Escuela
Politécnica
Superior

Aplicación Mascotas

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Jose Francisco Vera Girona
Tutora:
Estela Saquete Boró

Universitat d'Alacant
Universidad de Alicante

Junio 2021

Aplicación Mascotas

Diseño, desarrollo y lanzamiento de una aplicación sobre mascotas

Autor
Jose Francisco Vera Girona

Tutora
Estela Saquete Boró
Departamento de Lenguajes y Sistemas Informáticos

Grado en Ingeniería Informática

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Junio 2021

Preámbulo

Este trabajo partió de un concepto diferente para llegar a lo que es hoy. Si bien concibe aún la estrecha relación con el mundo de las mascotas como en su origen, ha pasado por un proceso evolutivo mediante multitud de sesiones para adquirir un enfoque mucho más desenfadado, sin ceder en utilidad, planteando la posibilidad de obtener así un público objetivo más amplio.

Agradecimientos

Este trabajo ha sido posible gracias a la confianza y al apoyo de mi tutora, Estela Saquete Boró, que me dió la libertad suficiente para convertir su desarrollo en un acto pasional y enriquecedor. También quiero agradecer el apoyo de mis padres, Jose Francisco Vera Ros y Aure Girona Díez, y a la que un día como hoy es mi pareja, Elisabet Rosique Campa, por animarme a seguir cuando he querido abandonar y por soportarme en las noches de frustración cuando había tantos errores como estrellas en el firmamento.

Agradecer también a mis amigos, que tenían la certeza de que aquel joven *yo* que ingresaba a la carrera en 2017 acabaría convirtiéndose en un profesional, y a mis compañeros en PCCOM INNOVATION IS THE WAY S.L., en especial a Sergio Navarro González y a Pablo Sánchez Porras, por mover cielo y tierra ofreciéndome conocimiento, recursos y apoyo siempre que ha sido posible.

Mención especial a mi antiguo profesor de informática, Javier Martín Juan, que sin darse cuenta recondujo la vida de sus alumnos y nos llevó a los míos y a unos cuantos más por la senda de la Ingeniería Informática.

A mi pequeño pomerania, Reina, que me ha hecho amar a los animales durante una vida tan plena que ya casi se le escapa

El optimismo es un riesgo laboral
de la programación;
el feedback es su tratamiento

Kent Beck.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

Este informe contiene una exhaustiva descripción del diseño, desarrollo y lanzamiento de una aplicación móvil sobre mascotas.

1.1 Resumen

La aplicación realizada es una aplicación Android desarrollada en Flutter que pretende satisfacer algunas de las necesidades de los dueños de una o más mascotas, además de suponer una alternativa ociosa a modo de red social en la que compartir material estrictamente relacionado con estas. Aporta, pues, una única solución para la realización de consultas a la comunidad, para la puesta en común sobre conocimiento de locales que aceptan mascotas, para la localización de algunos de los veterinarios cercanos y para la escritura compartida de un histórico de los eventos de la mascota a modo de fragmentos de texto cortos, que pueden ir acompañados de una imagen y que pertenecen a la sección de 'General', si consisten en eventos ociosos y despreocupados o a la de 'Consultas', si consisten en eventos que mantienen preocupados a sus dueños y requieren de atención urgente.

1.2 Motivación y estudio de mercado

En el momento de la redacción de este informe no había ninguna aplicación que mantuviese una alta popularidad con estas características. He consultado a diferentes perfiles de dueño de mascotas para conocer las necesidades anteriormente mencionadas y cómo las satisfacen hoy en día. La respuesta más frecuente ha sido: 'Facebook'.

Facebook mantiene un sistema de comunidades donde compartir información relevante sobre algún tema en particular. Entre estas comunidades encontramos varias que recogen locales que aceptan mascotas, veterinarios y publicaciones sobre mascotas. Aunque sea una correcta ejecución de las herramientas de la plataforma, no es el enfoque real y origen de la misma y, por lo tanto, parece no cubrir totalmente esta necesidad. La popularidad de Facebook tampoco garantiza que estas comunidades tengan la misma popularidad, lo que nos lleva a la segunda respuesta más frecuente: no usar ninguna herramienta y recurrir a la experiencia de sus cercanos.

Si buscamos aplicaciones sobre mascotas en la Google Play Store encontraremos aplicaciones sobre entrenamiento de mascotas, juegos sobre mascotas o algunas otras utilidades como silbatos en los primeros resultados de las búsquedas. No hay mucho campo cubierto y, dada esta realidad, no sorprende a nadie la ausencia de etiquetas como 'Mascotas', 'Perros' o similares al intentar publicar una aplicación en la plataforma. Debido a eso, la aplicación ha sido etiquetada como una red social genérica: 'Blog', 'Entretenimiento', 'Estilo de vida',

'Medicina' y 'Social'.

Llega a mí este concepto siendo yo mismo seguidor de perfiles de mascotas en la red social Instagram. Estos perfiles llegan a acumular millones de seguidores, siendo Nala Cat el perfil con el Récord Guinness Mundial de mayor cantidad de seguidores en Instagram de un gato: 4,3 millones de seguidores[1]. Esta situación sí se podría considerar un incorrecto uso de dicha red social a menos que esta gata tenga en su posesión un correo electrónico o un número de teléfono, pues son datos requeridos para registrarse en la plataforma en cuestión.

1.3 Objetivo

El objetivo consiste en construir una solución que no solo acoja estas situaciones y necesidades, sino que lo haga de forma específica. El usuario debe poder publicar desde los perfiles de sus respectivas mascotas sin necesidad de proporcionarles un número de teléfono y recoger toda la información que le sea valiosa de este ámbito en una aplicación cuyo propósito sea fácil de identificar. El grupo de potenciales usuarios de esta aplicación engloba a todos aquellos que posean una o más mascotas de entre las disponibles, siendo en la versión inicial solo perros y gatos y siendo el número de tipos de mascotas disponibles sujeto a cambios en futuras actualizaciones, que dispongan de un smartphone con Android de sistema operativo.

Además, un valor añadido de este trabajo ha sido incorporar características de accesibilidad, puesto que se trata de una tarea pendiente en numerosas ocasiones. En concreto, se ha utilizado OpenDyslexic, que es una tipografía apta para los usuarios que padecen dislexia, además de un tamaño de letra elevado para que sea más accesible para los usuarios con presbicia. También se ha implementado un clasificador que ayudará a filtrar contenido inapropiado, así que la aplicación es apta para un usuario mayor de 13 años.

[1] Ostenta el récord desde el 3 de mayo de 2017, que se le fue otorgado con 3,4 millones de seguidores.

2 Recursos

Esta sección consta de todo lo referente a medios, utilidades, lenguajes de programación y demás recursos utilizados para la realización de la aplicación.

2.1 Herramientas

He utilizado las siguientes herramientas para desarrollar la aplicación:

- El IDE Visual Studio Code.
- Android Studio con Android Virtual Devices para emular Android y probar la aplicación en un Smartphone, una Tablet de 7 pulgadas y una Tablet de 10 pulgadas y con Android Resource Manager para realizar los gráficos de la aplicación para el sistema.
- Android Debug Bridge para probar la aplicación en un dispositivo real.

2.2 Tecnologías y lenguaje de programación

El lenguaje de programación utilizado ha sido Dart, utilizando el marco de desarrollo Flutter. Dart es un lenguaje de programación de código abierto desarrollado y mantenido por Google que lleva desde 2011 ofreciendo una alternativa a JavaScript, y su popularidad está creciendo gracias al lanzamiento de Flutter 2.0 en marzo de 2021. Flutter pretende facilitar el desarrollo de aplicaciones móviles, web y de escritorio con el uso de Dart y, en mi experiencia personal, es el que más cómodo encuentro para obtener fácilmente un producto mínimo viable, además de que su desarrollo con Visual Studio Code está correctamente integrado gracias al plugin oficial.

Otra de sus virtudes es que el uso de Firebase con Flutter está adaptado mediante un conjunto de librerías llamadas FlutterFire que facilitan mucho su uso, lo que me ha llevado a utilizar Firebase para la autenticación de usuarios (Firebase Authentication), el almacenamiento de imágenes subidas por el usuario (Firebase Cloud Storage) y la base de datos (Firebase Realtime Database).

