acoger en ambas pantallas, con la diferencia de que una protectora solo verá sus propios perros en esas listas.

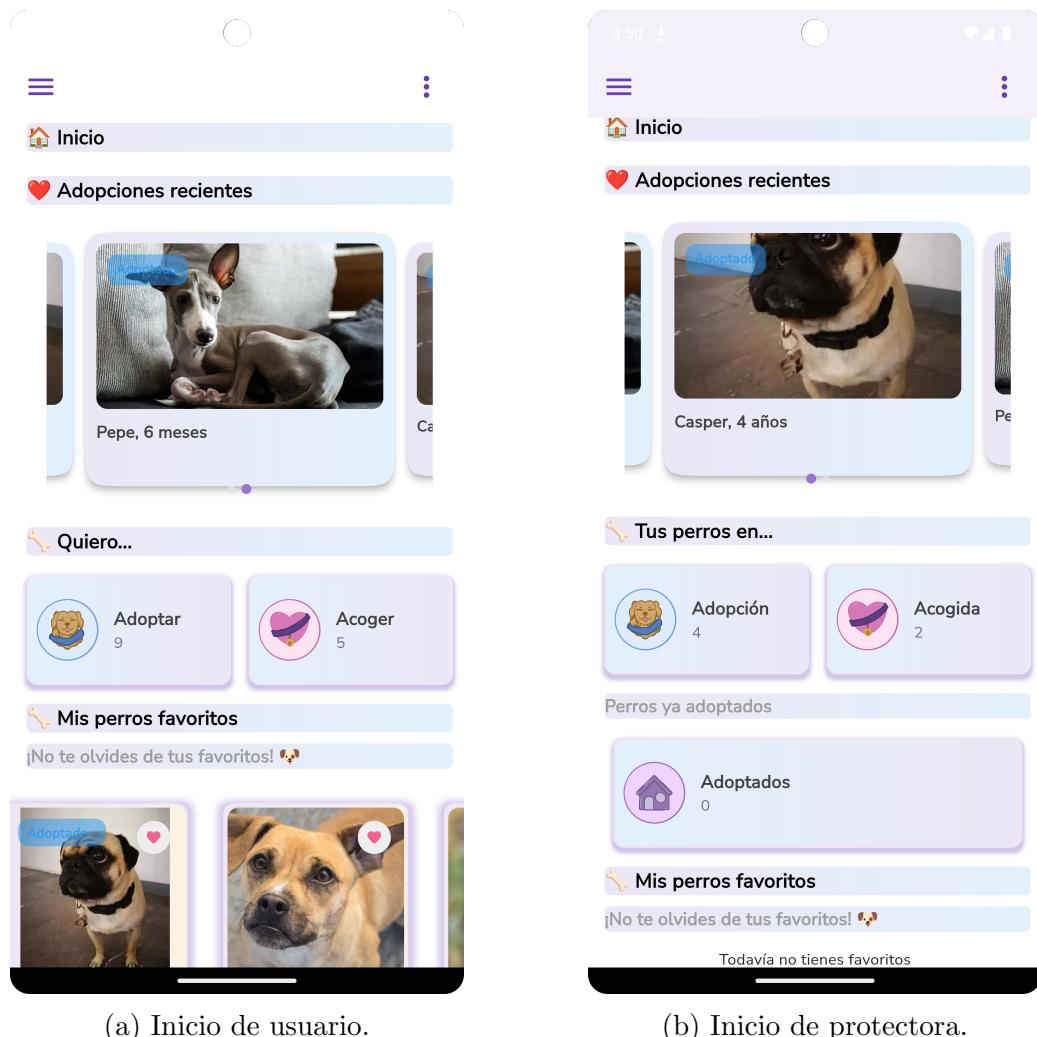
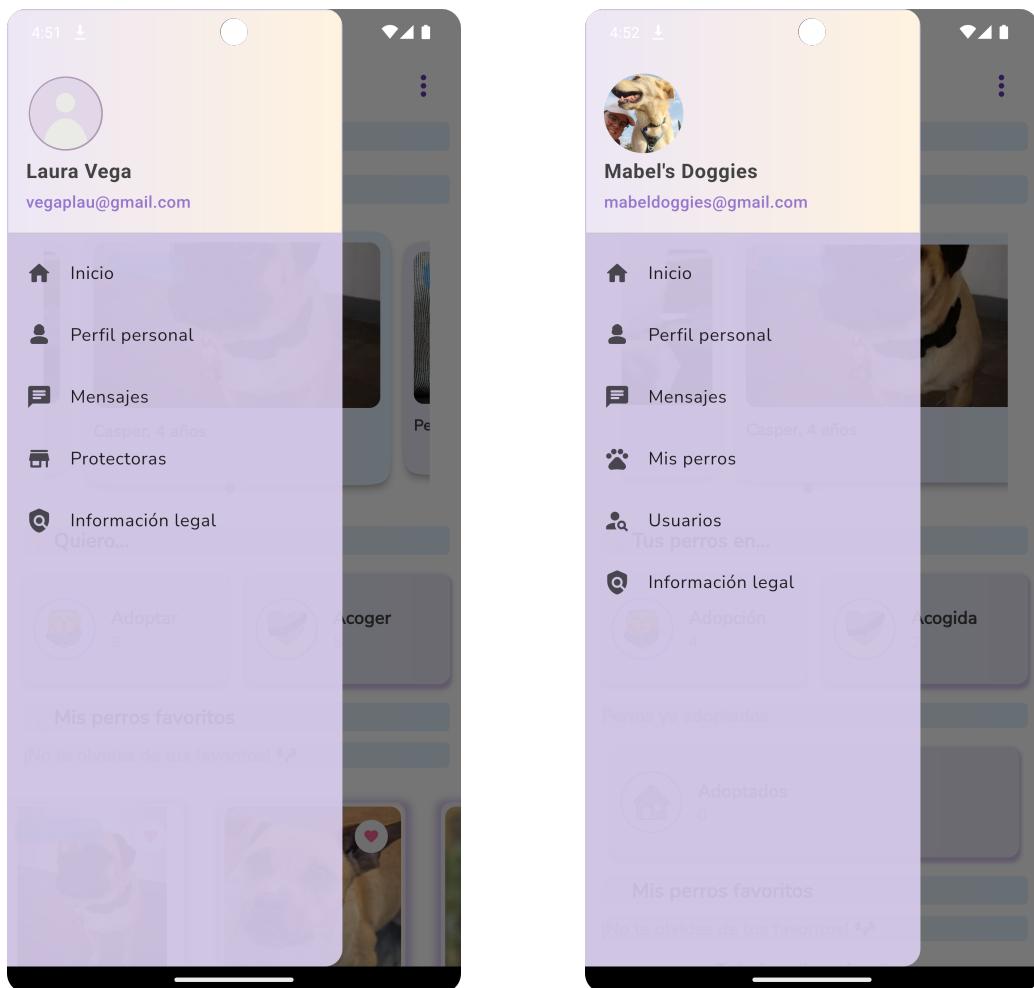


Figura 65: Pantallas de inicio por rol.

Para cada uno de los roles también se han implementado menús laterales con distintas opciones.



(a) Menú lateral de usuario.

(b) Menú lateral de protectora.

Figura 66: Menús laterales por rol.

Si una protectora inicia sesión sin estar verificada por un administrador, se muestra una pantalla informándole que tiene que esperar la verificación para poder seguir usando la aplicación. Aunque la protectora no pueda subir perros o mensajear, si podrá abrir un chat con un administrador desde la página de contacto.



Figura 67: Pantalla de protectora no verificada.

Listas

A lo largo de la aplicación se ha condensando la información en forma de listas.

Ahora mismo no es disponible crear listas personalizadas y guardarlas, por ello, se han proporcionado una serie de listas genéricas que se construyen con unos filtros fijos a la hora de instanciarse. Por ejemplo, para construir la lista de los perros puestos en adopción, se hace de la siguiente manera:

```
AvailableDogsPage(  
    'Perros en adopción',  
    [  
        Condition('forAdoption', 'isEqualTo', true),  
        Condition('adopted', 'isEqualTo', false)  
    ],  
    true  
)
```

Estas clases disponen de un método que se encarga de concatenar todos los filtros fijos en la consulta base de la lista haciendo uso del método proporcionados por la clase *Condition*.

```
getCollectionWithFilters() {  
    Query<Map<String, dynamic>> collection = FirebaseFirestore.instance.collection('dogs')  
        .where('available', isEqualTo: true);  
    for (Condition condition in conditions) {  
        collection = Condition.addClauseByCondition(collection, condition);  
    }  
    return collection;  
}
```

Cuando se aplican filtros dinámicos, se utiliza el siguiente método que se encarga de concatenar todos los filtros rápidos aplicados a la consulta que ya contiene los filtros fijos.

```
getQuickFiltersAndSnapshot() {  
    Query<Map<String, dynamic>> collection = widget.getCollectionWithFilters();  
    for (Condition condition in quickFilters) {  
        collection = Condition.addClauseByCondition(collection, condition);  
    }  
    return collection;  
}
```

Todas las listas de la aplicación ordenan los resultados según la distan-

cia.

