UNIVERSIDAD DE GRANADA

Universidad de Granada

Ingeniería Informática

Trabajo fin de grado

Planificación y desarrollo de una app para adopción canina

Autora

Laura Vega Palacios

Tutor

Juan José Escobar Pérez

Junio de 2024

Resumen

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 caninas. Centralizando la información de las protectoras y facilitando la comunicación entre los adoptantes y las protectoras.

En esta aplicación se pueden dar de alta protectoras o refugios, con sus correspondientes datos y ubicación. Las protecotras o refugios que estén verificadas por los administradores, tendrán la capacidad de subir a la plataforma todos los caninos que quieran, estos aparecerán en las listas de adopción u acogida dependiendo de lo que se haya marcado para cada canino. También se podrán dar de alta usuarios que busquen adoptar o acoger algún perro.

Existen los formularios correspondientes para dar de alta o editar la información del perro, con distintos apartados para cumplimentar todos los datos requeridos como puede ser la descripción, la raza, edad etc. Los usuarios también disponen de formularios para darse de alta y editar su información.

La aplicación utiliza un sistema de geolocalización para mostrar a los usuarios resultados dentro de la aplicación según distancia. Se podrán ver los resultados de la aplicación en un mapa insertado en la aplicación. También se incluyen barras de búsqueda y filtros rápidos para acotar resultados según los requisitos del usuario.

Se incluye dentro de la aplicación un sistema de mensajería para facilitar la comunicación dentro de la misma, tanto entre usuarios y protectoras como con los administradores. El chat permite mensajes de texto e imágenes.

Se añade, además, un sistema de favoritos y la posibilidad de compartir caninos de la aplicación de forma externa a través de enlace.

La aplicación incluirá una página de información legal y de contacto a disposición del usuario.

Abstract

The development of this *Final Degree Project* aims at planning and developing a mobile application for managing dog adoptions. Centralizing information from shelters and facilitating communication between adopters and shelters.

In this application, shelters or rescues can be registered, along with their corresponding data and location. Shelters or rescues that are verified by administrators will have the ability to upload all the dogs they want to the platform, which will appear in adoption or fostering lists depending on what is marked for each dog. Users who are looking to adopt or foster a dog can also register.

There are corresponding forms for registering or editing dog information, with different sections to fill in all the required data such as description, breed, age, etc. Users also have forms to register and edit their information.

The application uses a geolocation system to show users results within the application based on distance. Results can be viewed on a map embedded in the application. Quick search bars and filters are also included to narrow down results according to user requirements.

A messaging system is included within the application to facilitate communication, both between users and shelters and with administrators. The chat allows text messages and images.

Additionally, a favorites system is added, along with the ability to share dogs from the application externally through a link.

The application will include a legal information and contact page available to the user.

A mis tíos, que me dieron la oportunidad y confiaron siempre

Agradecimientos aqui

Índice

Índice de cuadros

Índice de figuras

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 estas en todo el mundo. Existen estudios que indican que casi 300.000 perros y gatos fueron recogidos durante el 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 en la que se trata 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.
- 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 conciencación 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 constan de las condiciones necesarias para vivir adecuadamente. Algunos carecen de cuidados veterinarios, de correcta alimentación y de higiene. En algunos artículos podemos ver 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.

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 organización:

- **Elección del perro:** en el que se deberá tener en cuenta las necesidades de la mascota para ver cual se adapta mejor al estilo de vida del hogar donde va

a ir. Lo normal en este paso es interactuar con diferentes mascotas candidatas a ser adoptadas.