2.3 Otros recursos utilizados

He utilizado MegaCreator de Icons8 para las ilustraciones de la aplicación, Google Domains para adquirir los dominios `www.firecoffeestudios.com` y `www.guidewoof.com`, Google Websites para hacer la página que está colgada en `www.firecoffeestudios.com` y las siguientes librerías de Flutter:

- **Firebase_core** (Licencia BSD), que es necesario para utilizar cualquiera de las relacionadas con Firebase.
- **Firebase_auth** (Licencia BSD), que es necesario para utilizar Firebase Authentication.
- **Firebase_database** (Licencia BSD), que es necesario para utilizar Firebase Realtime Database.
- **Firebase_storage** (Licencia BSD), que es necesario para utilizar Firebase Cloud Storage.
- **Firebase_messaging** (Licencia BSD), que es necesario para Firebase Messaging.
- **Tflite** (Licencia MIT), que es necesario para utilizar TensorFlow Lite y se ha utilizado para utilizar un clasificador que ayude a filtrar contenido inapropiado en la aplicación.
- **Animations** (Licencia BSD), que es necesario para algunas de las animaciones de la aplicación (navegación por cajas).
- **Flutter_staggered_animations** (Licencia MIT), que es necesario para algunas de las animaciones por fases en listas.
- **Flutter_map** (Licencia BSD) y **Latlong** (Licencia Apache 2.0), que permiten utilizar un mapa y establecer marcadores en él.
- **Geolocator** (Licencia MIT), que permite obtener la ubicación del usuario.
- **Url_launcher** (Licencia BSD), que permite abrir una URL desde la aplicación.
- **Image_picker** (Licencia Apache 2.0), que permite abrir la galería para seleccionar una imagen.
- **Intl** (Licencia BSD), para traducir algunos de los componentes al español.
- **Flutter_local_notifications** (Licencia BSD), para lanzar la notificación al recibir un 'me gusta' o un comentario.
- **Diacritic** (Licencia BSD), para normalizar todos los caracteres de una frase mediante un mapeo de correspondencias.
- **Multi_select_flutter** (Licencia BSD), para crear campos de selección múltiple.
- **Introduction_screen** (Licencia MIT), para crear la pantalla de introducción a la aplicación.

Para la pantalla de 'Veterinarios' se han utilizado los mapas de OpenStreetMap bajo su licencia ODbL, y las tipografías utilizadas son Westhouse (100% libre) para los títulos y OpenDyslexic 2 (licencia SIL-OFL) para el resto de los textos.

3 Requisitos

En esta sección se definen los requisitos de nuestra aplicación, tanto funcionales como no funcionales, que se deben satisfacer para considerar completado su desarrollo.

3.1 Requisitos funcionales

Los requisitos funcionales de la aplicación quedan recogidos en la siguiente tabla de forma jerárquica, de modo que un requisito funcional padre queda satisfecho en el instante en que todos los requisitos funcionales hijos quedan satisfechos.

RFAUT	Autenticación	
RFAUT-1	Registro	El usuario debe poder registrarse y tener constancia de que lo ha hecho.
RFAUT-2	Inicio de sesión	El usuario debe poder iniciar sesión.
RFAUT-3	Gestión de errores de registro e inicio de sesión	El usuario debe conocer por qué no puede registrarse o iniciar sesión.
RFAUT-4	Cerrar sesión	El usuario debe poder registrarse y tener constancia de que lo ha hecho.
RFPET	Mascotas	
RFPET-1	Crear mascota	El usuario debe poder crear una mascota de tipo perro o gato con nombre, descripción, imagen, fecha de nacimiento y raza.
RFPET-2	Vista de mascota simple	La mascota se mostrará visualmente como un elemento que no cubra pantalla completa y con información reducida en las pantallas que lo requieran.
RFPET-3	Vista de mascota detallada	La mascota se mostrará visualmente a pantalla completa con toda su información al interactuar con su vista simple.
RFPET-4	Lista de mascotas	Las mascotas se listarán en, al menos, dos pantallas distintas según si la mascota pertenece o no al usuario que ha iniciado sesión.

RFPET-5	Borrar mascota	El usuario debe poder borrar una mascota creada por él mismo desde su vista detallada.
RFPOST	Publicaciones	
RFPOST-1	Crear publicación	El usuario con mascota debe poder crear una publicación en su nombre de tipo consulta o general con contenido, fecha de publicación e imagen de ser necesaria.
RFPOST-2	Producir error 'Sin mascotas' al intentar publicar	El usuario sin mascotas debe obtener un diálogo de error al intentar crear una publicación advirtiéndole de que es una condición necesaria para proceder.
RFPOST-3	Vista de publicación simple	La publicación se mostrará visualmente como un elemento que no cubra pantalla completa que muestre toda su información sin listar sus comentarios, siendo campos calculados su cantidad de 'Me gusta' y su cantidad de comentarios.
RFPOST-4	Vista de publicación detallada	La publicación se mostrará visualmente como una vista de publicación simple con un botón para comentar y su lista de comentarios cubriendo pantalla completa.
RFPOST-5	Dar 'me gusta'	El usuario podrá dar 'me gusta' a una publicación desde cualquiera de las pantallas en las que sea visible y se podrá saber que se ha hecho mediante un indicador visual.
RFPOST-6	Recibir notificación con la aplicación activa	El usuario con publicación recibirá una notificación con la aplicación activa si esta recibe un comentario o un 'me gusta'.
RFPOST-7	Comentar publicación	El usuario con mascota podrá comentar una publicación desde su vista detallada en su nombre.
RFPOST-8	Producir error 'Sin mascotas' al intentar comentar	El usuario sin mascotas debe obtener un diálogo de error al intentar comentar una publicación advirtiéndole de que es una condición necesaria para proceder.

RFPOST-9	Lista de publicaciones	Las publicaciones se mostrarán en, al menos, una pantalla principal con dos secciones (general y consultas) y, de forma específica, desde una pantalla accesible a través de un botón de la vista detallada de mascotas que te muestre solo las realizadas en su nombre.
RFPOST-10	Borrar publicación	El usuario con publicación debe poder borrar la publicación desde cualquiera de las pantallas en las que sea visible y, además, estas deben ser borradas automáticamente en caso de borrar la mascota cuyo nombre es utilizado para su publicación.
RFVET	Veterinarios	
RFVET-1	Mapa con ubicación	El usuario debe poder consultar el mapa y conocer su ubicación.
RFVET-2	Marcadores de veterinarios	Los veterinarios disponibles se mostrarán visualmente como una medalla en el mapa con su nombre menos las palabras relacionadas con 'Veterinario' o 'Clínica'.
RFVET-3	Realizar búsqueda de veterinario	El usuario debe poder realizar una búsqueda de un veterinario interaccionando con su marcador en el mapa.
RFLOCAL	Locales dog-friendly	
RFLOCAL-1	Registrar locales dog-friendly	El usuario debe poder registrar un local dog-friendly con nombre, tipo, provincia, si acepta perros grandes o no, si acepta razas de perros potencialmente peligrosas o no, descripción e imagen.
RFLOCAL-2	Vista de local simple	El local se mostrará visualmente como un elemento que no cubra pantalla completa y con información reducida en las pantallas que lo requieran.
RFLOCAL-3	Vista de local detallada	El local se mostrará visualmente a pantalla completa con toda su información al interactuar con su vista simple.
RFLOCAL-4	Lista de locales	Los locales se mostrarán en, al menos, una pantalla general.
RFLOCAL-5	Borrar locales	El usuario debe poder borrar un local creado por él mismo desde su vista detallada.

RFACC	Información de cuenta	
RFACC-1	Seguir mascotas	El usuario debe poder seguir o dejar de seguir mascotas desde su vista detallada.
RFACC-2	Mostrar publicaciones de mascotas seguidas	Las publicaciones que aparecen en la página principal de la sección de general serán únicamente de las mascotas que sigues o de las que te pertenecen.
RFACC-3	Seleccionar y deseleccionar razas favoritas	El usuario debe poder seleccionar y deseleccionar razas favoritas de mascotas y estas se mostrarán las primeras en las listas.
RFACC-4	Elegir tipo de animal	El usuario debe poder decidir si quiere ver solo perros, solo gatos o ambos y se mostrarán únicamente los pertenecientes al tipo electo en las listas de mascotas.
RFACC-5	Elegir provincia	El usuario debe poder elegir la provincia en la que vive y se mostrarán primero los locales pertenecientes a esta provincia.
RFACC-6	Cambiar contraseña	El usuario debe poder cambiar su contraseña mediante un correo electrónico.
RFACC-7	Pantalla de ayuda	El usuario debe poder acceder desde el menú principal a una pantalla de ayuda que le explique algunas de las funcionalidades de la aplicación.

Tabla 3.1: Tabla de requisitos funcionales.