Usuarios

Las listas de usuarios pueden incluir tanto como usuarios normales como protectoras, algunas de las listas construidas en la aplicación contienen filtros específicos para mostrar únicamente listas de un rol determinado. Este tipo de listas disponen de una barra de búsqueda que filtra resultados según el texto que se introduzca, dicho texto busca coincidir con diferentes propiedades del usuario tales como la dirección o el nombre.

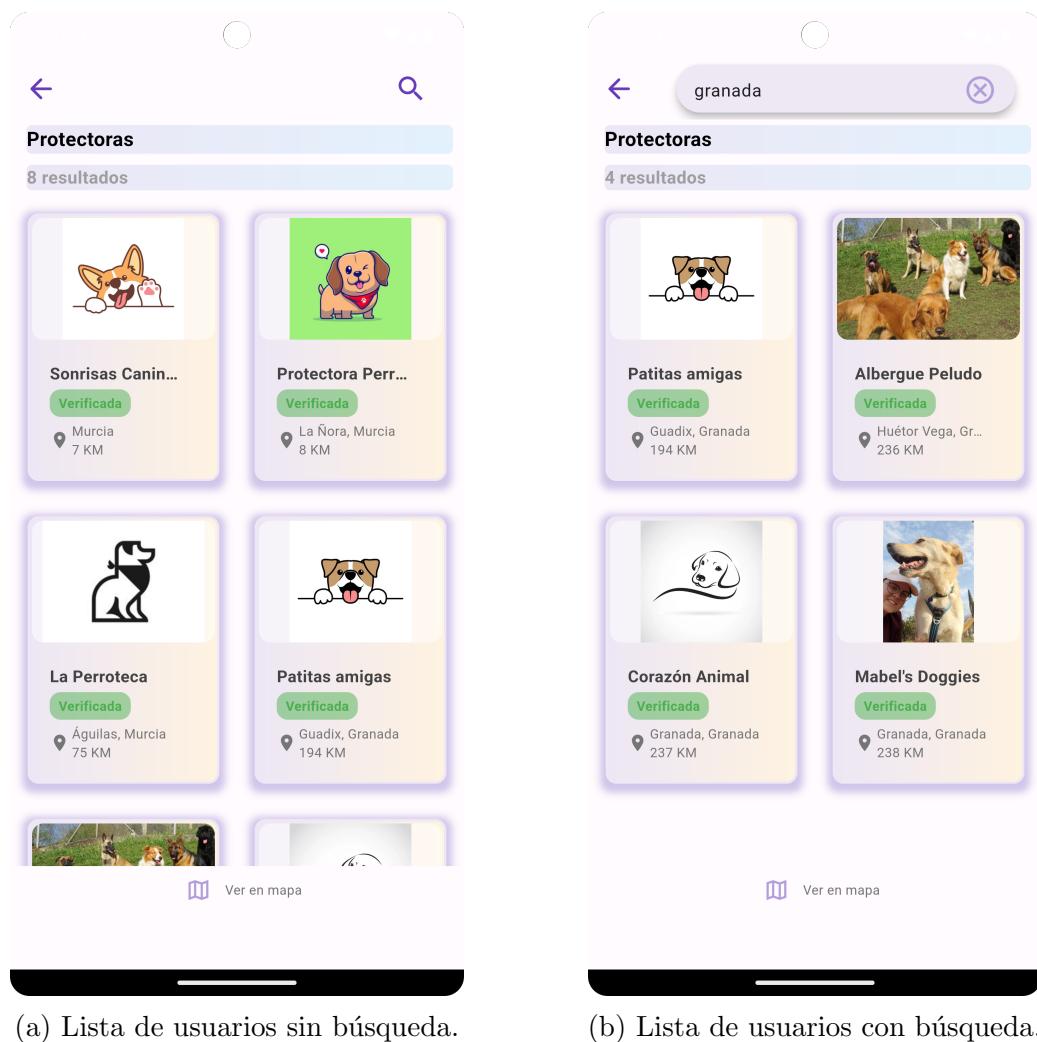


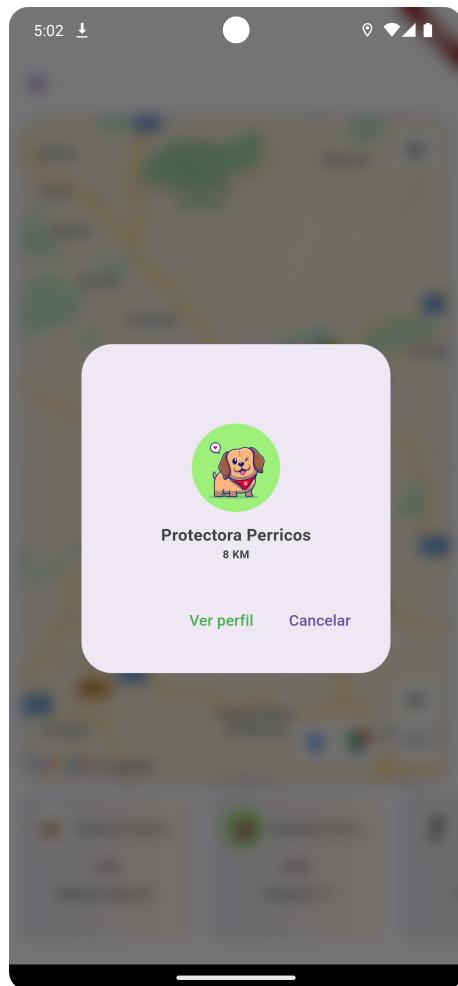
Figura 68: Lista de usuarios.

Estas listas, además, incluyen un botón que permite ver los resultados de la lista colocados en un mapa en forma de marcador, la misma lista se colocará

debajo del mapa de la página. Los marcadores pueden ser pulsados para abrir un pop-up que redirige al perfil de la protectora. Si la lista contiene usuarios sin dirección, no se añadirán al mapa.



(a) Mapa y lista.

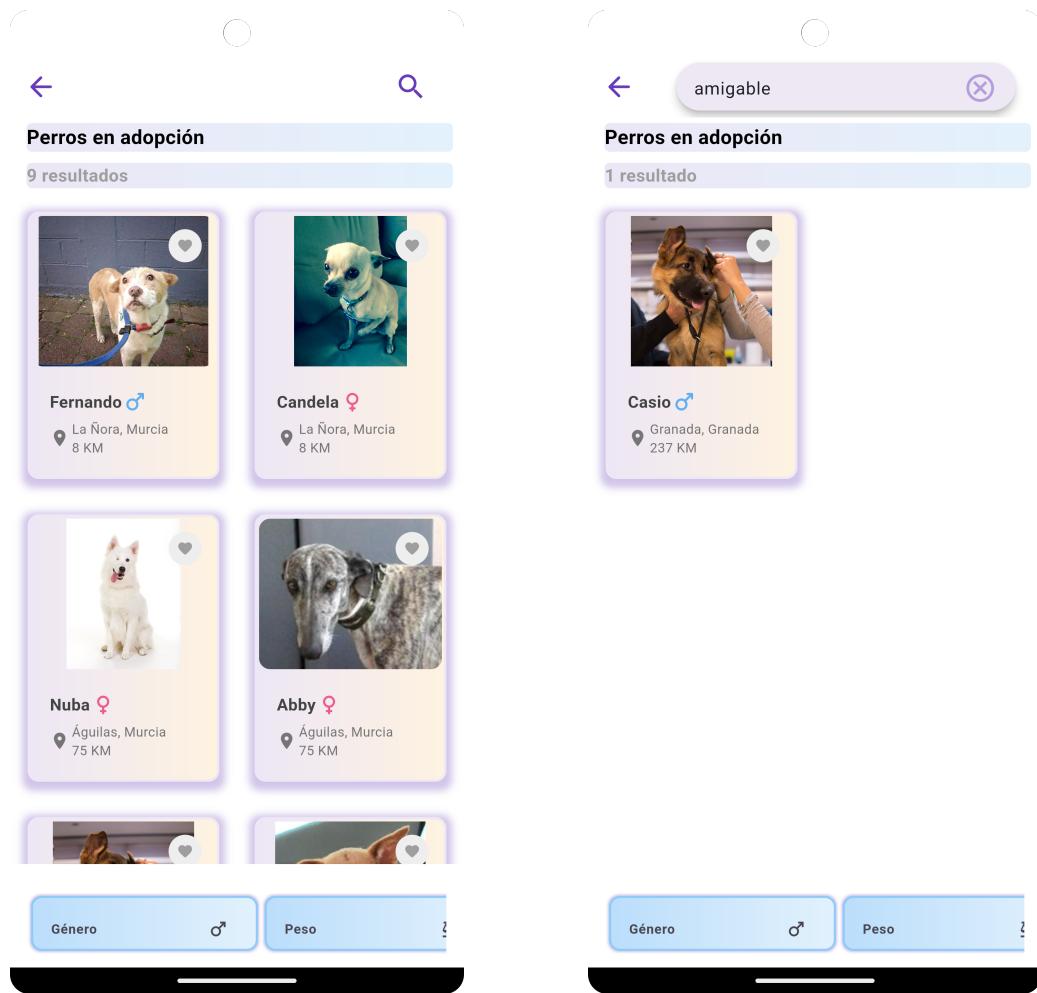


(b) Acción del marcador.

