



**Universidad
Europea**

LAUREATE INTERNATIONAL UNIVERSITIES

UNIVERSIDAD EUROPEA DE MADRID



ESCUELA ARQUITECTURA INGENIERÍA Y DISEÑO

CICLO FORMATIVO DE GRADO SUPERIOR
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

PROYECTO FIN DE CICLO

PetAware

**Borja Olalla Lagarejo
Carlos Ortiz Menéndez
Sergio Cuadrado Vereau**

CURSO 2017-18

TÍTULO: PetAware

AUTORES: Borja Olalla Lagarejo
Carlos Ortiz Menéndez
Sergio Cuadrado Vereau

TUTOR DEL PROYECTO: Ernesto Ramiro Córdoba.

FECHA DE LECTURA: 13 de Junio de 2018

CALIFICACIÓN:

Fdo: Ernesto Ramiro Córdoba.

Ernesto Ramiro Córdoba
Tutor/a del Proyecto

RESUMEN:

En este documento se recoge el trabajo realizado, el objetivo, y las motivaciones que han hecho posible que el proyecto de PetAware haya sido posible. PetAware es una aplicación de Android destinada a propietarios de mascotas. La aplicación permite registrar las mascotas del usuario, guardando los datos en Firebase; además de crear *eventos* y *anuncios*. Los *eventos* son distintas actividades que los usuarios pueden organizar y son eliminados temporalmente. Por otro lado, los *anuncios* son creados para avisar a los demás usuarios de que se ha perdido una mascota. Para crear anuncio el usuario debe de introducir un teléfono de contacto para que otros usuarios sean capaces de llamarle en caso de que vean la mascota. Los anuncios y los eventos aparecen en el mapa como marcadores.

Lo que se ha querido hacer con esta aplicación ha sido dar a los usuarios que tengan mascotas un servicio mediante el cual puedan compartir noticias en caso de que su mascota se pierda, de forma de que los demás usuarios puedan estar atentos en caso de que esto pase. Sin embargo, también se ha intentado dar a la aplicación un toque de “red social”, permitiendo que los usuarios puedan hacer sus propios eventos para realizar actividades. De esta forma la aplicación cumple dos funciones: agiliza el proceso de búsqueda de mascotas y permite organizar actividades temporales en las que pueden participar los usuarios que lo deseen.

ABSTRACT:

This document contains data of the work process, the goal and motivations that made the PetAware project possible. PetAware is an Android app destined to pet owners. This app can register its users' pets, storing the pet data into Firebase Database; as well as create events and news. Events are different activities users can organize and are eventually deleted. On the other hand, news are created to warn other users about missing pets. To create these, the user must provide a phone number in order to make contact possible with other users who might have seen the lost pet. Both event and news appear in the map as markers.

What this app was meant to do was provide its users a service they could use to share news in case their pets get lost, so other users can be aware if this comes to happen. However, this app was also given a “social network” touch, letting users create their own events to make all sort of activities. Just like that, this app has two functionalities covered: it speeds the process of lost pet searching up and allows users to organize temporary activities for them and other users to take part into.

AGRADECIMIENTOS

Desde el 404Team queremos agradecer personalmente a amigos, familiares y profesores por la ayuda brindada y la paciencia. Muchas gracias.



Esta obra se distribuye bajo una licencia Creative Commons.

Se permite la copia, distribución, uso y comunicación de la obra si se respetan las siguientes condiciones:

- Se debe reconocer explícitamente la autoría de la obra incluyendo esta nota y su enlace.
- La copia será literal y completa
- No se podrá hacer uso de los derechos permitidos con fines comerciales, salvo permiso expreso de los autores.