3.2 Requisitos no funcionales

Los requisitos no funcionales de la aplicación quedan recogidos en la siguiente tabla de forma jerárquica, de modo que un requisito no funcional padre queda satisfecho en el instante en que todos los requisitos no funcionales hijos quedan satisfechos.

RNFACC	Accesibilidad	
RNFACC-1	Combinación de colores apta para daltónicos	Los colores elegidos para la interfaz de la aplicación no coincidirán con colores que pudieran suponer problemas a usuarios daltónicos.
RNFACC-2	Tipografía apta para disléxicos	La tipografía general de la aplicación será apta para usuarios disléxicos.
RNFACC-3	Tamaño de fuente elevado	Se mantendrá, a lo largo de toda la aplicación, un tamaño de fuente cómodo para usuarios con presbicia.

RNFPRIV	Privacidad	
RNFPRIV-1	Anonimato	Los protagonistas de la aplicación serán únicamente las mascotas, no existiendo perfil de dueño consultable por la comunidad y no pudiéndose distinguir si dos mascotas pertenecen al mismo dueño o no; tampoco habrá información relativa a seguidores.
RNFCOMPAT	Compatibilidad	
RNFCOMPAT-1	Compatible con los dispositivos sugeridos	La aplicación será totalmente compatible y estará correctamente adaptada a los tipos de dispositivos sugeridos por Google Play.[1]
RNFREND	Rendimiento	
RNFREND-1	Responde adecuadamente	La aplicación responde a las acciones del usuario en un tiempo menor a 2 segundos, incluso en situaciones de carga de datos en condiciones óptimas.
RNFCAP	Capacidad	
RNFCAP-1	Almacenamiento escalable	Los servidores de almacenamiento utilizados deberán ser escalables por si la situación lo requiriese.
RNFCAP-2	Compresión de imágenes	Las imágenes se subirán comprimidas y redimensionadas para reducir la ocupación del servidor de almacenamiento sin que el usuario perciba una reducción de calidad de las mismas.
RNFGAR	Garantías	
RNFGAR-1	Filtrado de contenido inapropiado	Se garantizará que el contenido que aparece en la aplicación es apto para mayores de 13 años mediante la inclusión de alguna herramienta de filtrado automático o una opción de reportar.

Tabla 3.2: Tabla de requisitos no funcionales.

[1] Google Play requiere capturas de pantalla de smartphones, tablets de 7 pulgadas y tablets de 10 pulgadas.

4 Planificación

Esta sección detalla la planificación del proyecto, su estimación temporal y el tiempo dedicado a su desarrollo, así como su gestión.

4.1 Estimación temporal

La estimación temporal se ha realizado mediante dos diagramas de Gantt en dos situaciones diferentes. El primer diagrama corresponde al primer semestre y solo comprende los requisitos de autenticación y veterinarios. El segundo diagrama corresponde al segundo semestre y comprende el resto de requisitos funcionales y cuestiones de diseño.

Cabe mencionar que estos diagramas únicamente pretenden recoger requisitos funcionales y cuestiones de diseño, y no comprenden otro tipo de tareas como las organizativas, la página web o la configuración de la ficha de Google Play.

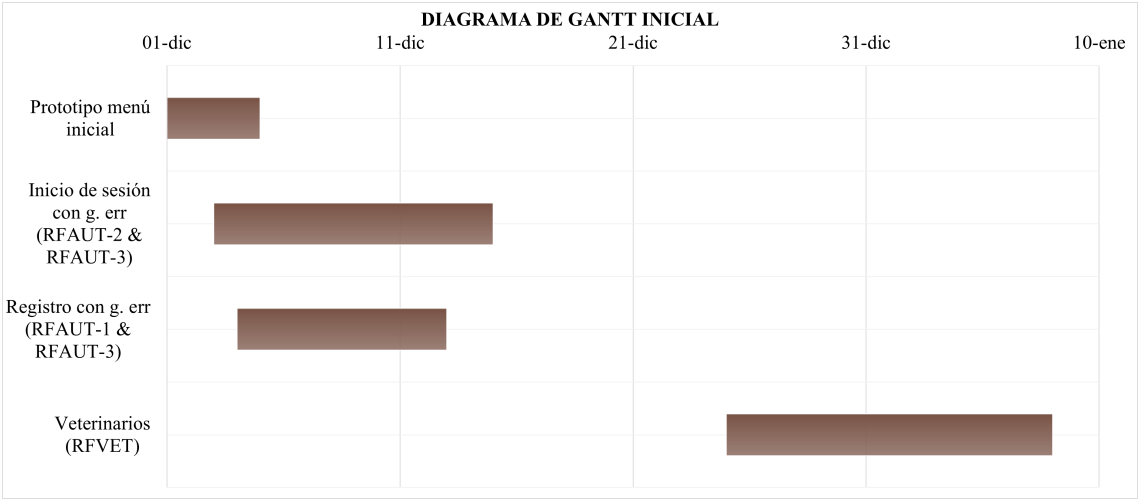


Figura 4.1: Diagrama de Gantt inicial (diciembre)

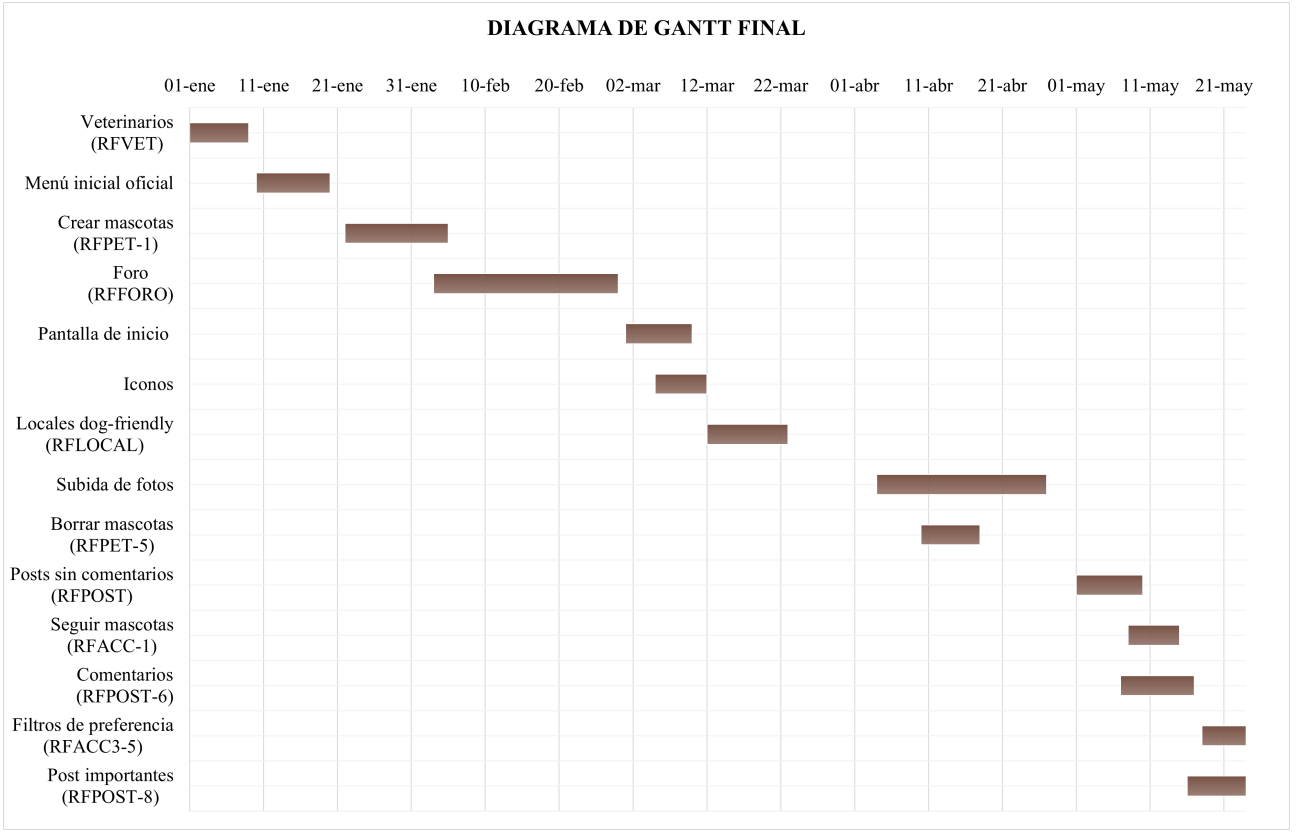


Figura 4.2: Diagrama de Gantt final (enero-mayo)

[1]RFFORO es un requisito funcional que ha sido descartado en posteriores iteraciones.
[2]Subida de fotos corresponde a mover las fotos insertadas de la creación en local al servidor de almacenamiento

Finalmente, podemos obtener el trabajo realizado mediante las gráficas de GitHub de *commits* realizados y de adiciones y eliminaciones por semana.