Figura 69: Página de mapa.

Perros

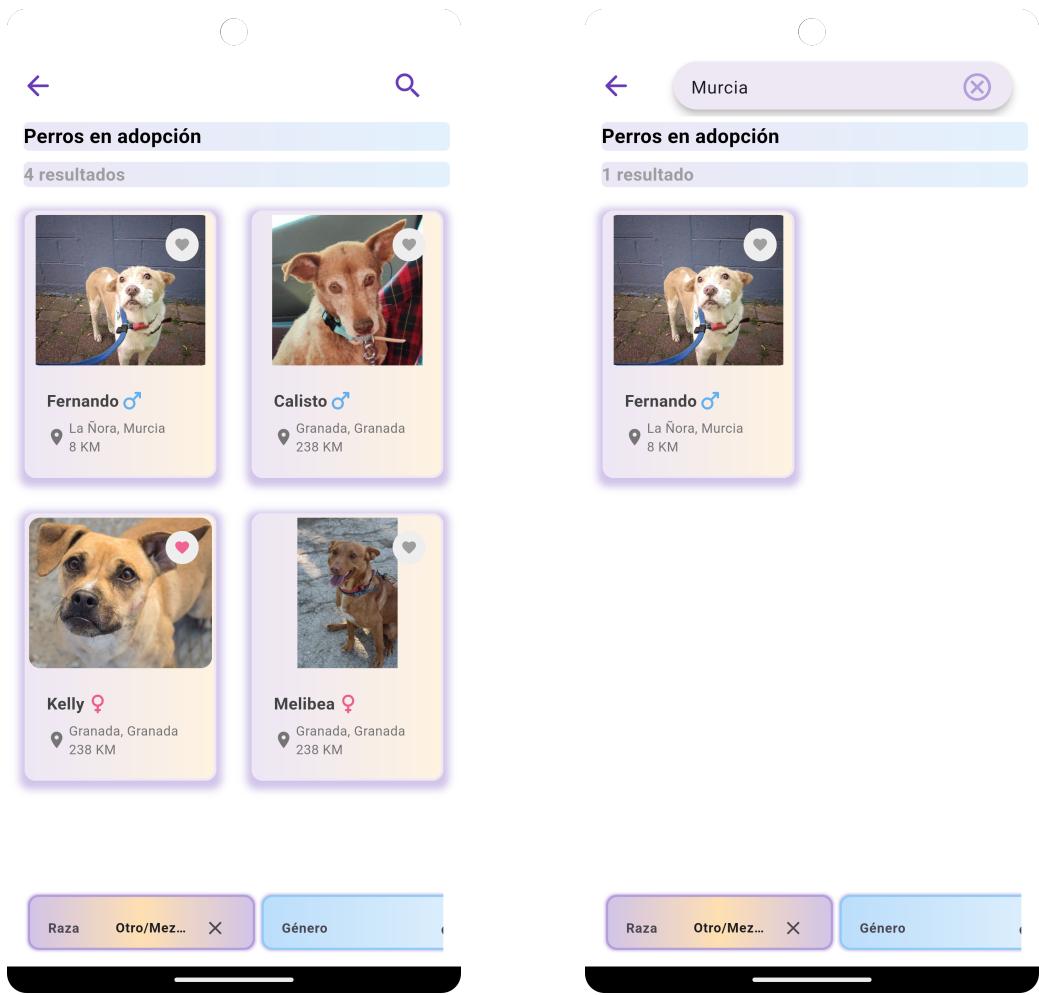
Las listas de perros incluyen cualquier tipo de perro que esté dado de alta en la aplicación. Los resultados dependerán de los filtros introducidos por el usuario o los definidos a la hora de construir la lista. Estas listas también incluyen una barra de búsqueda para filtrar resultados además de una barra con varios filtros rápidos que corresponden con algunas de las propiedades de los perros.



(a) Lista de perros sin búsqueda o filtros.

(b) Lista de perros con búsqueda

Figura 70: Lista de perros.



(a) Lista de perros con filtros.

(b) Lista de perros con búsqueda y filtros

Figura 71: Lista de perros con filtros.

Los widgets de los filtros abren un pop-up con las diferentes opciones disponibles cuando se pulsa en ellos. Se incluye una barra de búsqueda para filtrar las opciones disponibles.

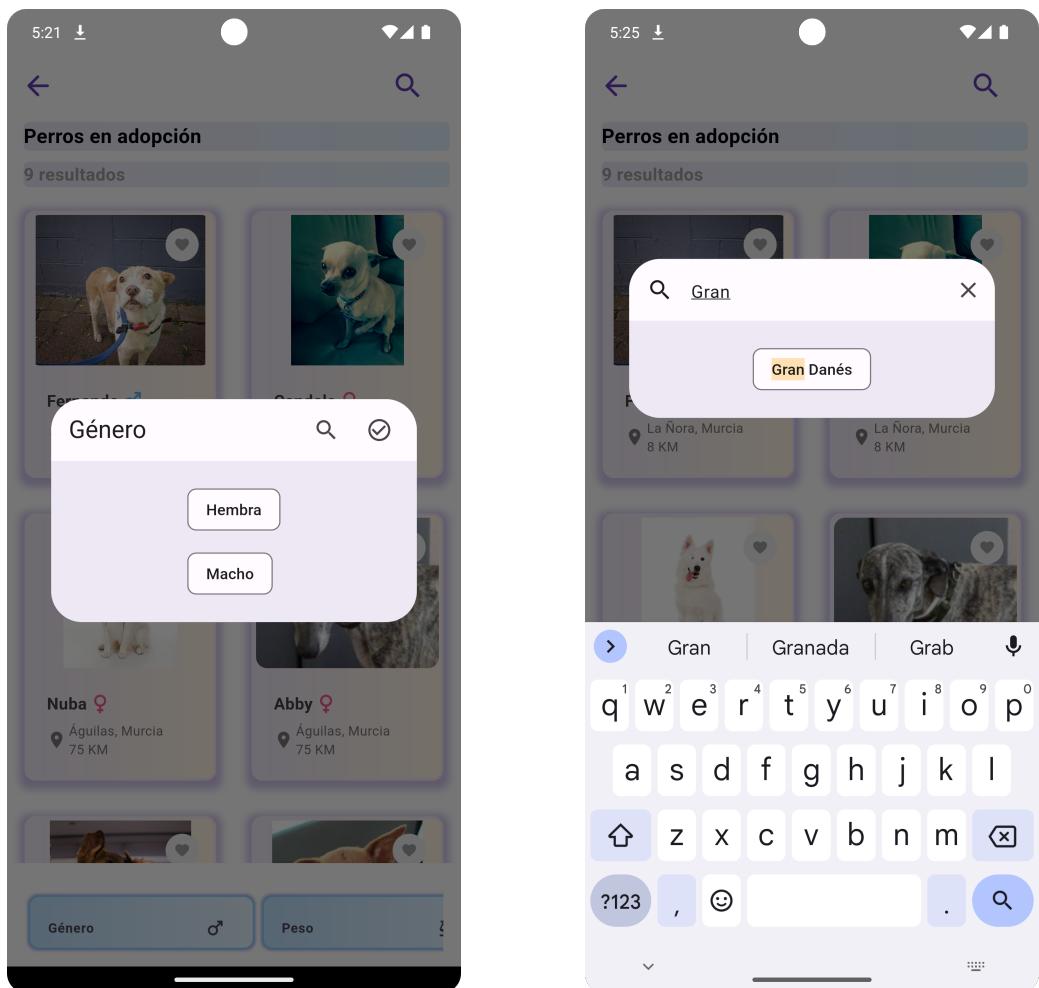


Figura 72: Filtros.

Perfil personal

A continuación se muestra la pantalla de perfil personal que se genera para cada uno de los usuarios registrados. Dependiendo de algunas condiciones se mostrarán unos botones u otros:

- Si el usuario es una protectora , se incluye un botón para abrir un mapa con la protectora ubicada en él.
- El perfil que se visita no es el propio, se incluye un botón de contacto para abrir un chat con el usuario.