El texto precedente no es la licencia completa sino una nota orientativa de la licencia original completa(jurídicamente válida) que puede encontrarse en:

<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es>

INDICE

1. Introducción	7
1.1. Objetivos	8
1.2. Motivación	8
1.3. Antecedentes	9
2. DESARROLLO DEL Proyecto	9
2.1. Herramientas tecnológicas	9
2.1.1. Android Studio	9
2.1.1. Firebase Realtime Database	12
2.1.1. Google Maps API	15
2.1.1. Java	16
2.2. Planificación	17
2.3. Descripción del trabajo realizado	18
2.3.1. Pantalla de inicio	23
2.3.2. Pantalla principal y Fragment del mapa	26
2.3.3. Pantalla de perfil	33
2.3.4. Pantalla de Mascotas	34
2.4. Resultados y validación	36
3. CONCLUSIONES	37
3.1. Innovación	
3.2. Trabajo futuro	
4. BIBLIOGRAFÍA Y WEBGRAFÍA	38
5. ANEXOS	39
5.1. Archivo AndroidManifest.xml	
5.2. Códigos fuente.	
5.2.1. HomeActivity.java	
5.2.2. MapaFragment.java	

1. INTRODUCCIÓN

La idea sobre este proyecto surgió cuando el equipo estaba buscando ideas. En la página HackForGood se vio una petición para hacer algún tipo de programa o app para ayudar a localizar a mascotas perdidas, y se decidió crear una aplicación Android que cubriera los objetivos propuestos por la misma petición mencionada antes y además que pudiera alcanzar algunos objetivos añadidos por el propio equipo.

1.1. Objetivos

La aplicación ha sido creada con los siguientes objetivos en mente:

- Crear anuncios sobre mascotas perdidas.
- Crear anuncios sobre eventos de mascotas.
- Geolocalizar el lugar que se describe en los anuncios.
- Compartir la ubicación de mascotas perdidas con otros usuarios.

Crear anuncios sobre mascotas perdidas: En caso de que el usuario pierda a su mascota, será capaz de crear un anuncio inmediatamente en el lugar de la desaparición. El usuario debe aportar además una foto de la mascota perdida para que sea reconocida fácilmente por el resto de los usuarios.

Crear anuncios sobre eventos de mascotas: El usuario puede crear eventos temporales y compartirlos con el resto de los usuarios (por ejemplo, organizar paseos de perros). No hay que adjuntar una imagen para este tipo de anuncios.

Geolocalizar el lugar que se describe en los anuncios: Los anuncios creados por los usuarios aparecen directamente en el mapa en forma de marcadores.

Compartir la ubicación de mascotas perdidas con otros usuarios: Si el usuario ve una mascota perdida, puede compartir la ubicación con el resto de los usuarios. En caso de que un usuario localice a una mascota perdida descrita en un anuncio se podrá poner en contacto con el dueño del animal a través del número de contacto del anuncio de dicha mascota.

1.2. Motivación

Lo que empujó al equipo a realizar este proyecto fue que todos los miembros de este son amantes de los animales. Cuando se descubrió la propuesta en HackForGood y viendo que no había muchas aplicaciones que pudiera servir como “Tablón de anuncios” para mascotas y como red social, se decidió que este proyecto sería muy interesante y, sobre todo, algo muy original.

1.3. Antecedentes

En la investigación llevada a cabo por el equipo, se encontraron algunas aplicaciones similares a la esta. Sin embargo, hay algunos cambios notables.

Una de las aplicaciones que se encontraron era un localizador de mascotas. La aplicación mostraba la posición de la propia mascota en el mapa, de forma que sería fácil localizarla. Sin embargo para poder utilizar la aplicación el usuario debía comprar una especie de dispositivo GPS, el cual era el localizador que mostraba el mapa. El principal matiz que separaba esta aplicación de este proyecto es que en caso de los miembros de este equipo no se ha buscado forzar al usuario a realizar una compra para que utilice la aplicación, sino que se ofrece un servicio similar (aunque tal vez no con la precisión de un GPS) de forma gratuita.

También es notable que en esa aplicación no hay prácticamente ningún elemento de red social que sirva para conectar usuarios, así que eso adicionalmente es un punto que las diferencia.

2. DESARROLLO DEL PROYECTO

Se comenzará esta sección comentando las diferentes herramientas de las que se ha valido el equipo para llevar a cabo este proyecto.

2.1. Herramientas tecnológicas

Las herramientas utilizadas son las siguientes:

- Android Studio
- Firebase Realtime Database
- Google Maps
- Java

2.1.1 Android Studio



Android Studio es el entorno de desarrollo integrado (IDE) oficial utilizado para programar aplicaciones del sistema operativo Android. Está basado en IntelliJ, otro entorno de desarrollo para Java. Tiene bastantes características que lo distinguen de otros entornos como como por ejemplo su propio emulador de Android sobre el que hacer pruebas, la opción de hacer proyectos para todas las versiones de Android, buena integración con GitHub (se pueden crear repositorios de GitHub directamente desde Android Studio) y también da soporte a código escrito en C++ y Kotlin.