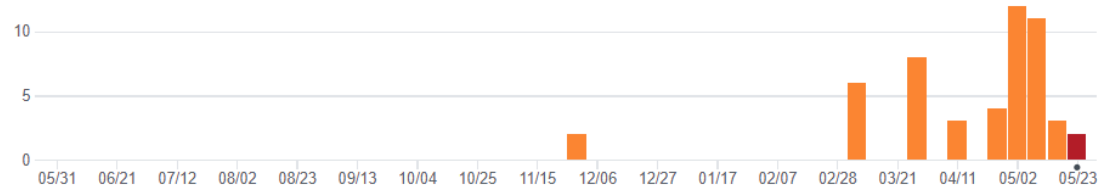


Figura 4.3: Gráfica de barras de *commits* realizados

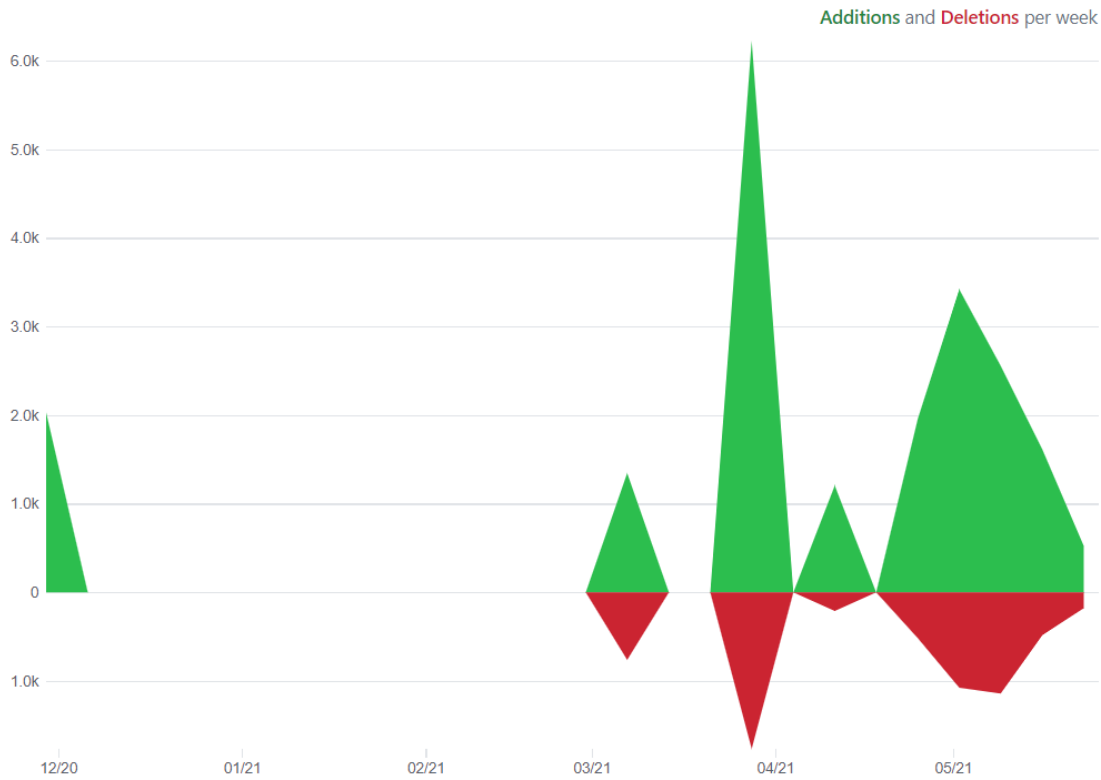


Figura 4.4: Polígono de frecuencia de código por semanas

5 Diseño

Esta sección profundiza en el diseño de la aplicación, tanto en la inspiración y toma de decisiones inicial como en los bocetos y su resultado.

5.1 Inspiración y toma de decisiones

En la fase inicial de la aplicación prototipé el menú principal. Para ello, tomé de inspiración la interfaz usada por Microsoft en Windows 8: Modern UI.

Figura 5.1: Aplicaciones de Microsoft con Modern UI

Sin embargo, renuncié a las esquinas en punta y opté por un esquema de colores basado en Material Design que recordase a las mascotas. Tras varias iteraciones, acabé con un esquema de colores apto para daltónicos, como se puede ver en la figura 5.2 extraída de https://davidmathlogic.com.

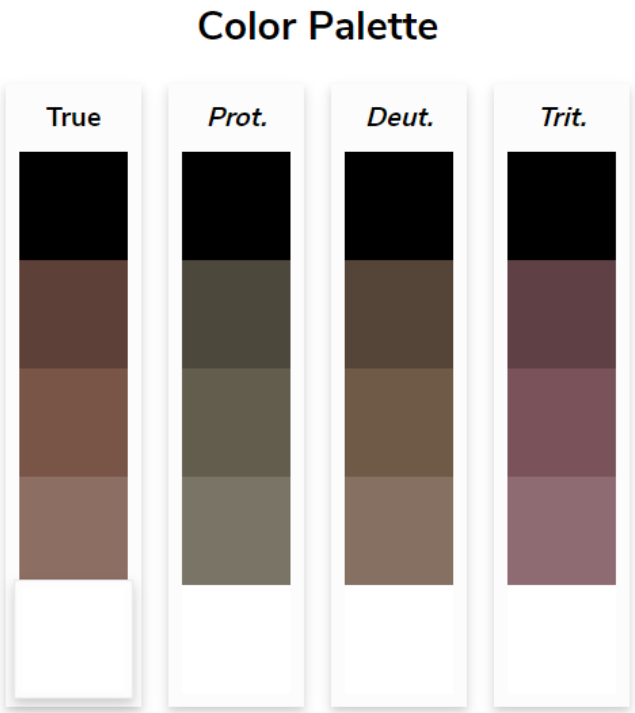


Figura 5.2: Paleta de colores utilizada y cómo se vería con protanopía, deuteranopía y tritanopía

Decidí que la aplicación se llamaría 'Guidewoof' y que la tipografía para el logo y las cabeceras sería Westhouse.

Figura 5.3: Logo de la aplicación con color principal de fondo

También escogí cuál sería la tipografía para el resto de la aplicación: OpenDyslexic 2. Con esto decidido, inicié los bocetos.

Figura 5.4: Fragmento de la primera de las pantallas de ayuda

5.2 Bocetos

Los bocetos de la aplicación se realizaron en papel para poder trabajar en cualquier momento que se me ocurriese un nuevo diseño. Insertaré un par de ejemplos, pero no toda la aplicación parte de estos bocetos ya que Flutter facilita mucho construir el diseño de sus *widgets*, así que me he visto en más de una ocasión en la que tuve el ordenador delante programando el diseño y realizando los ajustes necesarios.