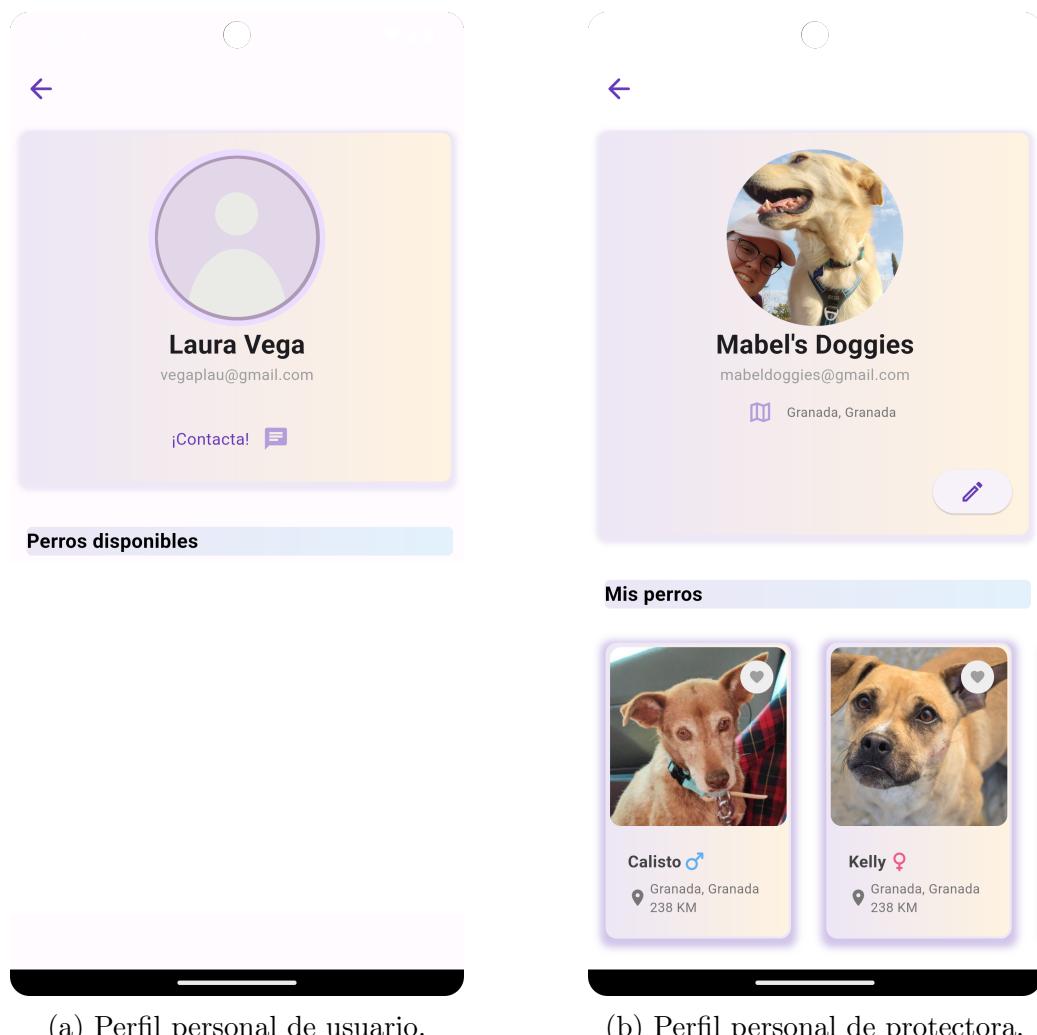
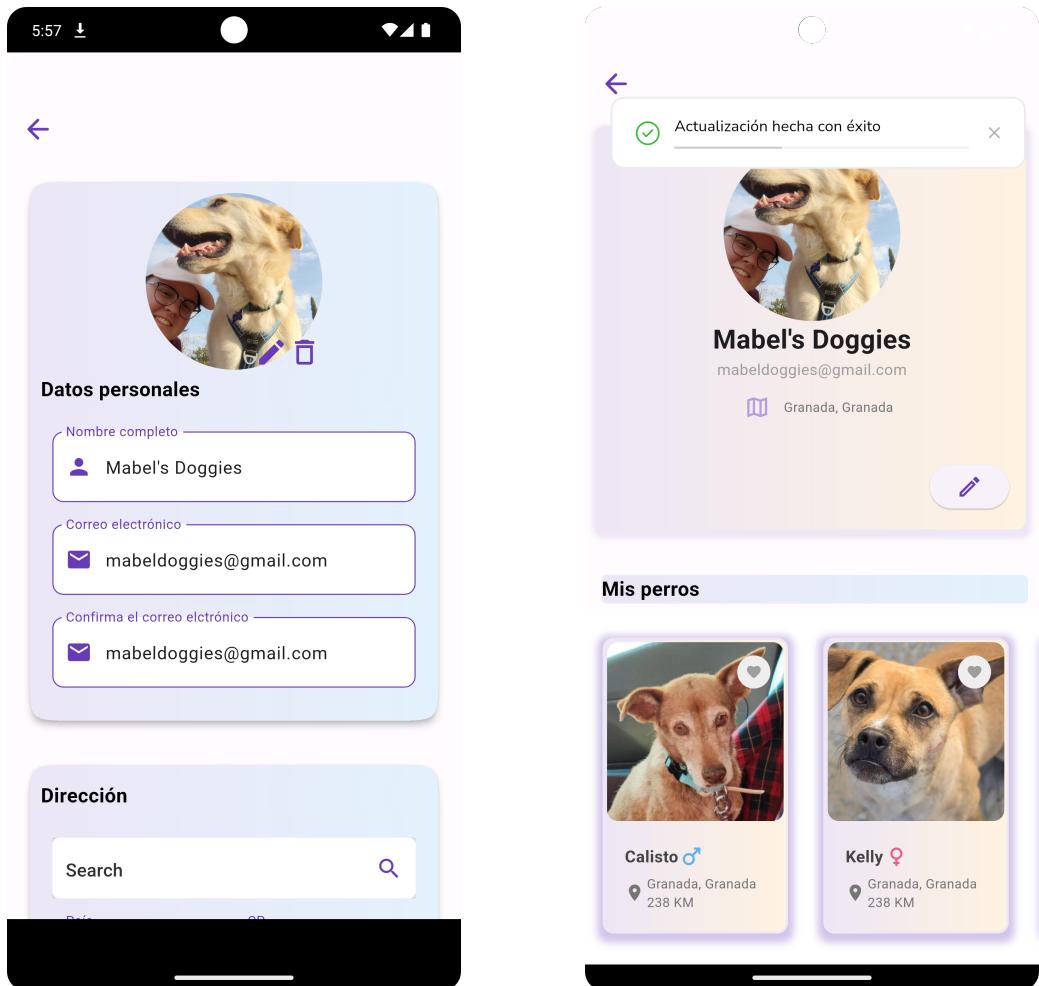


Figura 73: Pantalla de perfil personal.

Actualizar datos de usuario

Los formularios para actualizar datos de usuario son casi idénticos a los proporcionados para registrarse. En primer lugar, los formularios cargan todos los datos actuales del usuario y permite que se introduzcan todos los cambios que se requieran. Posterior a confirmar los cambios, si la actualización es correcta, el usuario es redirigido a su perfil y una notificación es mostrada indicando que la actualización se ha hecho correctamente.



(a) Formulario de actualización.

(b) Notificación de correcta actualización.

Figura 74: Actualización de datos personales.

Registro perros

Las protectoras pueden añadir perros a la aplicación desde su lista personal llamada *Mis perros*. En esta pantalla se incluye una barra superior con un botón específico que redirige al formulario de registro para perros. El formulario proporcionado incluye todos los atributos que requiere un perro para ser dado de alta.

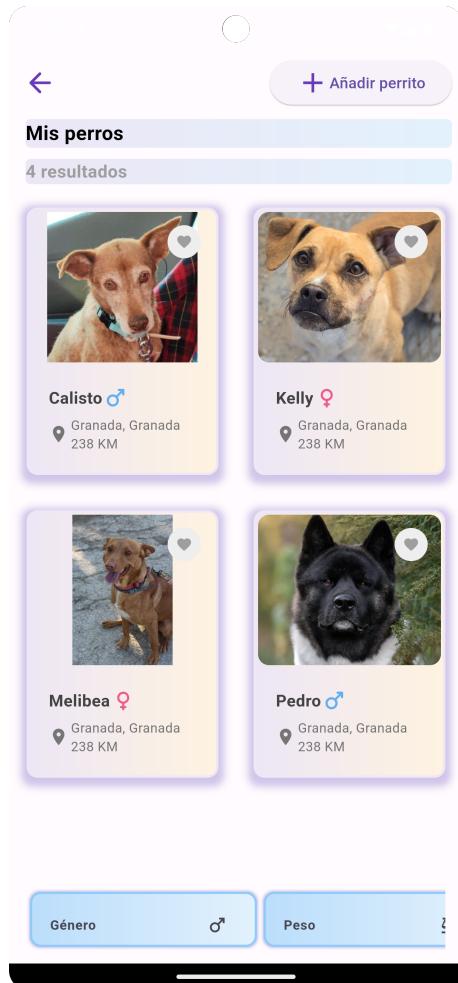


Figura 75: Pantalla *Mis perros*.