La estructura de un proyecto de Android Studio (como por ejemplo este) es la siguiente:

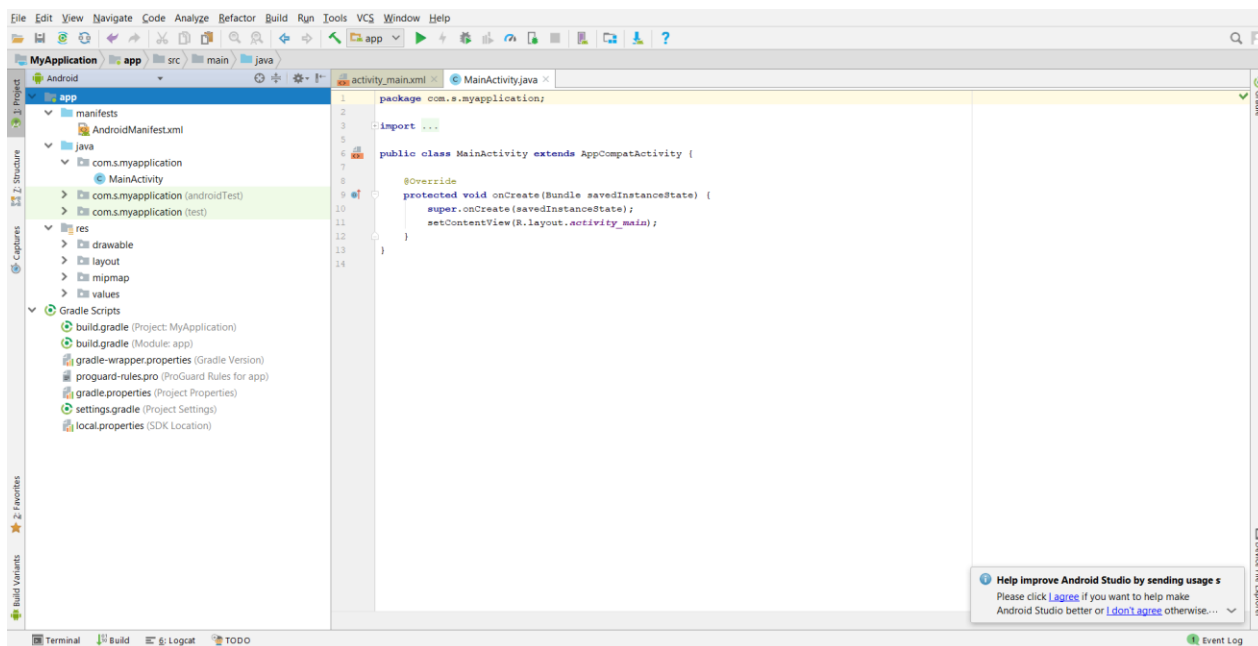


Ilustración 1: Vista de un proyecto sencillo en Android Studio.

Todos los proyectos contienen las siguientes carpetas: manifests, java y res. Además, cada proyecto cuenta con sus “Gradle Scripts”.

- **Manifests:** En esta carpeta se encuentra el archivo AndroidManifest.xml, en el que figuran distintos datos de la aplicación (por ejemplo, el nombre, número de Activities, o estilos).
- **Java:** Esta carpeta contiene los archivos Java que le dan su funcionalidad a la aplicación.
- **Res:** En esta carpeta se almacenan diferentes recursos utilizados por la aplicación. Entre ellos se encuentran los layouts (como las interfaces), imágenes o iconos, además de otros archivos XML como el archivo Strings (para guardar cadenas de caracteres) o el archivo Arrays.
- **Gradle:** Los scripts de Gradle son usados por el Android Studio como base de compilación. Permiten entre otras cosas compilar librerías externas (por ejemplo, de GitHub) o compilar una misma aplicación, pero con funcionalidades diferentes.