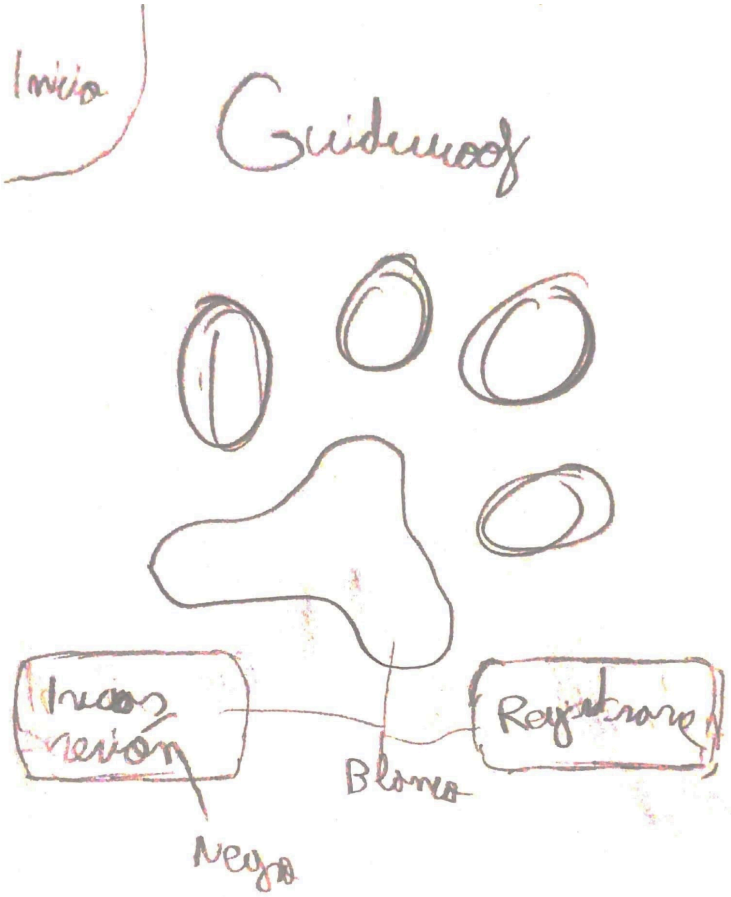


Figura 5.5: Página del boceto de la pantalla de inicio

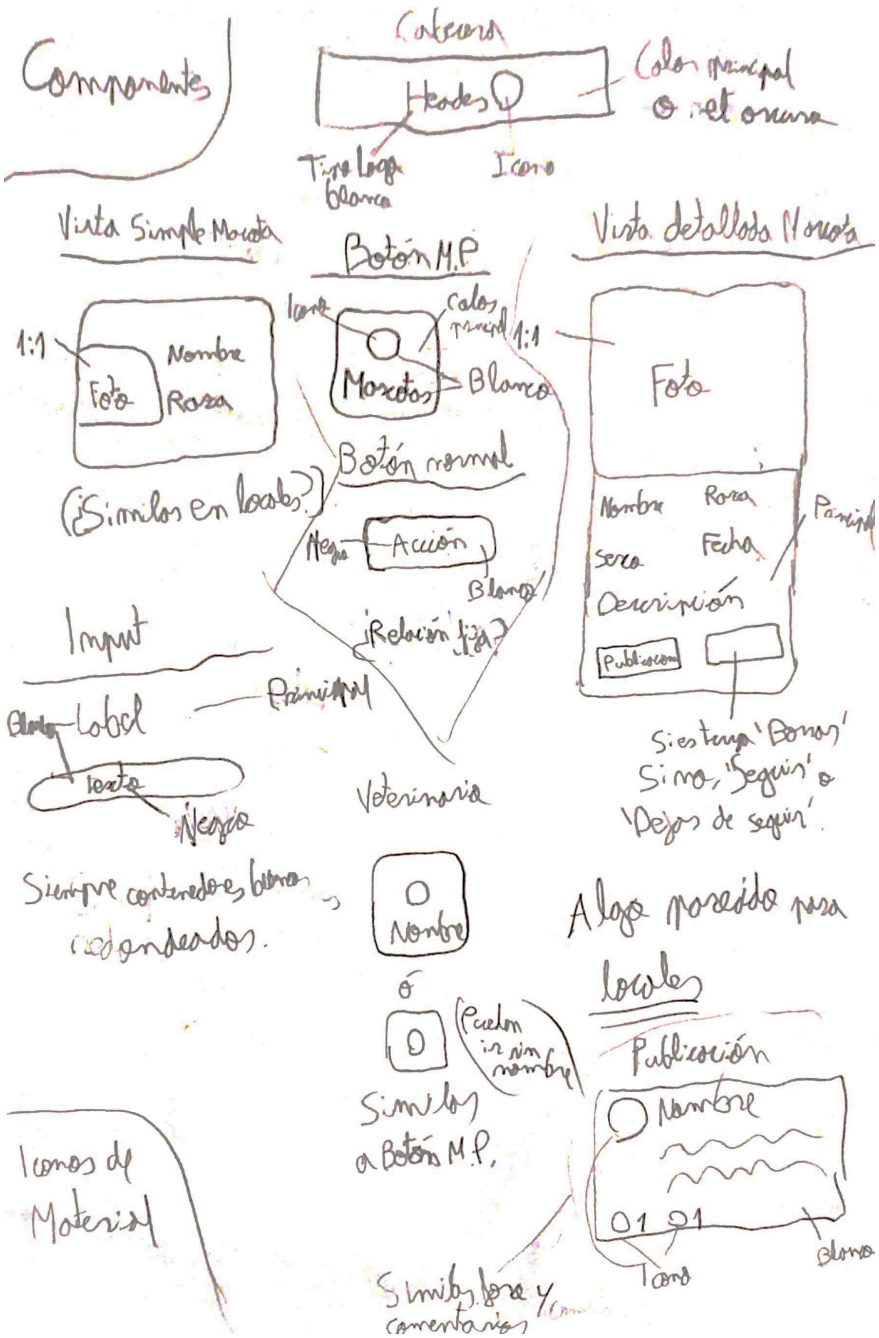


Figura 5.6: Página de bocetos de componentes

1

[1]Si bien podría haber optado por fingir haber hecho otro tipo de bocetos u omitir esta sección en particular, he optado por desnudar mi alma y mi disgrafía reflejando la más cruda realidad de este desarrollo.

5.3 Diseño final

En esta sección recopilaré el resultado final del diseño, es decir, tal y como se encuentra actualmente en la aplicación.

(a) Pantalla de inicio.

(b) Pantalla de inicio de sesión.

(c) Pantalla de registro.

(d) Pantalla del menú principal.

Figura 5.7: Pantallas de la aplicación

(e) Primera pantalla de ayuda.

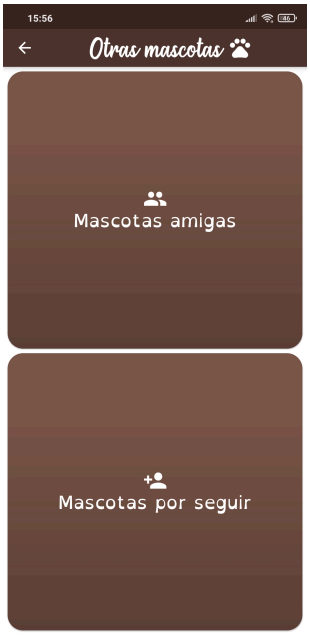
(f) Segunda pantalla de ayuda.

(g) Tercera pantalla de ayuda.

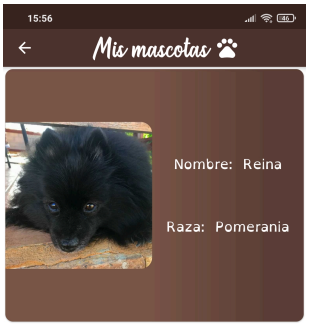
(h) Cuarta pantalla de ayuda.

Más pantallas de la aplicación

(i) Quinta pantalla de ayuda.



(j) Pantalla del menú de mascotas.

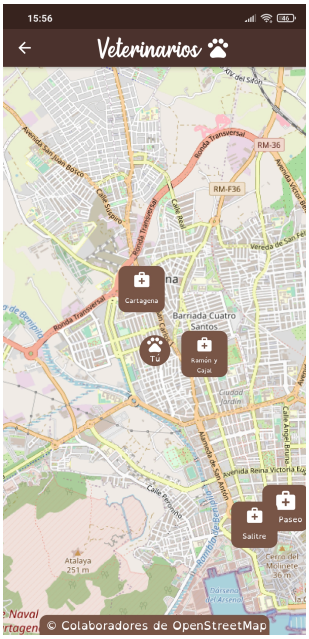


(k) Pantalla de mis mascotas.

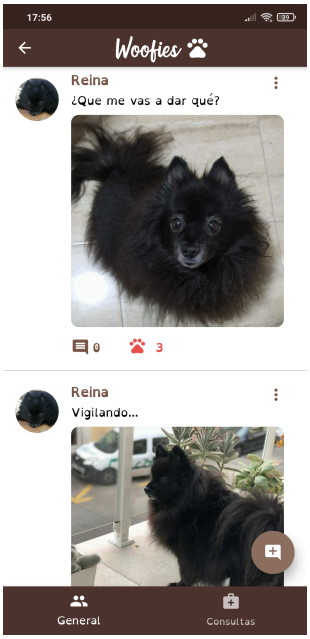
(l) Pantalla de vista detallada de mascota.

Más pantallas de la aplicación

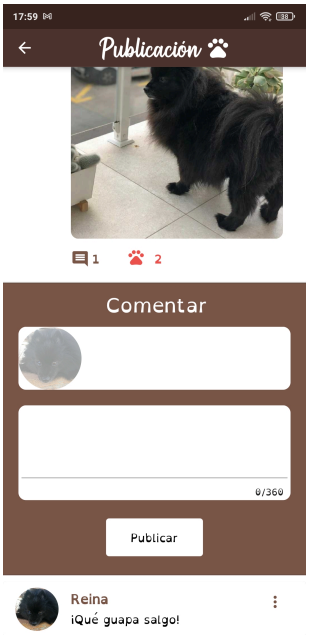
(m) Pantalla de crear mascota.