- **Solicitud de adopción:** es común que se tenga que cumplimentar un formulario y una posterior entrevista para asegurar que el animal va a ir a un hogar adecuado.
- **Evaluación del hogar:** lo normal es organizar una visita al hogar donde va a ir para asegurarse de que el entorno es seguro y adecuado, también verificar que se tienen ya todo lo necesario para recibir una mascota como comederos, juguetes, cama etc.
- **Contrato de adopción:** es indispensable para garantizar protección legal a las mascotas adoptadas. El nuevo dueño tiene que comprometerse legalmente a garantizar que el animal va a tener atención veterinaria y que no va a ser abandonado entre otros puntos. En la mayor parte de organizaciones es común llevar a vacunar y a poner el chip al animal antes de que vaya a casa del nuevo dueño.
- **Recogida del animal:** el día en el que se recoge la mascota, se recibe todo su hisotrial médico (si se tiene) y la organización suele dar consejos para los nuevos dueños.
- **Seguimiento:** posterior a la adopción es requerido un seguimiento tanto para garantizar el bienestar del animal como para dar apoyo o asesoramiento al dueño de este.

Todo este proceso es guiado por la correspondiente organización y es importante que se cumplimente, aunque pueda hacerse algo pesado de seguir. Para realizar este proceso de adopción, previamente la persona deberá ponerse en contacto con las distintas protectoras o refugios en su zona. Esta fase puede generar diferentes problemas a los adoptantes. En la mayoría de protectoras, trabajan voluntarios y se sostienen de donaciones, es por ello que es común que no consten de fondos para poder generar visibilidad de la organización. Muchas de las protectoras utilizan diferentes redes sociales o webs propias para promocionarse, lo que implica que una persona que quiere adoptar podría que tener utilizar diferentes plataformas para poder contactar con alguna. Debido a que muchas veces se utilizan redes sociales para contactar, a veces es difícil mantener una comunicación adecuada con diferentes protectoras, o incluso post adopción. Otro problema es que los futuros adoptantes pueden encontrar dificultades escogiendo mascota, y es que normalmente la información de las diferentes mascotas está dispersa y puede convertirse en una tarea tediosa buscar una que cumpla todos los requerimientos.

Después de ver cuales son los pasos y posibles inconvenientes para aplicar

a un proceso de adopción, se propone un nuevo proyecto a modo de aplicación móvil. Esta aplicación se propone como una solución a la fase inicial de buscar protectoras por la zona, facilitando a los usuarios listas de protectoras cercanas y de los perros de los que consta, incluyendo filtros para acotar la búsqueda. Se propone también como una manera de facilitar a las protectoras gestionar a sus perros e interesados. Se añade a la solución un chat para establecer comunicación directa entre usuarios y facilitar la misma. 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. Estructura del documento

La estructura del documento es la siguiente

- **Resumen:** Se incluye un breve resuemn de la funcionalidad del proyecto, tanto en español como en inglés.
- **Introducción:** Breve introducción al proyecto, donde se incluyen varios puntos
  - Motivación del proyecto
  - Estructura del documento
  - Objetivos propuestos para la realización del proyecto
- **Planificación:** Incluye la planificación del proyecto, teniendo en cuenta las diferentes etapas del mismo y también los presupuestos.
- **Análisis:** Aspectos más relevantes relacionados con los requisitos del sistema, además de los casos de uso y flujos de sistema.
- **Diseño:** Estructura y diseño inicial donde se proponen las clases e interfaz de usuario que posteriormente serán implementadas.
- **Implementación:** Se trata con profundidad todos los aspectos relacionados con el desarrollo del proyecto. Se habla de los problemas que hayan podido aparecer además de las soluciones finales, que pueden diferir de las propuestas en el diseño.
  - Herramientas y tecnologías utilizadas. Un breve resumen en el que se habla de la tecnología que se ha escogido para implementar el proyecto.
  - Desarrollo y sus fases. Se exponen todos los pasos seguidos durante el desarrollo de la aplicación.
  - Diseño final y funcionalidad. Incluye imágenes del resultado final de la aplicación además de su funcionalidad.
- **Conclusiones:** Se incluye una conclusión acerca de la viabilidad del proyecto y futuro del mismo. Además tambien se añade un resumen y valoración personal del proyecto y de todas las fases que ha tenido.
- **Bibliografía:** Fuentes de información utilizadas durante cualquier fase del proyecto, tales como vídeos, wikis, artículos, etc.
- **Anexos:** Cualquier otro documento relevante en el desarrollo del proyecto.