Como se ha mencionado anteriormente, Android Studio posee un emulador en el que se pueden probar las aplicaciones creadas, pero también puede reconocer los dispositivos Android externos (como un teléfono o PDA que utilice Android y tenga activadas las opciones de desarrollo) en los que Android Studio podrá instalar dicha aplicación. Siguiendo estos mismos pasos podemos a su vez generar archivos APK de las aplicaciones. Los archivos APK pueden ser fácilmente generados si son de modo *debug* (o modo pruebas).

Sin embargo, también podemos generar APK firmadas (*signed APK*) las cuales se pueden subir a Google Play. Para crear una APK firmada primero se debe tener un archivo *keystore*.

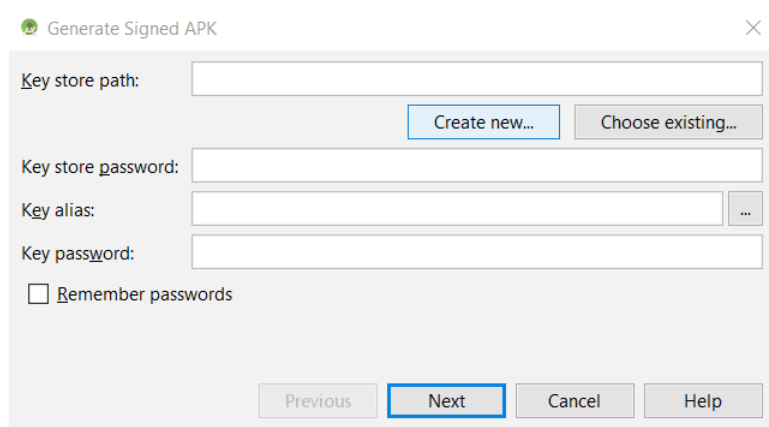


Ilustración 2: Ventana de Generar APK firmada.

Si no disponemos de una *keystore*, podemos crear una siguiendo el siguiente *wizard*:

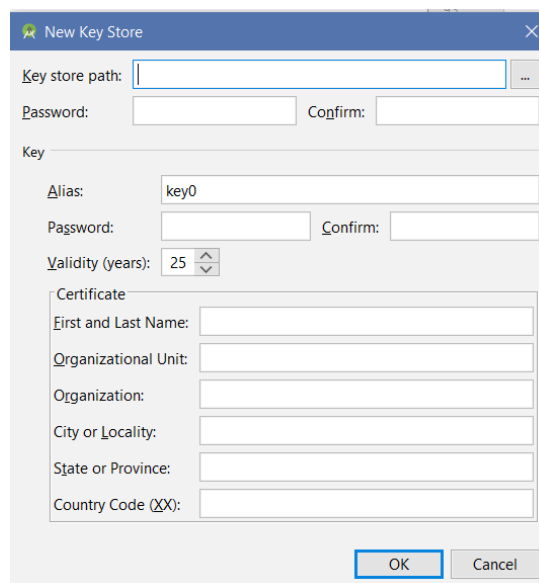


Ilustración 3: Ventana de crear Keystore.

Una vez creado, volvemos a la pantalla anterior, introducimos las contraseñas y el alias y proseguimos. En la siguiente pantalla debemos especificar la ruta, la versión (en caso de que tengamos versiones como *debug*, *release* o algo similar) y el soporte de *JAR*. De esta forma se puede generar una APK firmada.

Otra de las funciones que más se ha utilizado en el desarrollo de este proyecto ha sido el depurador de Android Studio. Este entorno te permite crear varios puntos de *debug* (*breakpoints*) a lo largo del código y después poder ir controlando punto a punto (o línea a línea) de código cómo funciona la aplicación. Las opciones del depurador se encuentran al lado del botón de Run:



Ilustración 4: De izquierda a derecha, botones de Run, Apply changes, Debug, Profile y Attach Debugger.

2.1.2 Firebase Realtime Database



La base de datos de Google, Firebase Realtime Database, es una base de datos no relacional que actúa como un fichero JSON en el que se almacenan los diferentes datos, los cuales se sincronizan en tiempo real. Firebase es compatible con sistemas operativos (Android, iOS), con software como Unity y con otros lenguajes de programación (C++ o JavaScript en caso de programación web).