(n) Pantalla de veterinarios con mapas de OpenStreetMap.

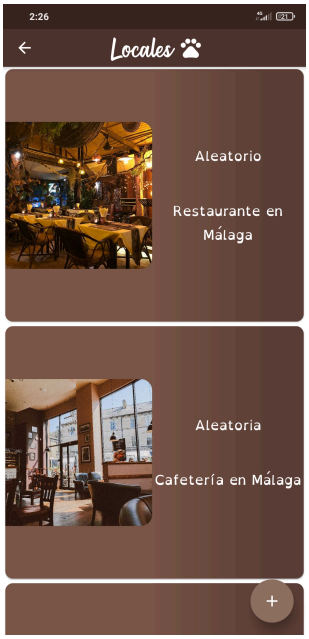


(o) Pantalla principal de publicaciones.



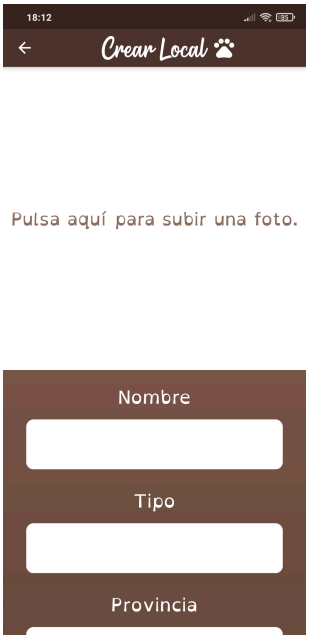
(p) Pantalla de vista detallada de publicación.

Más pantallas de la aplicación



(q) Pantalla de locales dog-friendly.

(r) Pantalla de vista detallada de local.



(s) Pantalla de crear local.

(t) Pantalla de resetear contraseña.

Más pantallas de la aplicación

6 Arquitectura

En esta sección se detalla la arquitectura utilizada para la aplicación y algunos detalles sobre organización y buenas prácticas.

6.1 Modelo-vista-controlador

La aplicación utiliza una arquitectura de modelo-vista-controlador. El modelo constituye la capa de acceso a datos: ahí se mantienen las instancias de Firebase y las estructuras de datos. El modelo presenta los datos a la vista, que les proporciona un aspecto gráfico y permite interactuar con ellos manejando dicha interacción a través del controlador, que es el encargado de manipular el modelo.

A raíz de esto, la organización de las carpetas ha resultado como en la figura 6.1, siendo *constants* una carpeta que contiene valores reusables y finales a lo largo de toda la aplicación y estando la carpeta *view* dividida en dos subcarpetas: la referente a *widgets*, *components*, y la referente a las pantallas, *screens*. Además, se ha seguido la guía de estilo oficial de Effective Dart para escribir código robusto, consistente y conciso.

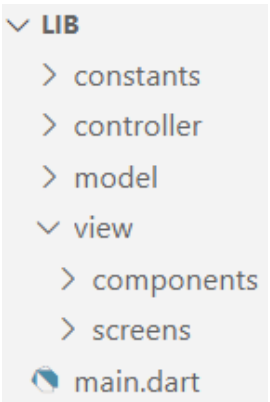


Figura 6.1: Jerarquía de carpetas

Aunque se podría considerar que hay otras arquitecturas más apropiadas para el desarrollo en Flutter como pueden ser BLoC o Redux, he optado por una arquitectura modelo-vista-controlador porque es la que hemos aprendido y madurado durante la carrera.

Tampoco se han utilizado librerías para ayudar en su implementación pese a haber varias que facilitan la tarea.

7 Implementación

En esta sección se detalla la implementación de la aplicación, tanto *front-end* como *back-end*, así como la base de datos utilizada y su estructura.

7.1 Front-end

El *front-end* está implementado con Flutter, utilizando los *widgets* de Material Design. Para empezar un proyecto como este tenemos que descargarnos Flutter SDK, Android Studio y algún IDE. Visual Studio Code tiene una buena integración con el plugin de Flutter y Dart proporcionando un analizador de código estático, un formateador de código, herramientas para depurar y la posibilidad de trabajar con *hot reload*[1].

```
flutter create guidewoof
```

Figura 7.1: Comando para crear un nuevo proyecto

El directorio que genera el comando de la figura 7.1 se compone de las siguientes carpetas:

- **android**, donde encontraremos los archivos necesarios para compilar a Android, tales como buid.gradle, AndroidManifest.xml...
- **ios**, carpeta similar pero en este caso con iOS, que permitiría compilar el proyecto en XCode.
- **lib**, carpeta donde se encontraría todo nuestro código y que parte con un archivo main.dart.

Una aplicación en Flutter con Material debe tener en su raíz un *widget* de tipo MaterialApp para funcionar. MaterialApp contendrá un Scaffold, que puede tener barra de aplicación y un *widget* hijo. El desarrollador puede agrupar múltiples *widgets* en uno solo creado por él, para facilitar reusar elementos con características concretas o un grupo de elementos a la vez. Así, un posible diagrama resultante de esta estructura podría ser el de la figura 7.2.

[1] *Hot reload* refiere a la capacidad de un *framework* de reflejar los cambios en el código sin necesidad de realizar un compilado completo.

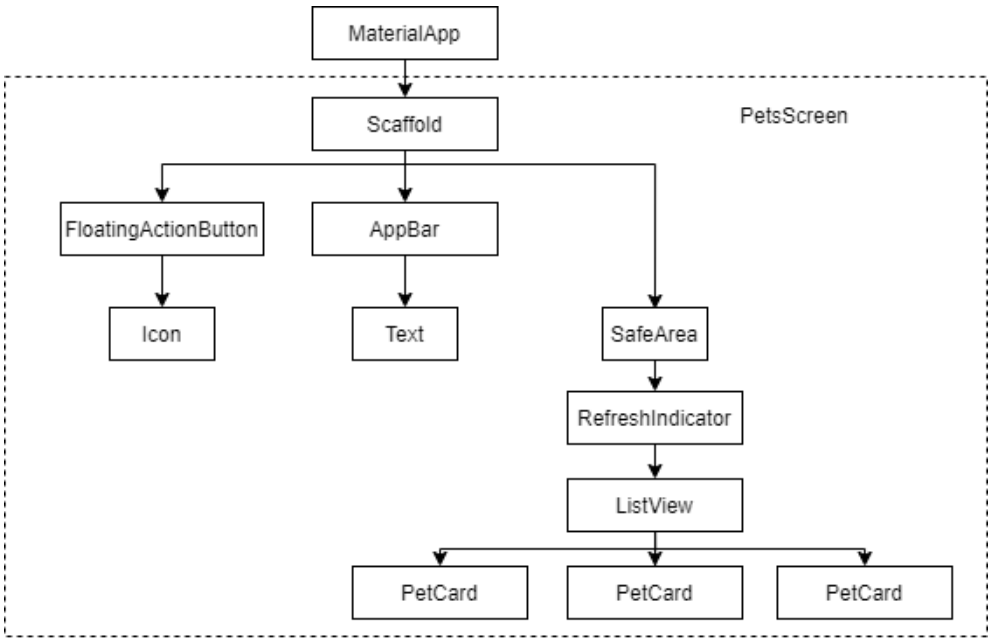


Figura 7.2: Diagrama que representa la estructura de la aplicación en la pantalla de 'Mascotas por seguir'

Hay que tener en cuenta que esto es una simplificación, muestra de como PetsScreen contiene, a su vez, todo estos *widgets* y que, de ir al más bajo nivel, el *widget* PetCard también se descompondría en más *widgets* hijos por ser un componente realizado por mí para representar la vista simple de las mascotas.

7.2 Back-end

El *back-end* de la aplicación se basa en Firebase. Firebase nos proporciona múltiples herramientas que podemos usar de forma gratuita con su plan Spark, de las cuales tres han sido los órganos vitales de la aplicación:

- **Firebase Authentication** nos ofrece un sistema seguro y gratuito de autenticación de usuarios por diferentes medios, entre los cuales he optado por implementar la autenticación clásica de combinación correo-contraseña.

- **Firebase Cloud Storage** nos ofrece un servidor de almacenamiento gratuito, siempre que no se alcance la cantidad de GB establecidos tanto en el almacenamiento como en las descargas diarias, y ha sido la opción ideal para implementar las imágenes de las publicaciones, de las mascotas y de los locales.