1.3. Objetivos del proyecto

Hemos hablado anteriormente de unas necesidades y motivación que ha llevado a la propuesta de este proyecto. Para poder cubrirlas, se han de definir unos objetivos. Estos objetivos podrán ser obligatorios, que serán los requeridos para que la aplicación funcione como se espera u opcionales, para añadir funcionalidades extra o facilitar el uso de la aplicación. Una vez realizado el proyecto, se volverá a hacer una lista similar, para comprobar si se han cumplido o no los objetivos, y en caso de que no, el porqué.

Objetivo 1

Título	*Registro de usuarios con diferentes roles*
Tipo	Obligatorio
Descripción	La aplicación deberá proporcionar la posibilidad de darse de alta como usuario o como protectora. Deberá propocionar los formularios correspondientes además de almacenar los datos en la base de datos.

Objetivo 2

Título	*Registro de caninos*
Tipo	Obligatorio
Descripción	La aplicación deberá proporcionar la posibilidad de dar de alta a distintos perros con diferentes características a los usuarios registrados como protectoras.

Objetivo 3

Título	*Listas de caninos con filtros y búsqueda*
Tipo	Obligatorio
Descripción	La aplicación debe mostrar diferentes listas de perros y permitir aplicarle filtros rápidos y realizar búsquedas.

Objetivo 4

Título	*Listas de usuarios con búsqueda*
Tipo	Obligatorio
Descripción	La aplicación deberá mostrar diferentes listas de usuarios que permita realizar búsquedas en ellas

Objetivo 5

Título	*Sección de perros favoritos*
Tipo	Obligatorio
Descripción	Incluir un botón de favoritos en los perfiles de los perros que permita a los usuarios añadir a su sección de favoritos los perros que marquen como favoritos.

Objetivo 6

Título	*Sistema de mensajería entre usuarios*
Tipo	Obligatorio
Descripción	Permitir a los usuarios abrir un chat con otros usuarios dentro de la aplicación. Los usuarios deberán recibir una notificación cuando reciban un mensaje.

Objetivo 7

Título	*Geolocalización y mapas con resultados*
Tipo	Obligatorio
Descripción	Incluir el uso de la ubicación para ordenar los resultados de la aplicación según la distancia del usuario. Además, se incluir una página con un mapa con la ubicación de los resultados y una lista de los mismos.

Objetivo 8

Título	*Actualizar datos de caninos*
Tipo	Obligatorio
Descripción	Permitir actualizar los datos de los caninos, tanto como el peso, edad, descripción etc. Además de marcar si un perro está disponible para adoptar/acoger o si este ya ha sido adoptado.

Objetivo 9

Título	*Actualizar datos de usuarios*
Tipo	Obligatorio
Descripción	Permitir a un usuario actualizar sus datos personales así como credenciales.

Objetivo 10

Título	*Barra de búsqueda de direcciones*
Tipo	Opcional
Descripción	Permitir a un usuario buscar su dirección y autocompletar los diferentes campos del formulario atuomáticamente.

Objetivo 11

Título	*Tema oscuro en la aplicación*
Tipo	Opcional
Descripción	Permitir al usuario cambiar a tema oscuro dentro de la aplicación

Objetivo 12

Título	*Blog*
Tipo	Opcional
Descripción	Permitir al usuario añadir posts con imágenes y textos además de añadir comentarios en los posts, para compartir ideas con otros usuarios.

Objetivo 13

Título	*Compartir perros a través de enlace*
Tipo	Opcional
Descripción	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	*Mandar imágenes dentro del chat*
Tipo	Opcional
Descripción	Permitir mandar imágenes dentro del chat.

2. Planificación

Consideramos la planificación una de las etapas cruciales para que el proyecto, no solo para que se realice correctamente, sino garantizar que la solución sea sostenible y escalable a largo plazo. La planificación consta de diferentes fases, cada una con objetivos específicos para garantizar que se abarcan todos los aspectos relevantes en el proyecto.