En este caso, se ha integrado Firebase con el propio proyecto Android. Para hacer esto se debe crear una nueva base de datos de Firebase para el proyecto. Una vez creada, debemos integrarla a la aplicación deseado introduciendo el fichero JSON de Google Services dentro de la carpeta *app* del proyecto y modificar los archivos de *Gradle* para que se compilen los plugin necesarios para su funcionamiento. Completados estos pasos, la aplicación podrá trabajar conjuntamente con una base de datos de Firebase.

Dentro de las principales funcionalidades de Firebase se puede destacar su apartado de Database. Dentro de este apartado se encuentra el almacenamiento de datos, los cuales son almacenados como si de un fichero JSON se tratase. Los datos pueden ser introducidos manualmente o desde la propia aplicación (creando una instancia de la base de datos dentro del código). Los tipos de datos aceptados en Firebase son los siguientes:

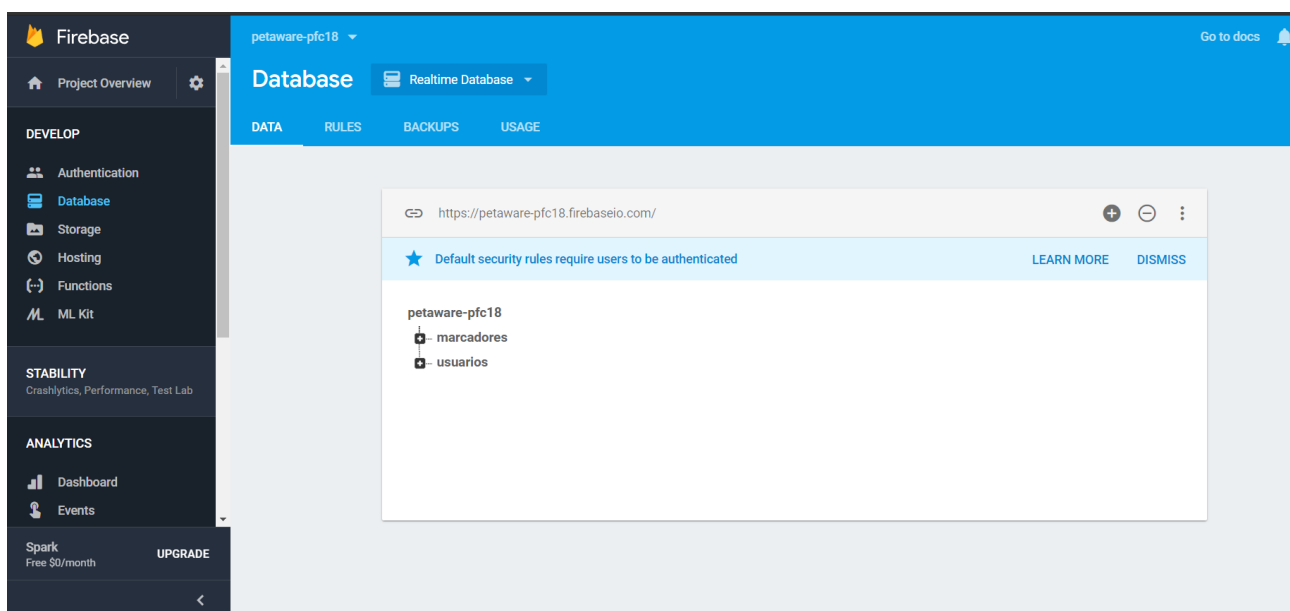


Ilustración 5: Vista de la Base de Datos de Firebase.

- Boolean.
- Byte.
- Fecha y hora.
- Float.
- Integer.
- Referencia(URL).
- Punto geográfico (latitud, longitud).
- Map.
- Null.
- String.

Otra funcionalidad que se ha utilizado en gran medida en este proyecto es la gestión de usuarios. El apartado de *Authentication* podemos encontrar, entre otras cosas

- Usuarios que se han registrado en la aplicación. Se puede ver en la pestaña *Users*.
- Métodos por los cuales pueden los usuarios registrarse o iniciar sesión (por ejemplo, por cuenta de Google o por correo electrónico y contraseña). Se puede ver en la pestaña *Sign-in Methods*.
- Gestión de contraseñas de usuarios (por ejemplo, podemos crear un mensaje propio con un enlace para restaurar la contraseña en caso de que el usuario la olvide). Esto se puede ver en la pestaña *Templates*.
- Numero de verificaciones de usuarios. Se puede ver en la pestaña *Usage*.