- **Firebase Realtime Database** nos ofrece una base de datos NoSQL adecuada para reaccionar a cambios en tiempo real y gratuita, siempre que no se alcance ni la cantidad de GB establecidos tanto en el almacenamiento de datos como en las descargas al mes ni las conexiones simultáneas establecidas, y ha sido la opción ideal para gestionar los

datos de los locales, las mascotas, las publicaciones y la información de cuenta de los usuarios.

Por la particularidad de su implementación, voy a desarrollar un poco más cómo se ha implementado Firebase Realtime Database.

Firebase Realtime Database basa la obtención y manipulación de los datos en local mediante la posibilidad de crear *listeners* sobre los nodos resultantes de una *query*, aunque también permite utilizar **once**, que realiza una instantánea de los datos tal cual aparecen actualmente en la base de datos. Estos *listeners* pueden escuchar los siguientes eventos:

- **onValue**, que se lanza al descargar todos los resultados de una *query* y cada vez que los resultados se actualizan.
- **onChildRemoved**, que se lanza cada vez que se borra un nodo resultante de una *query*.
- **onChildAdded**, que se lanza cada vez que se crea un nodo y la primera vez que se descarga.
- **onChildChanged**, que se lanza cada vez que un nodo se actualiza.

Con estos detalles y para representar su uso mediante un ejemplo, voy a explicar cómo se han controlado los 'me gusta' de las publicaciones para que funcionasen en tiempo real.

Al descargar una publicación mediante el evento de onChildAdded de la query correspondiente, suscribiremos dos *listeners* a la lista de *likes*: uno para onChildAdded y otro para onChildRemoved. La estructura de datos de la publicación en local almacena los 'me gusta' como un número y si posee nuestro 'me gusta' o no como un booleano. Con esto en mente, al dispararse el evento de onChildAdded sumamos una unidad a los 'me gusta' y comprobamos si el 'me gusta' es nuestro y de serlo, actualizamos el booleano a verdadero. De igual manera, al dispararse el evento de onChildRemoved restamos una unidad a los 'me gusta' y comprobamos si el 'me gusta' eliminado es el nuestro y de serlo, actualizamos el booleano a falso.

7.3 Base de datos

La base de datos de Firebase Realtime Database es una base de datos NoSQL. Las bases de datos NoSQL se diferencian de las bases de datos SQL en que las primeras no son bases de datos relacionales. Firebase Realtime Database almacena sus datos como objetos JSON, lo que permite una gran escalabilidad y flexibilidad: se pueden modificar estructuras de datos sin detener el servicio, convirtiéndose así en una opción más que recomendable en un proyecto en constante evolución como este. El diagrama de la figura 7.3 representa cómo se mantienen los datos en la base de datos.

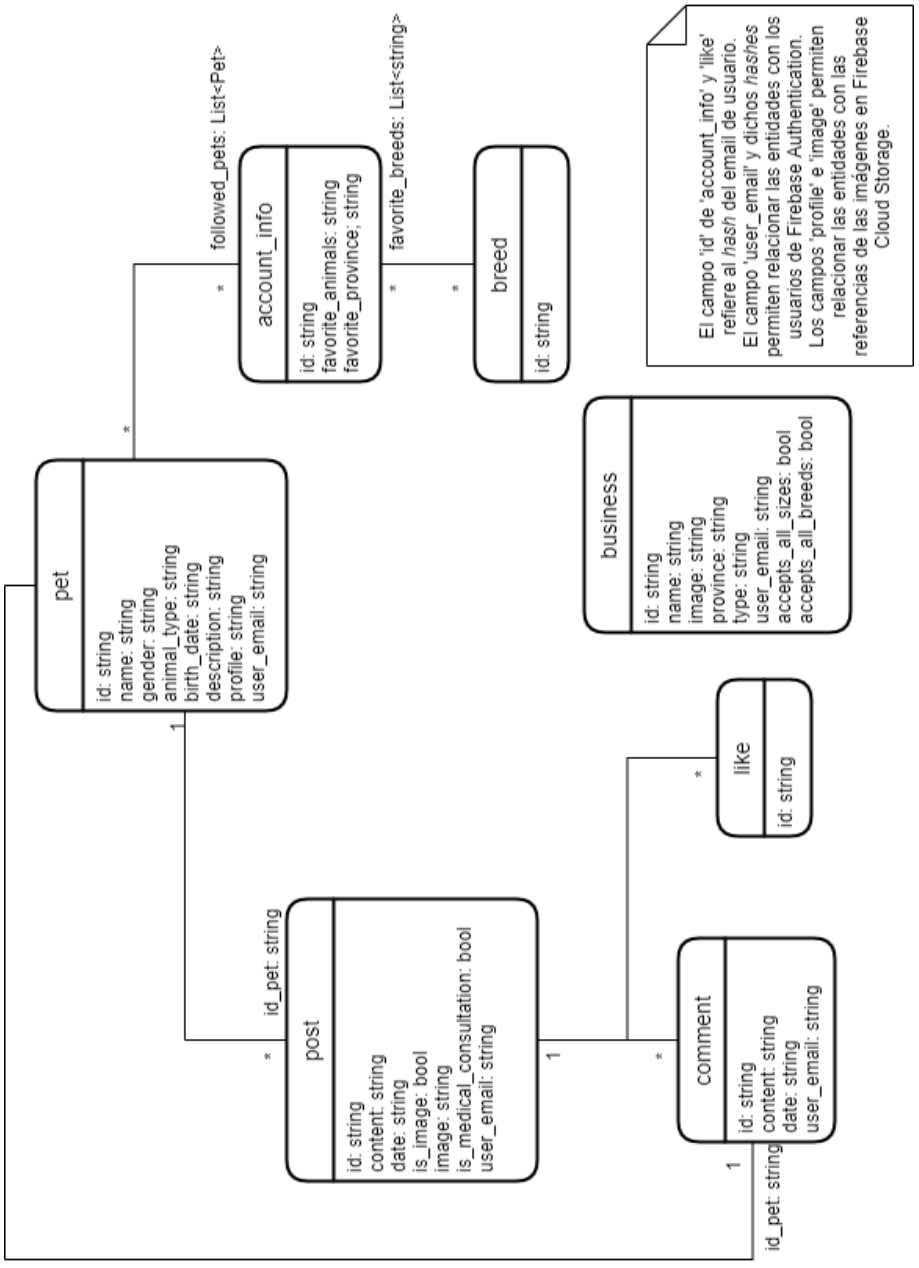


Figura 7.3: Diagrama de la base de datos

8 APIs utilizadas

En este breve capítulo se detalla cómo se ha implementado la API de OpenStreetMap para satisfacer los requisitos funcionales referentes a veterinarios.

8.1 OpenStreetMap

La librería utilizada para gestionar mapas en la aplicación, **flutter_map**, permite utilizar APIs para descargar las imágenes necesarias de un *tile server*. OpenStreetMap mantiene uno de estos *tile server* de su versión oficial mediante donaciones con una URL con esta estructura:

https://'s'.tile.openstreetmap.org/'z'/'x'/'y'.png

Donde 's' debe ser sustituido por 'a', 'b' o 'c' y corresponde a los posibles subdominios, 'z' debe ser sustituido por un número y corresponde al nivel de zoom que posee el mapa y 'x' e 'y' deben ser sustituidos por números y corresponden al valor X e Y del mapa. Así, el enlace `https://a.tile.openstreetmap.org/7/63/49.png` genera la figura 8.1.

Figura 8.1: Tile de OpenStreetMap centrado en Cartagena, Murcia

Teniendo esto en cuenta, la librería flutter_map se encarga de gestionar cómo cambian los valores 'z', 'x' e 'y' según la interacción del usuario.

9 Seguridad

En esta sección se detalla la seguridad implementada en el proyecto.

9.1 Autenticación

Toda la aplicación está protegida a través de un sistema de autenticación con Firebase Authentication: ni Firebase Realtime Database ni Firebase Cloud Storage permiten la descarga de datos si no es a través de un usuario autenticado con Firebase Authentication gracias al uso de reglas.

Las cuentas en Firebase Authentication no permiten al administrador acceder al valor almacenado de la contraseña de las cuentas en cuestión. Todas ellas están almacenadas en *hash* mediante un algoritmo modificado que parte de Scrypt. La figura 9.1 son los parámetros que Firebase Authentication expone de este algoritmo con la finalidad de facilitar migraciones de cuenta a la plataforma.

```
hash_config {
  algorithm: SCRYPT,
  base64_signer_key:
  base64_salt_separator:
  rounds:
  mem_cost:
}
```

Figura 9.1: Los parámetros expuestos por Firebase Authentication

Además, a raíz de los requisitos obtenidos, los únicos datos sensibles del usuario que se almacenan son su nombre, su correo, su provincia (voluntariamente) y la contraseña que proporcione. Al no existir ningún tipo de perfil de dueño, no existen apenas datos sensibles vulnerables y, por lo tanto, es menor la gravedad de un robo de datos que en caso de almacenar datos como DNI, dirección o número de teléfono.