2.1. Análisis

En esta fase, se identicican tanto los recuros de la aplicación como todos los requisitos de la misma. Esta toma de decisiones está basada en las necesidades de la aplicación y los usuarios a los que está dirigida.

2.1.1. Definición y especificacíón de requisitos

En esta fase se estudian y se definen los diferentes requisitos que debe cumplir la aplicación a partir de los objetivos definidos anteriormente.

- Los requisitos funcionales son aquellos que la aplicación debe poder hacer, como por ejemplo, mostrar listas de perros o tener un chat.
- Los requisitos no funcionales son aquellas condiciones bajo las cuales debe operar la aplicación, como rendimiento, seguridad.

2.1.2. Recursos

En esta fase se identifican y asignan los recursos encesarios para el desarrollo y ejecución del proyecto. Los recursos están divididos en diferentes tipos.

- Los recursos humanos, que se refieren a todas las personas que se requieren y van a estar involucradas durante el desarrollo del proyecto.
- Los recursos tecnológicos, que engloban tanto el hardware como el software que va a estar involucrado en el proyecto. Se pueden incluir tanto servidores, equipos de trabajo, como diferentes herramientras de diseño, bases de datos etc.
- Recursos financieros, que incluyen sueldos, costes de licencias, gastos en hardware entre otros.
- Recursos de tiempo, en los que se definen los plazos y la planificación de sprints o ciclos de desarrollo.

2.2. Diseño del proyecto

Durante el diseño del proyecto se tienen que barajar diferentes soluciones, se escoge la que más se ajuste a los recursos y requisitos definididos anteriormente. En esta fase además, es en la que se garantiza la robustez y escalabilidad de la aplicación.

2.2.1. Diseño de base de datos

El diseño de la base de datos implica definir la estructura en la que se almacenarán los datos para asegurar su integridad, accesibilidad y rendimiento. Durante el diseño de la base de datos se tienen en cuenta las partes esenciales como son las entidades y sus atributos, las relaciones entre entidades y las consideraiones de integridad y seguridad.

2.2.2. 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. Se definen bocetos y prototipos interactivos que cumplen con los requisitos funcionales establecidos. Además, se define un estilo y tema que se debe mantener a lo largo de todos los bocetos y prototipos. Es importante tener en cuenta que el diseño sea responsive para que se adapte adecuadamente a todos los dispositivos móviles.

2.2.3. Arquitectura de sistema

Se define la estructura y la organización de la aplicación, así como las tecnologías que finalmente se van a utilizar. A la hora de definir esta arquitectura, se tienen en cuenta el backend, el frontend, los microservicios y el despliegue de la aplicación.

2.3. Implementación

En esta fase se hace el desarrollo e implementación de la aplicación, engloba diferentes apartados ya que es la más extensa de todas. A lo largo de la fase se fueron completando los diferentes requisitos definidos anteriormente, los cuales espeficiacremos más adelante. También en esta fase, es necesario el estudio de las diferentes tecnologías usadas, así como de bibliotecas que han sido necesarias a lo largo del desarrollo.

2.4. Pruebas y correcciones

Posterior al desarrollo, se hacen diferentes pruebas y correcciones que sean necesarias a la aplicación. En esta fase se pueden detectar faltas de funcionalidad o bugs que se pueden añadir para completar en futuras iteraciones del proyecto.

2.5. Documentación y memoria

En todo el desarrollo del proyecto, se ha ido recopilando información de todas las fases que ha tenido. Posterior a la finalización del mismo, se comienza la memoria en la que se recogerá toda la información almacenada.

2.6. Diagrama de Gannt

Para condensar toda esta información se ha generado un Diagrama de Gannt, con comienzo en el mes de Enero de 2024 y que finaliza en Junio del mismo año.