Authentication					WEB SETUP
USERS					
SIGN-IN METHOD					
TEMPLATES					
USAGE					
<div> <input type="text"/> Search by email address, phone number, or user UID </div>					<div> ADD USER </div>
Identifier	Providers	Created	Signed In	User UID ↑	
calle@falsa.com		May 24, 2018	May 24, 2018	4Nm4VQlaTdetfXVmLANyMvWkr...	
p4@a.es		May 22, 2018	May 22, 2018	BiD5FCaDZyWTFwtL54dgZ3GBnY...	
p5@a.es		May 22, 2018	May 22, 2018	D874QNdUUQZf9RlZVeLDY9udP6q1	

Ilustración 6: Vista de la pestaña *Users*.

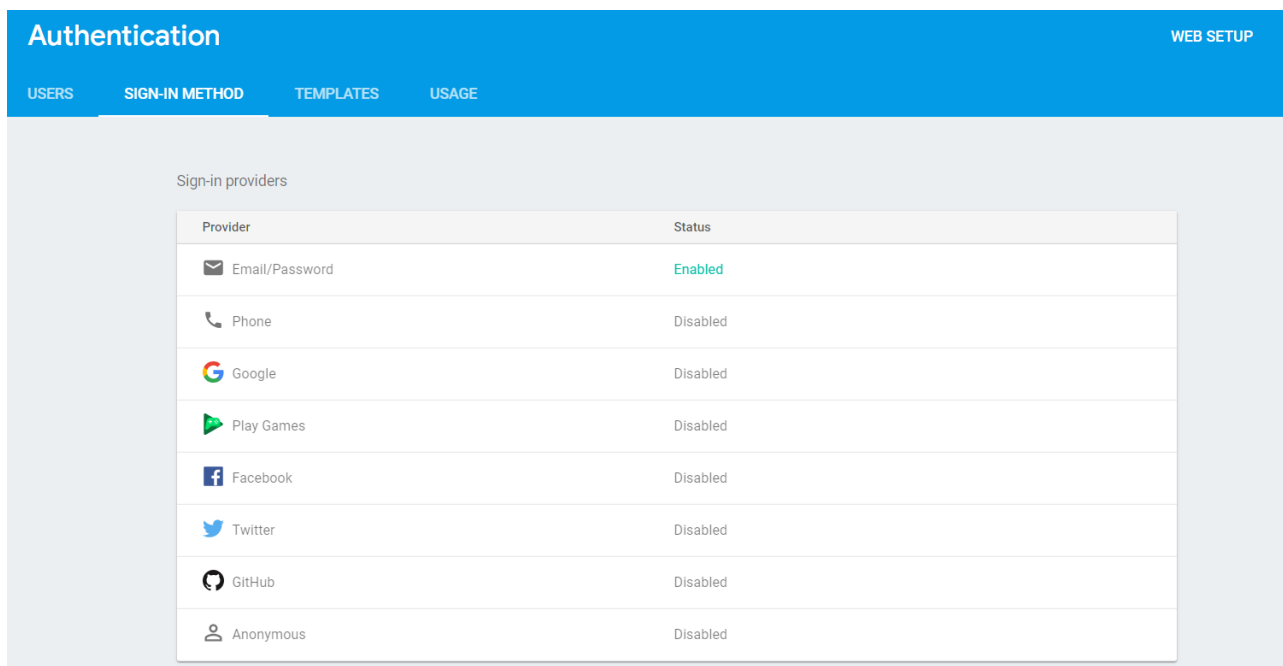


Ilustración 7: Vista de la ventana Sign-In Methods.

También se puede destacar el apartado de *Storage*, en el que se pueden almacenar archivos (tales como imágenes o documentos) y ser descargados o vistos por los usuarios usando su URL correspondiente. Los usuarios también pueden subir archivos a *Storage* a través de la aplicación. Para usar esta función es necesario dar permisos al proyecto en el archivo `AndroidManifest.xml`.