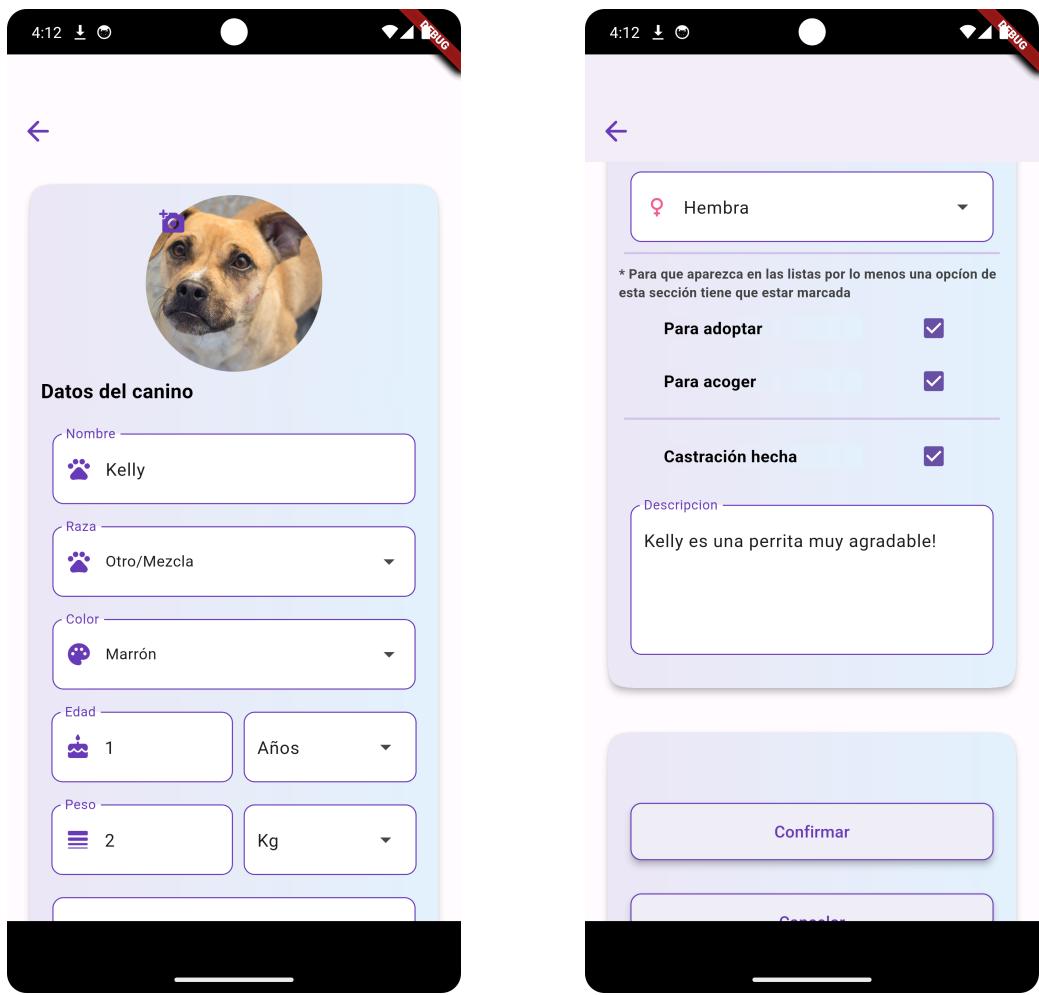


Figura 76: Pantalla de registro de perro.

Perfil perro

Para mostrar toda la información de un perro a un usuario, se ha condensado toda la información en una pantalla. En esta pantalla se incluye toda la información relacionada con el perro, tales como nombre, raza, peso, etc. Se incluye también un mapa que muestra la ubicación de la protectora a la que pertenece, además de un botón de contacto para abrir chat con la protectora. Si el usuario que visita el perfil es el dueño del perro, puede visualizar una sección inferior con diferentes botones para abrir la edición del perfil, eliminar el perro o marcarlo como adoptado.

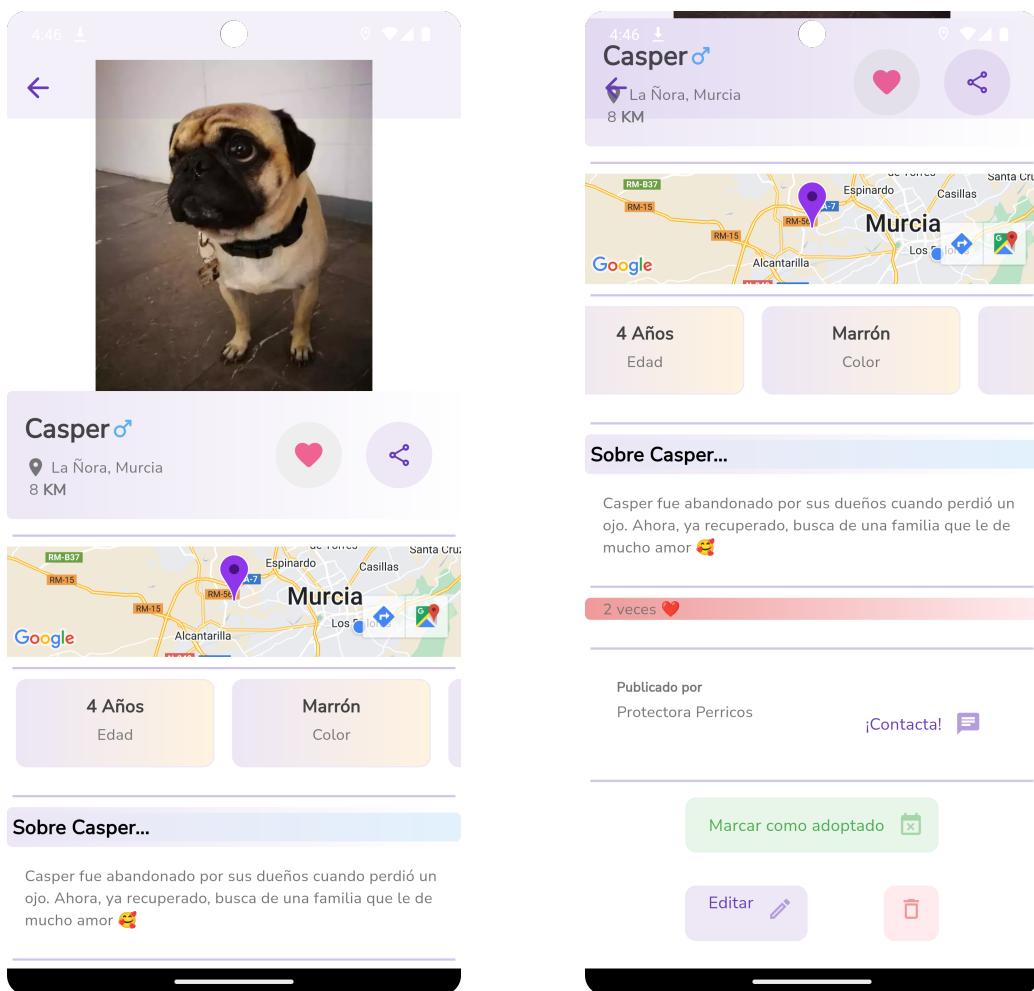


Figura 77: Pantalla de perfil de perro.

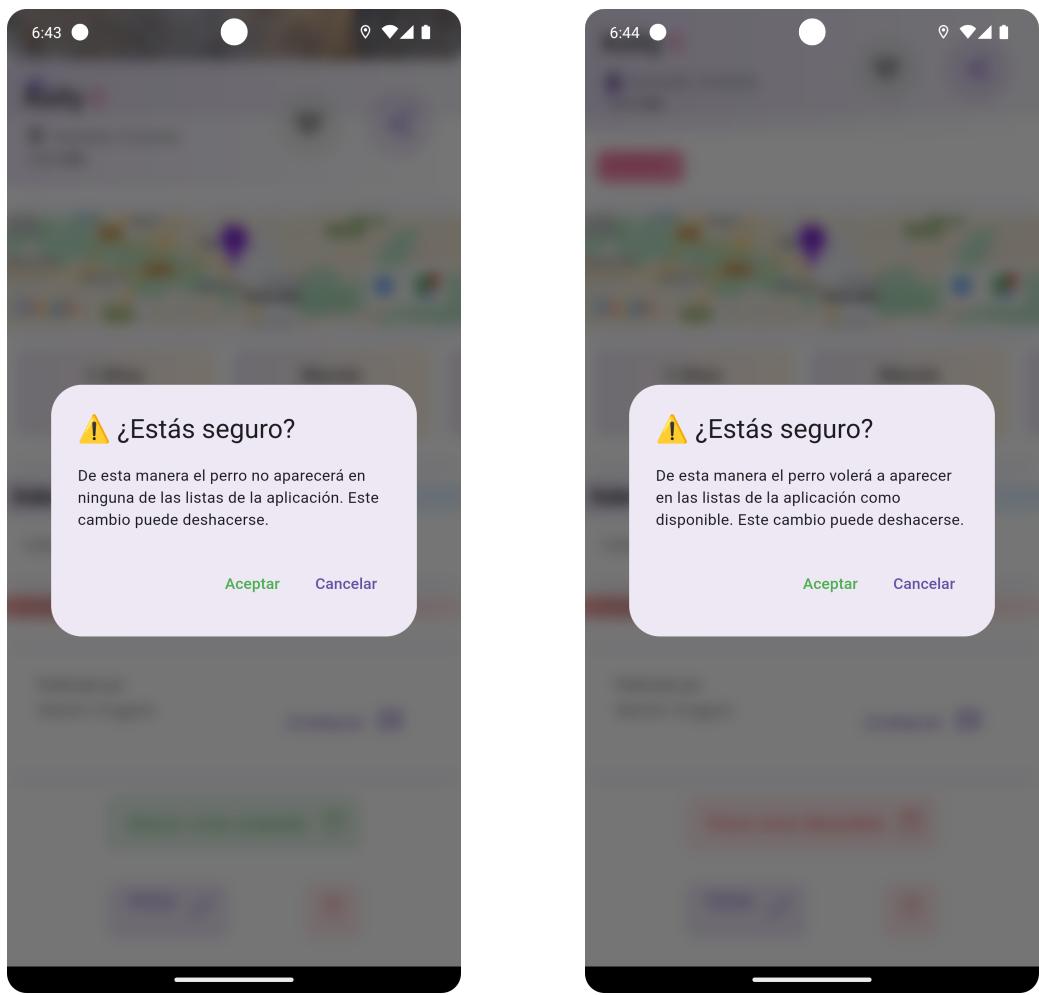


Figura 78: Pop-up de marcar/desmarcar como adoptado.