10 Posibles mejoras y conclusiones

Esta sección hace referencia a las posibles mejoras que se pueden realizar a la aplicación y las conclusiones extraídas de su desarrollo.

10.1 Posibles mejoras

Debido a que los requisitos de la aplicación han sido recogidos con una fecha límite estricta de entrega se han dejado fuera algunas funcionalidades que podrían ser muy útiles dentro de la aplicación, por listar algunas de ellas:

- Autenticación de usuarios a través de Google Play.
- Enlazar otras mascotas mediante mención a una publicación.
- Enlazar un local mediante mención a una publicación.
- Listar publicaciones que mencionan a un local.
- Permitir cuentas de veterinario verificadas de algún modo para responder las consultas con carácter profesional.
- Permitir búsquedas más específicas de mascotas.
- Permitir búsquedas más específicas de locales.
- Permitir un campo 'Ciudad' en locales.
- Recibir notificaciones *push* cuando la aplicación está cerrada.
- Permitir cambiar la foto de una mascota.
- Permitir que un dueño de local lo publique con un distintivo tras ser verificado de algún modo.
- Añadir más tipos de mascotas como pájaros, conejos o tortugas.
- Permitir ordenar mascotas por popularidad o frecuencia de publicaciones.
- Una pantalla de notificaciones recibidas para saber a qué publicaciones hacen referencia.
- Dar 'me gusta' a un comentario de una publicación.
- Versión para iOS.[1]

[1] A primeros de mayo de 2021, Flutter aún no había ofrecido una versión de su canal estable que fuese compatible con el chip M1 de los nuevos dispositivos de Apple.

10.2 Conclusiones

Este proyecto ha supuesto el primero de gran embergadura del que he sido único responsable. La necesidad de investigar para plantear soluciones a problemas ante los que nunca me he visto en esta carrera universitaria ni en mi escasa experiencia laboral me ha llevado a aprender en profundidad sobre un *framework* tan potente como el que es Flutter y sobre la alternativa tan económica que ofrece Firebase como solución en la nube. He dejado de manifiesto en numerosas ocasiones cómo este desarrollo había alcanzado una motivación mucho más pasional y, a mi juicio, ha quedado patente en el resultado final de la aplicación.

Esto también ha sido un golpe de realidad descubriéndome, al fin, la diferencia que define a un alumno a la víspera de ser graduado en Ingeniería Informática de un mero programador: la diferencia tantas veces mencionada por el profesorado y tan pocas veces comprendida por sus alumnos queda ahora reflejada entre el joven *yo* que entraba a la carrera hace cuatro años con el que a día de hoy escribe estas páginas.

Con todo esto, 'Guidewoof' se ha convertido en una aplicación de la que estoy orgulloso: un proyecto que se enriquece junto a su comunidad y que seguirá recibiendo mi tiempo y mi apoyo para hacer sonar lo mejor posible esta humilde oda a la humanidad y al amor animal.

La aplicación se encuentra accesible en Google Play desde el día 24 de mayo de 2021.

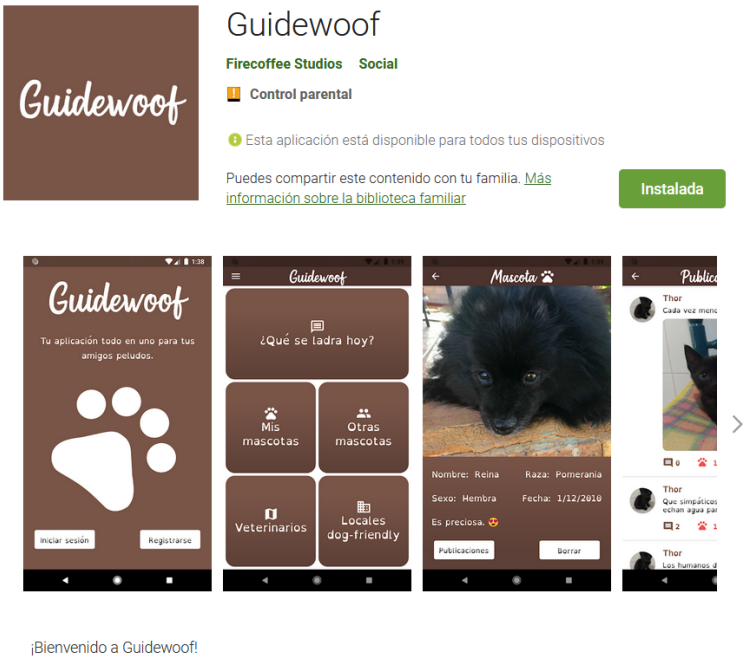


Figura 10.1: Ficha de Google Play de la primera versión en producción de Guidewoof

Bibliografía

agilord.com. (2021). diacritic. En *pub.dev* (.1.3 ed.). Descargado de `https://pub.dev/packages/diacritic`

baseflow.com. (2021). geolocator. En *pub.dev* (7.0.3 ed.). Descargado de `https://pub.dev/packages/geolocator`

ch.botha61@gmail.com. (2021). multi_select_flutter. En *pub.dev* (4.0.0 ed.). Descargado de `https://pub.dev/packages/multi_select_flutter`

dexterx.dev. (2021). flutter_local_notifications. En *pub.dev* (6.0.0 ed.). Descargado de `https://pub.dev/packages/flutter_local_notifications`

firebase.google.com. (2021a). firebase_auth. En *pub.dev* (1.2.0 ed.). Descargado de `https://pub.dev/packages/firebase_auth`

firebase.google.com. (2021b). firebase_core. En *pub.dev* (1.2.0 ed.). Descargado de `https://pub.dev/packages/firebase_core`

firebase.google.com. (2021c). firebase_database. En *pub.dev* (7.0.0 ed.). Descargado de `https://pub.dev/packages/firebase_database`

firebase.google.com. (2021d). firebase_messaging. En *pub.dev* (10.0.0 ed.). Descargado de `https://pub.dev/packages/firebase_messaging`

firebase.google.com. (2021e). firebase_storage. En *pub.dev* (8.1.0 ed.). Descargado de `https://pub.dev/packages/firebase_storage`

flutter.dev. (2021a). animations. En *pub.dev* (2.0.0 ed.). Descargado de `https://pub.dev/packages/animations`

flutter.dev. (2021b). url_launcher. En *pub.dev* (6.0.4 ed.). Descargado de `https://pub.dev/packages/url_launcher`

Gonzalez, A. (2020). *Opendyslexic.* Descargado de `https://opendyslexic.org/`

Google. (2021a). Effective dart [Manual de software informático]. Descargado de `https://dart.dev/guides/language/effective-dart`

Google. (2021b). Firebase documentation [Manual de software informático]. Descargado de `https://firebase.google.com/docs`

Google. (2021c). Flutter documentation [Manual de software informático]. Descargado de `https://flutter.dev/docs`

Google. (2021d). *Google play console.* Descargado de https://play.google.com/apps/publish

jeancharles.msse@gmail.com. (2021). introduction_screen. En *pub.dev* (2.1.0 ed.). Descargado de https://pub.dev/packages/introduction_screen

jpryan.me. (2021). flutter_map. En *pub.dev* (.12.0 ed.). Descargado de https://pub.dev/packages/flutter_map

mobiten.com. (2021). flutter_staggered_animations. En *pub.dev* (1.0.0 ed.). Descargado de https://pub.dev/packages/flutter_staggered_animations

office@mikemitterer.at. (2021). latlong. En *pub.dev* (.6.1 ed.). Descargado de https://pub.dev/packages/latlong

Openstreetmap. (2021). Descargado de https://www.openstreetmap.org/

Prasad, A. (2021, enero). *Flutter gems.* Descargado de https://fluttergems.dev/

Vera, J. F. (2021). *Guidewoof.* Descargado de https://play.google.com/store/apps/details?id=com.firecoffeestudios.guidewoof

Wellness, A. (2019, noviembre). Nala cat holds guinness world record. *Animal Wellness Magazine.* Descargado de https://animalwellnessmagazine.com/nala-cat-guinness-world-record/

x_ash@qq.com. (2021). tflite. En *pub.dev* (1.1.2 ed.). Descargado de https://pub.dev/packages/tflite