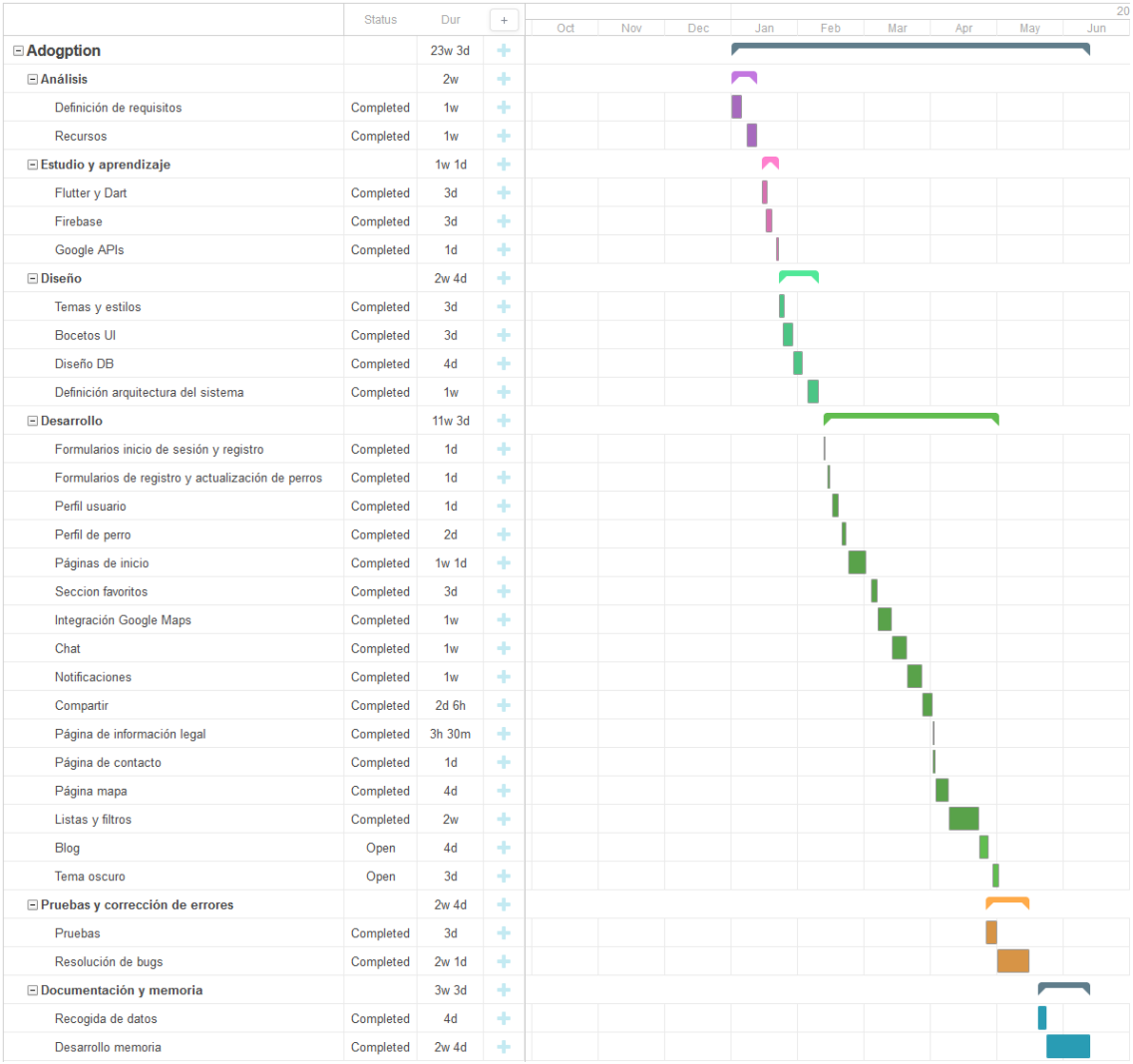

Figura 1: Diagrama de Gannt

3. Análisis

3.1. Definición y especificación de Requisitos

Para garantizar buenos resultados a la hora de desarrollar la aplicación, es esencial definir y comprender los requisitos que guiarán la creación y el diseño del sistema. Estos requisitos se dividen en dos categorías principales: requisitos funcionales y requisitos no funcionales.