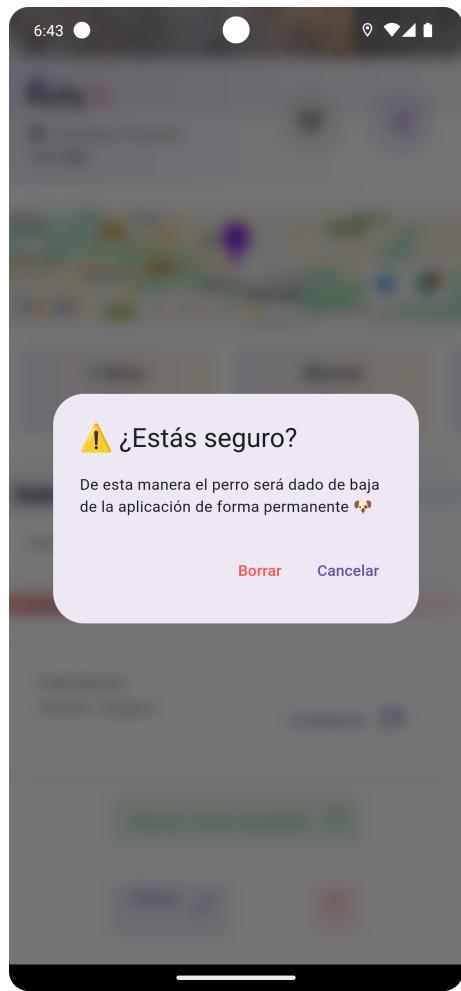


Figura 79: Borrar perro.

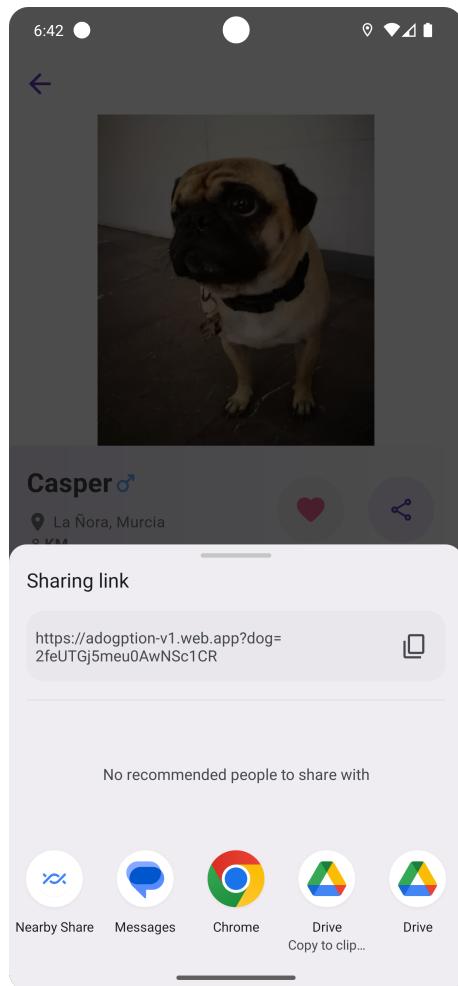


Figura 80: Compartir perro.

Actualizar datos de perros

Los formularios para actualizar datos de perros son casi idénticos a los proporcionados para darlos de alta. En primer lugar, los formularios cargan todos los datos actuales del perro y permite que se introduzcan todos los cambios que se requieran. Posterior a confirmar los cambios, si la actualización es correcta, se muestra una notificación indicándolo.

The figure consists of two side-by-side screenshots of a mobile application interface for updating dog data. Both screens have a light blue header bar with a back arrow icon. Below the header is a circular profile picture of a black dog named Pedro. The main area is titled "Datos del canino".

Left Screen (Initial Data):

- Nombre: Pedro (with a paw print icon)
- Raza: Akita (with a paw print icon)
- Color: Negro (with a paw print icon)
- Edad: 4 (with a candle icon)
- Peso: 13 (with a scale icon)
- Género: Macho (with a male symbol icon)

Right Screen (Updated Data):

- Nombre: Pedro (with a paw print icon)
- Raza: Akita (with a paw print icon)
- Color: Negro (with a paw print icon)
- Edad: 4 (with a candle icon)
- Peso: 13 (with a scale icon)
- Género: Macho (with a male symbol icon)
- Checkboxes:
 - Para adoptar: checked (with a checkmark icon)
 - Para acoger: unchecked (with an empty box icon)
 - Castración hecha: checked (with a checkmark icon)
- Descripción: ¡Es un perro muy animado!
- Bottom button: Confirmar (Confirm)

A small note at the bottom of the right screen states: * Para que aparezca en las listas por lo menos una opción de esta sección tiene que estar marcada (At least one option in this section must be checked for it to appear in the lists).

Figura 81: Formulario de actualización.

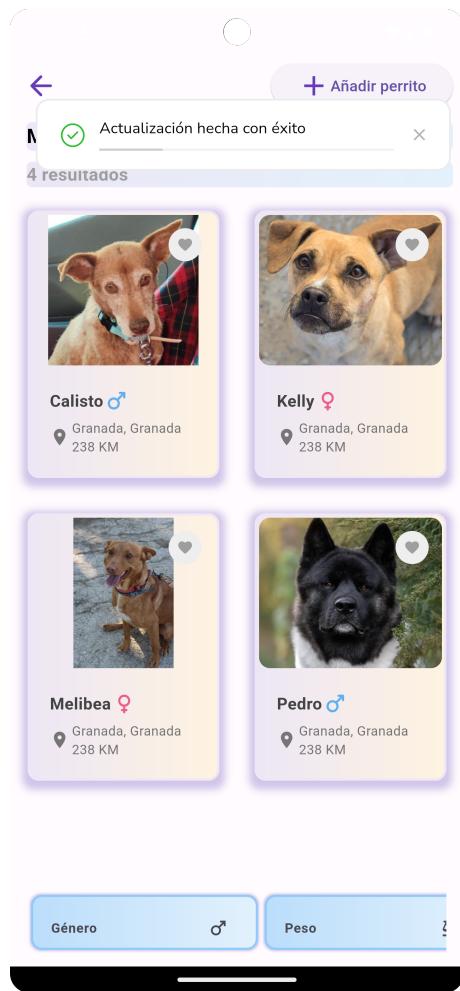


Figura 82: Notificación de actualización correcta.

Chats

La estructura de la pantalla de los chats es muy similar a cualquier otra aplicación de mensajería, se incluye una lista de chats que permiten ser borrados deslizando el widget. Cuando se hace click en alguno de estos widgets, se redirige a la pantalla del chat específico.