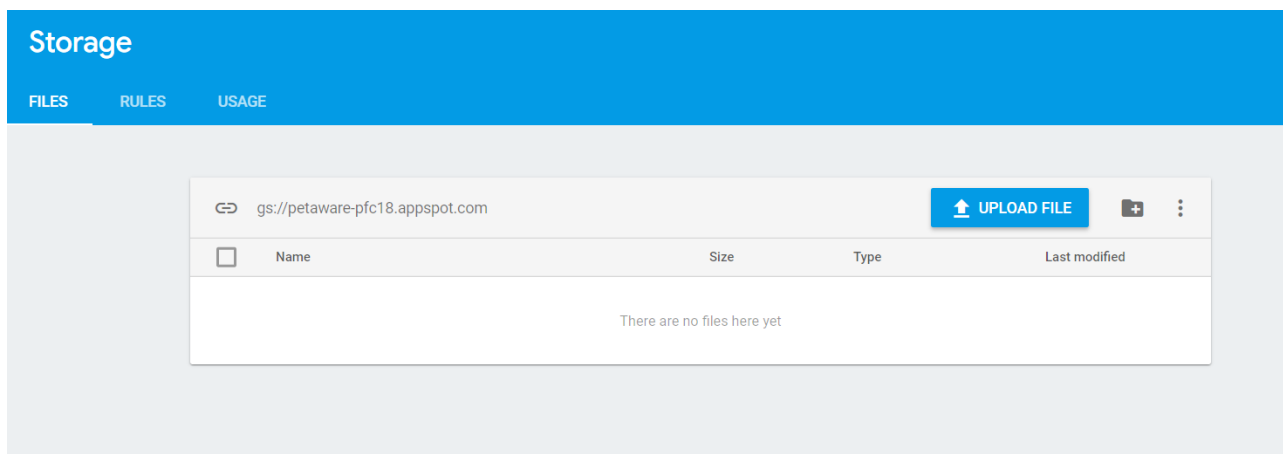


Ilustración 8: Vista del apartado de Storage.

2.1.3 Google Maps API



La API de Google Maps permite utilizar los servicios de Google Maps en un proyecto. Entre las características de Google Maps en Android se encuentran:

- Conocer ubicación del usuario.
- Ofrecer al usuario gran cantidad de datos.
- Mostrar diferentes tipos de mapas (por ejemplo, con edificios 3D, con imágenes de satélite o con imágenes de *StreetView*).
- Uso de marcadores personalizados.

Para poder usar Google Maps en una aplicación es necesario generar una clave (*key*) propia para la API. Para hacer esto se debe visitar la página oficial de Google Maps API. Existen dos tipos diferentes de clave:

- **API estándar:** es gratuita y se debe configurar para el proyecto en el que vaya a ser usada (en el caso de este proyecto, se ha usado una API estándar).
- **API Premium Plan:** es de pago, pero los usuarios que la utilicen pueden usar su ID de cliente en lugar de la propia clave API para utilizar Maps.

Una vez se disponga de una clave de API, en caso de que el proyecto se Android como este, se debe modificar el archivo `AndroidManifest.xml` y hacer lo siguiente:

- Añadir permisos de Internet (dentro del campo *uses-permission* se debe añadir `INTERNET`).
- Añadir permisos de ubicación (dentro del mismo campo `ACCESS_COARSE_LOCATION`, para que la localización se haga a través de datos móviles o Wi-Fi; o `ACCESS_FINE_LOCATION`, para que además de eso se use GPS).
- Añadir permisos de escritura (en el mismo campo se debe añadir `WRITE_EXTERNAL_STORAGE`).
- Añadir permisos para verificar el estado de la red (en el mismo campo añadir `ACCESS_NETWORK_STATE`).
- Especificar el requisito de OpenGL ES 2 (se añade el campo *uses-feature* con los valores `android:glEsVersion="0x00020000"` y `android:required="true"`).
- Especificar el número de la versión de Google Play Services (se añade el campo *meta-data* con los valores `android:name="com.google.android.gms.version"` y `android:value=valor`, donde valor es el número de versión).