En esta sección, se detallan tanto los requisitos funcionales como los no funcionales identificados para la aplicación en desarrollo.

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.

Explicación	Proporcionar formularios 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.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	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..

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	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 inferiro de la página de inicio de los usuarios.

RF11: Mostrar adopciones recientes.

Explicación	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.

Explicación	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.

Explicación	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.

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

RF15: Mostrar datos personales en el menú lateral.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	Propocionar 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.

Explicación	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.

Explicación	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 correspondinte, incluyendo foto de perfil, correo y los perros (si los tiene). Además incluirá el botón de contacto para abir chat.

RF27: Mostrar página de perfil de canino.

Explicación	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 bóton 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.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	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 ctualizar 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.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	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.

Explicación	Proporcionar una interfaz que rehúna 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.

Explicación	Proporcionar una interfaz que rehúna 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 inició según rol.

Explicación	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.

Explicación	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.

Explicación	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.

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. Aquí se listan las propuestas a tener en cuenta a la hora de desarrollar.
  - 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 denbe 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 desarrollen 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, como tiene un rango de usuarios muy amplio, se espera que un usuario poco experimentado sea capaz de usar la aplicación de form 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 nuestro sitio 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 correciones.
- **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 los 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. Recursos

Posterior a definir todos los requisitos de la aplicación, se pude concretar que va a ser necesario utilizar durante el desarrollo del proyecto, tanto mano de obra, como hardware y software.

3.2.1. 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). Es encargada de cumplimentar todos los puntos del desarrollo. El horario laboral sería de Lunes a Viernes de media jornada.

3.2.2. Hardware

Para este proyecto el hardware es un ordenador de sobremesa personal

- 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.

3.2.3. 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 prinicpalmente para el desarrollo de aplicaciones del lado del cliente. Es un lenguaje bastante popularizado, uno de los motivos de optar por este lenguaje, es el manenimiento que recibe a día de hoy y por la simple que resulta para programar. Para añadir funcionalidades extra, se han escogido varias bibiliotecas que ofrecen algunos microservicios o widgets.

Por último, para manejar la base de datos, entre otros, 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 convertia en el candidato adecuado para manejar el chat, la funcionalidad de compartir entre otras cosas 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
- **Lenguaje:** Dart 3.3.0 - Dart DEV
- **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`
  - `awesome_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

3.3. Costes

Con la lista de recursos necesarios, ya se puede hacer un presupuesto de la aplicación adecuado según los diferentes apartados.

3.3.1. Recursos humanos

Para calcular el coste relacionado con el personal, se toma como referencia el sueldo medio de la persona involcurada en su trabajo actual, teniendo en cuenta que cobra entre los *38.000€* - *43.000€* anuales.

Tomando como media los *40.000€* anuales, al mes corresponden unos *3.330€*. Como la duración del proyecto es de 5 meses y medio, el coste total en salarios para la empleada sería de *18.315€*.

3.3.2. Hardware

Debido a que el equipo utilizado no es nuevo, consideraremos los gastos de estos teniendo en cuenta que un equipo informático se puede amortizar hasta 8 años. El precio aproximado del orenador es de unos *1000€*. Un año tiene 12 meses, así que tenemos 96 meses en total de amortización, lo que resulta a unos *11€* al mes. Como la duración del proyecto es de 5 meses y medio, el coste del ordenador sería de *60,50€*.

El precio de la tablet es de unos *100€* así que con los mismos cálculos, el gasto de la tablet es de *5,72€* en total.

3.3.3. Software

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

- Firebase - Plan Spark - Guía de planes
- Google APIs - Ofrecen créditos gratuitos de hasta *200€* para cubir los gastos que se generen por el uso de la API. - Precios
- IDE + Framework - Gratuitos

3.3.4. 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€*, calculamos que van dirigidos a esta parte unos *165€* extra.