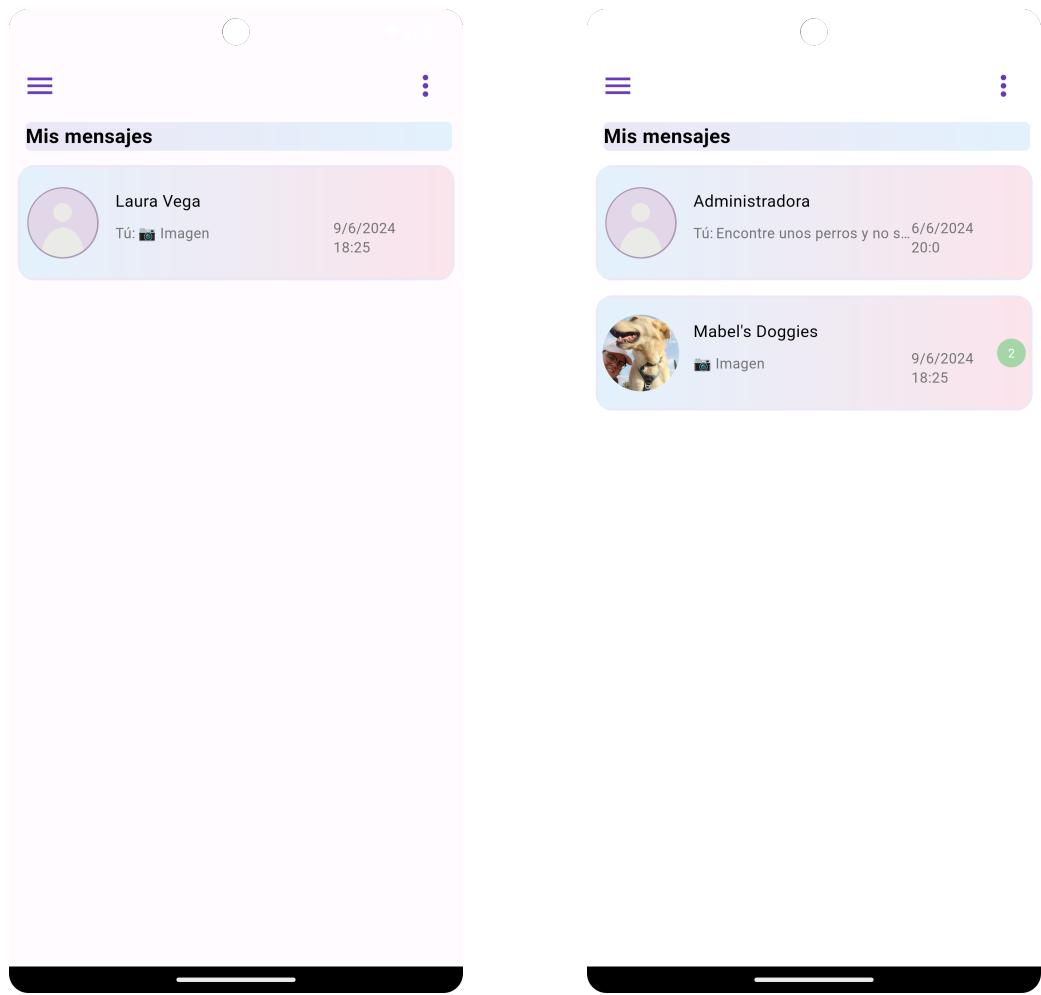


Figura 83: Pantalla de chats disponibles.

Información legal

La pantalla de información legal recoge los términos y condiciones de uso además de toda la información de política de privacidad de la aplicación.

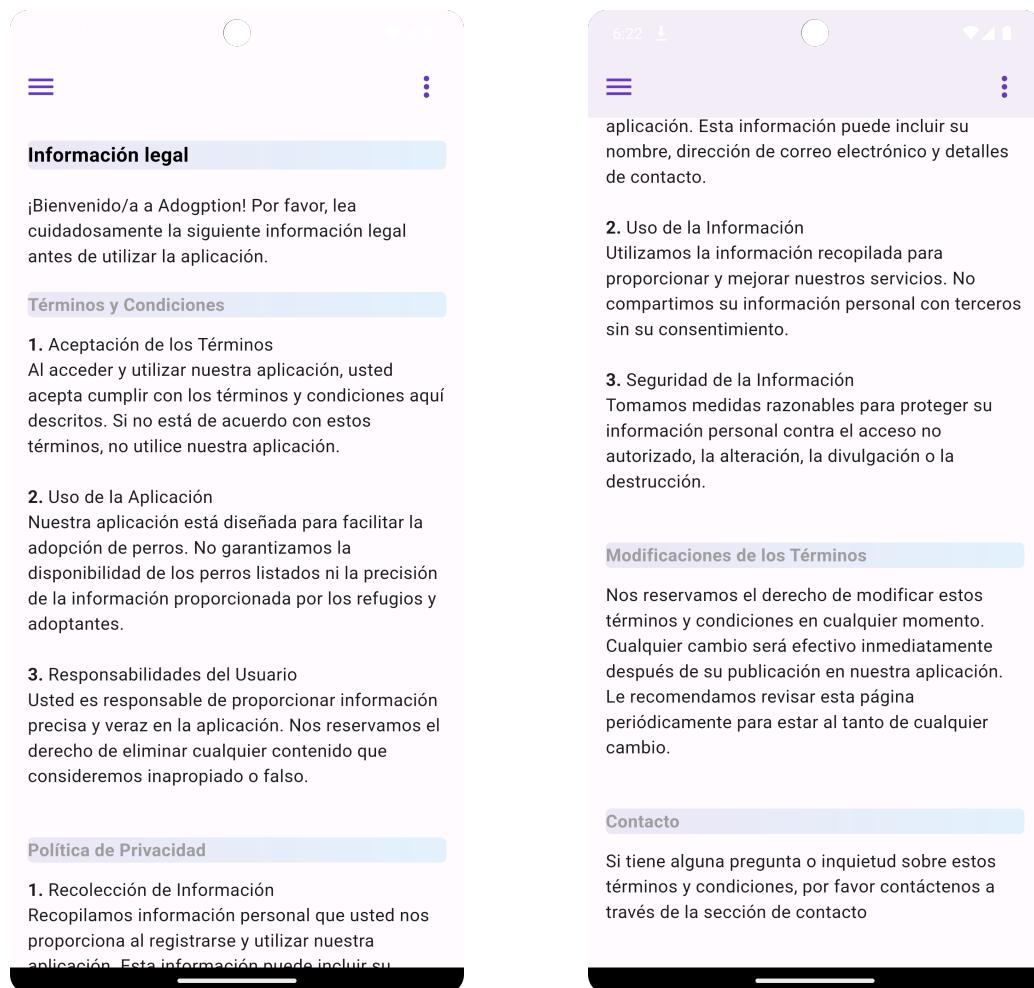
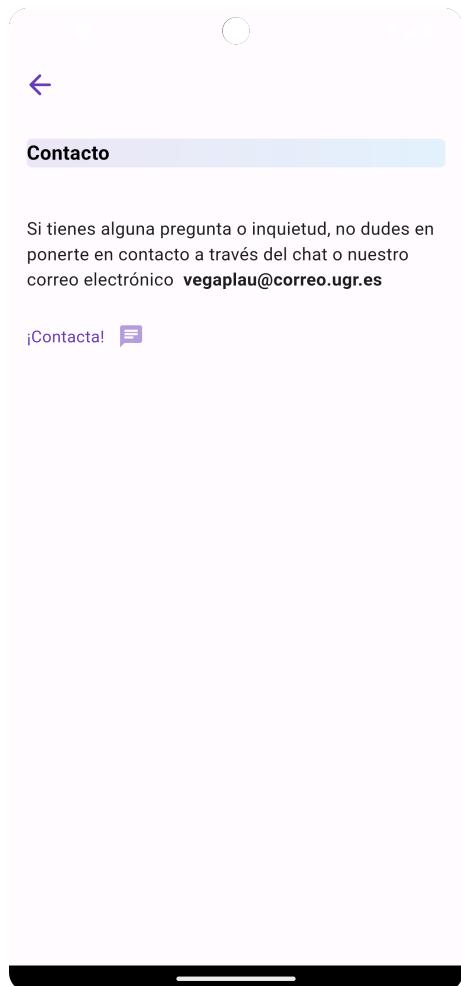


Figura 84: Pantalla de información legal.

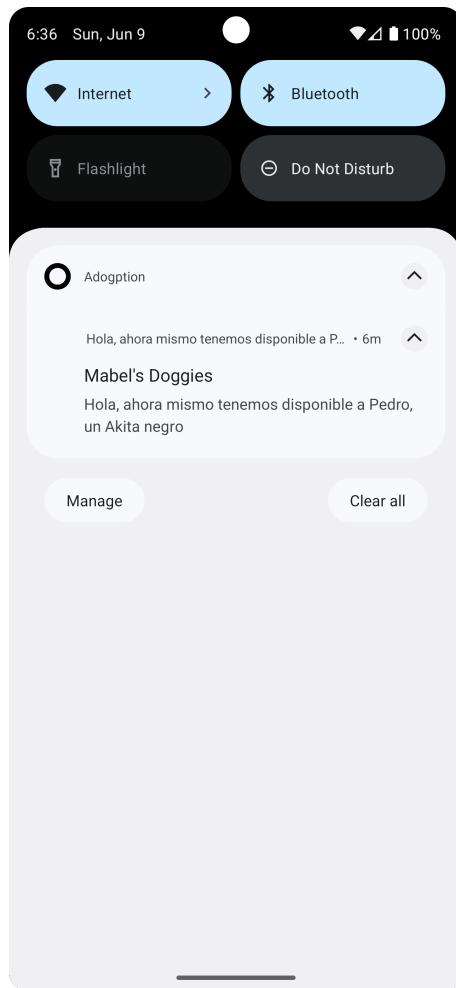
Contacto

En la pantalla de contacto se incluye un correo de contacto además de un botón que abre un chat con un administrador de la aplicación.



Notificaciones

Las notificaciones se lanzan cuando un usuario recibe un mensaje de otro usuario.

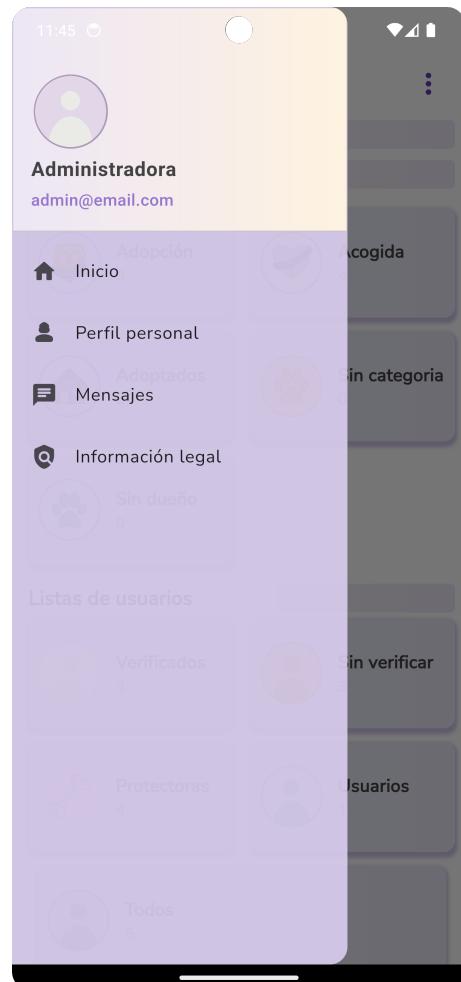


Vista de administrador

La vista de administrador es la más simple, contiene diferentes listas para manejar los usuarios y perros que están dados de alta en la aplicación. Puede editar datos de cualquier usuario o perro además de verificar y desverificar perfiles de usuario.

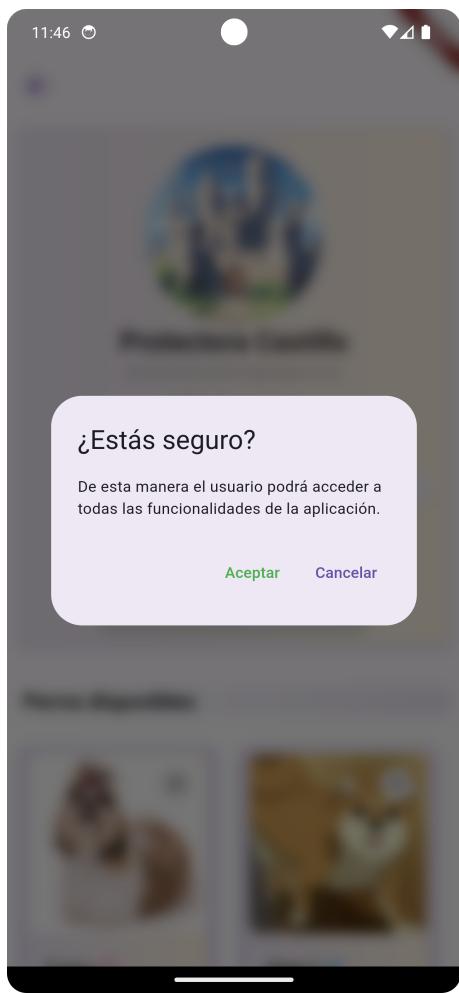


(a) Pantalla inicio administrador

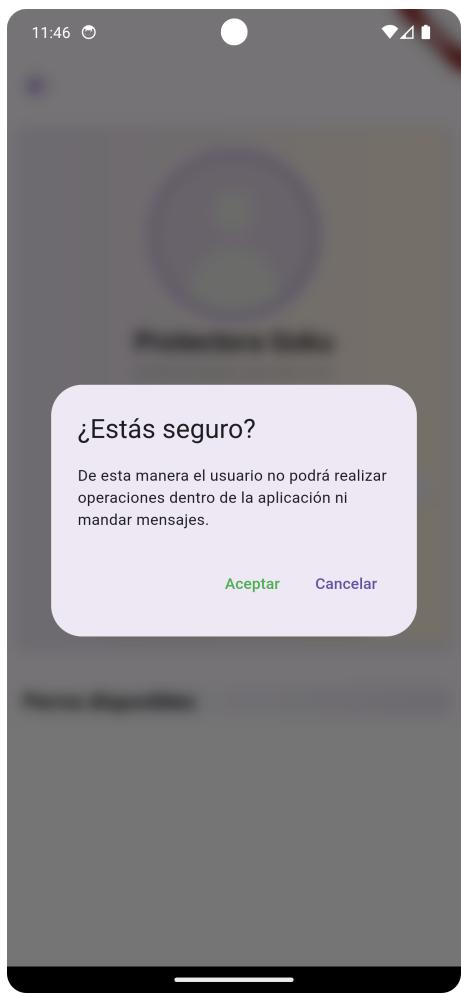


(b) Menú lateral administrador.

Figura 85: Pantalla inicio de administrador.



(a) Pop-up verificar.



(b) Pop-up desverificar.

Figura 86: Verificar/desverificar usuario.

6. Conclusiones

Tras completar el período de desarrollo de la aplicación, en este apartado se comprueba la completitud de todos los requisitos funcionales.

Se implementó el registro de usuarios por rol además de proporcionar diferentes vistas para cada uno de los roles, que finalmente se definieron como *Usuario*, *Protectora* y *Administrador*. Con relación a los usuarios, también se añadieron diferentes listas a lo largo de la aplicación, que finalmente incluyeron una barra de búsqueda para filtrar resultados. Además, para los usuarios se implementó una página de perfil en la que se puede acceder a la edición de datos y credenciales. Por otra parte, también se implementó el registro y edición de perros y se incluyeron diferentes listas en la aplicación, que aparte de una barra de búsqueda también incluyen los filtros de *Peso*, *Raza*, *Género* y *Color*. Además, en los perfiles de los perros se añadieron botones para compartir y marcar como favoritos, este último vino de la mano de la implementación de la sección de favoritos para los usuarios. Se crearon diferentes páginas adicionales, como la página de contacto o la de información legal, además de la página de mapas, en la que se incluyó un listado de resultados e hizo falta integrar las APIs de geolocalización. Esta misma API de geolocalización también se utilizó para desarrollar una barra de búsqueda de direcciones. Por último, se implementó un chat en tiempo real al que se le añadió la opción de mandar imágenes con sus correspondientes notificaciones.

Al revisar la lista de objetivos, se concluye que la aplicación cumple con todas las funcionalidades obligatorias. Sin embargo, algunos requisitosopcionales no se completaron en esta versión. Específicamente, el blog y el tema oscuro no se implementaron. El blog resultó ser demasiado ambicioso y no se pudo completar en el tiempo estipulado, ni si quiera dio tiempo a definir una estructura de datos para este, mientras que el tema oscuro se pospuso debido a retrasos en la implementación de otras funcionalidades obligatorias. Estos requisitos pendientes se considerarán para futuras versiones de la aplicación.

Para futuras iteraciones, se han planteado algunas propuestas para añadir valor a la aplicación y mejorar la experiencia de uso del usuario.

- Añadir la posibilidad de crear listas con filtros a las protectoras.
- Permitir subir múltiples imágenes para los perros dados de alta, además de vídeos y poder generar un álbum por cada perro.
- Permitir asignar a un usuario como adoptante/acogedor de un perro, además de indicar en el perfil del perro quien es su adoptante/acogedor.
- Sumar las funcionalidades mandar mensajes de audios y vídeos dentro del chat
- Añadir como atributos opcionales de las protectoras distintas redes sociales

e incluso el número de teléfono.

- Permitir compartir perfiles de protectoras.
- Integrar los servicios de Google para iniciar sesión.
- Incluir más filtros para las listas de caninos.

Todas estas propuestas buscan mejorar la experiencia de uso de la aplicación, algunas de ellas surgen a partir del feedback proporcionado por los usuarios que han probado la aplicación. Además, aunque la aplicación está diseñada inicialmente para facilitar las adopciones de perros, se contempla la posibilidad de incluir otros tipos de animales, como gatos, en futuras actualizaciones.

7. Bibliografía

Referencias

- [1] F. Affinity. (2022) Las cifras del abandono de perros y gatos. [Online]. Available: <https://www.fundacion-affinity.org/perros-gatos-y-personas/busco-un-animal-de-compania/las-cifras-del-abandono-de-perros-y-gatos-aun>
- [2] BOE. (2023) Ley de protección de los animales. [Online]. Available: <https://www.boe.es/buscar/doc.php?id=BOE-A-2023-7936>
- [3] P. Latino. (2024) Animales que sufren por el comercio de mascotas. [Online]. Available: <https://investigaciones.petalatino.com/animales-sufren-comercio-mascotas/>
- [4] Flutter. (2024) Documentación de Flutter. [Online]. Available: <https://flutter.dev>
- [5] Dart. (2024) Documentación de Dart. [Online]. Available: <https://flutter.dev>
- [6] Firebase. (2024) Consola de Firebase. [Online]. Available: <https://console.firebaseio.google.com>
- [7] ——. (2024) Precios de Firebase. [Online]. Available: <https://firebase.google.com/pricing?hl=es-419>
- [8] Google. (2024) Precios de APIs de Google. [Online]. Available: <https://mapsplatform.google.com/intl/es/pricing/>
- [9] A. desconocido, “search_map_location,” https://pub.dev/packages/search_map_location, 2024.