Recurso	**Coste (€)**
Personal	18.315
Hardware	Ordenador - 60,50 Tablet 5,72
Software	0
Total	**18.381,22€**

Costes del proyecto

4. Diseño

En esta parte tenemos que definir 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. Debemos almacenar distintas entidades y propiedades, consideramos las siguientes entidades y sus correspondientes tablas.

4.1.1. Direcciones

Para las protectoras almacenamos la dirección en la que están ubicadas. Esta dirección es necesaria para mostrar al usuario el texto de donde está y también para ubicarla en los mapas que se proporcionan en la app. Por una parte almacenamos la dirección por partes, incluyendo la ciudad, la calle, el código postal etc. Por otra parte, cuando generemos una dirección, a su vez guardaremos las coordenadas. Esta decisión se toma para evitar hacer cálculos repetidos cada vez que tengamos que cargar un mapa con marcadores, ya que estos requieren de las coordenadas geográficas directamente. Cuando se actualice una dirección, se actualizaran sus coordenadas.

Para todas las direcciónes se genera un ID automático, al que se hace referencia desde la tabla de usuarios.

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

Aqui 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 generar la tabla de chats, se decidió almacenar los ids de los usuarios implicados, aparte de para quién esta disponible el chat en ese momento. Esto es debido al requisito definido previamente en el que se define la posibilidad de eliminar un chat. Al igual que en otras aplicaciones de mensajería, cuando se elimina un chat, el chat simplemente desaparece para el usuario que lo borra, así que podemos actualizar la propiedad de para quien está visible el chat en ese momento. Para esta primera iteración del proyecto, los mensajes no se borran, así que si por ejemplo una persona A, borra su chat con la persona B, si vuelve a abrir un chat con esa persona, los mensajes anteriores aparecerán, en la parte de implementación de la memoria indagaremos más en profundidad acerca del comportamiento definido.

Almacenar aqui 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.

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

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

```
{
  "availableTo": [
    "kuaKjc6XoiVot1rMeGqBySA5pE13",
    "buO4Vkazs3O2oVKetugxrvTfRw12"
  ],
  "lastMessage": "BaKmuuo8debUTdSNsvw0",
  "users": [
    "kuaKjc6XoiVot1rMeGqBySA5pE13",
    "buO4Vkazs3O2oVKetugxrvTfRw12"
  ]
}
```

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.

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": "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": "S4wXwckahiSOLI0Q3exXDVRh2yl2",
  "profilePic": "",
  "weight": 1,
  "weightUds": "KG"
}
```


4.1.4. Mensajes

Aparte de almacenar toda la información de los chats, hay que almacenar los mensajes involucrados en los chats. Se propone guardar el tipo de mensaje, habiendo en esta primera iteración solo dos tipos, que serían **default** e **image**. Esta distinción es necesaria para cuando se cargue la preview del mensaje en la UI. Aparte, si el mensaje que se envía es una imagen, se generara una entrada para esa imagen.

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

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 recpetor 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

Aqui 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": "buO4Vkazs3O2oVKetugxrvTfRw12",
  "type": "default"
}
```


4.1.5. Notificaciones

Cuando un usuario manda un mensaje, al usuario receptor le tiene que llegar una notificación. Lo cierto es que para generar esa notificación se pueden seguir varios caminos. Para las notificaciones de los mensajes, se decidió crear una table que contenga las requests de notificaciones para un usuario en concreto. Cuando se inicia sesión, se añade un listener que escucha todos los cambios que se produzcan en esta tabla, cuando llega una request nueva, se lanza una notificación a través del canal correspondiente, posteriormente esta request se marca como vista.

...

4.1.6. Usuarios

Los usuarios son la otra entidad con gran relevancia dentro de la aplicación. Sin embargo, la información que se almacena de estos es escueta. También se tiene en cuenta que el servicio de *Autenticación* de Firebase se encarga de proporcionar el almacenamiento de muchos de los datos del usuario como pueden ser el correo y la contraseña.

...

4.2. Diseño de interfaz de usuario

4.3. Arquitectura de sistema

4.3.1. Diagrama de clases

5. Implementación