Una vez configurado el proyecto ya se pueden usar los objetos `GoogleMap` y empezar a usar sus diferentes funcionalidades. Lo primero que se debe hacer es añadir un elemento `MapView` dentro de uno de los layouts de la aplicación. El `MapView` es el elemento sobre el que se representará el mapa de Google. Una vez hecho esto, se debe asignar un

objeto de la clase `GoogleMap` a dicho `MapView`. Esto se debe de hacer a través del código presente en la clase Java que le dé funcionalidad al layout del `MapView`, ya que dicha clase deberá implementar el método `onMapReady()` de `OnMapReadyCallback` (interfaz de Google Maps). Este método recibe un objeto `GoogleMap` y describirá lo que se hará con el mapa una vez esté cargado. Dentro de este método se pueden desempeñar un gran número de funciones. Algunas de las que se han utilizado en este proyecto son las siguientes:

- Establecer el tipo de mapa (por ejemplo, mapa normal o satélite) con el método `setMapType`.
- Localizar la ubicación del usuario.
- Añadir marcadores y personalizarlos (cambiado el icono, título o posición).

2.1.4 Java



Java es un lenguaje de programación orientado a objetos creado en 1995 por la compañía Sun Microsystems. Java se ha extendido enormemente en muchos ámbitos debido a su gran potencial y, entre otras cosas, por ser un lenguaje multiplataforma. Esto último significa que los programas escritos en Java pueden ser ejecutados en diferentes sistemas operativos, con diferentes tipos de CPU sin tener que volver a ser compilados.

Aparte de las características mencionadas anteriormente, también debemos tener en cuenta otras cualidades importantes de Java, las cuales son:

- Es un lenguaje relativamente sencillo. Es fácil de usar y de aprender.
- Con Java no es necesario que el programador maneje las direcciones de memoria manualmente. Además no se permite que se manipulen las direcciones de memoria, lo cual le aporta seguridad.
- Permite trabajar con diferentes hilos (*Threads*) al mismo tiempo (Multihilo).
- Da soporte a diferentes protocolos para poder trabajar en red.
- Algunos programas de Java pueden ejecutarse desde un navegador web (*Applet*).

Como se ha dicho en el inicio, Java es un lenguaje orientado a objetos. Esto quiere decir que los programadores de Java trabajan con diferentes clases. Los objetos son instancias de estas clases y se pueden usar para acceder a sus atributos o usar sus métodos. Se pueden establecer relaciones con las clases y los objetos. Esto pasa por ejemplo cuando una clase hereda las cualidades de otra (atributos y métodos). Se debe mencionar además el caso de las interfaces. Una interfaz es una clase Java que contiene métodos

abstractos. Si otra clase Java implementa esa interfaz (*implements*) debe contener los métodos de la interfaz. Sin embargo, la funcionalidad de los métodos se debe establecer en la clase que implementa la interfaz (ya que los métodos son abstractos).

Las librerías son clases que añaden funcionalidad a los programas escritos en Java. Algunas de estas librerías son propias de Java, pero también se pueden usar librerías externas (por ejemplo, usando un archivo JAR). En el caso de este proyecto, se utiliza Java con las librerías de Android, de forma que el Java usado en Android Studio está “orientado” a aplicaciones de este sistema. Esto quiere decir que se pueden usar los métodos de las clases de las librerías de Android.

Java trabaja con diferentes tipos de datos. Estos datos pueden ser:

Tipos numéricos enteros:

- **byte.**
- **short.**
- **int.**
- **long.**

Tipos numéricos decimales:

- **double.**
- **float.**

Caracteres:

- **char.**
- **String** (cadena de char. Este “tipo” es en realidad una clase de Java en sí con sus propios métodos).

Tipos lógicos (su valor puede ser true o false):

- **boolean.**

Java permite crear conjuntos de estos tipos (tales como Arrays, Lists, Maps o Sets) pero también es posible crear un conjunto de una clase creada por el mismo programador utilizando los objetos de la misma clase (Por ejemplo, si creamos una clase llamada “Coche” con los atributos marca y matrícula, podemos guardar un objeto Coche dentro de un Array del tipo Coche que contenga esos atributos). Además es posible guardar conjuntos dentro de otros.

Aunque en caso de este proyecto se use Android Studio como IDE, algunos de los IDE más conocidos para trabajar con Java son Eclipse o NetBeans.

2.2. Planificación

El equipo se ha dividido en los siguientes roles:

- Borja Olalla: Designer.
- Carlos Ortiz, Scrum Master.
- Sergio Cuadrado: Git Master.

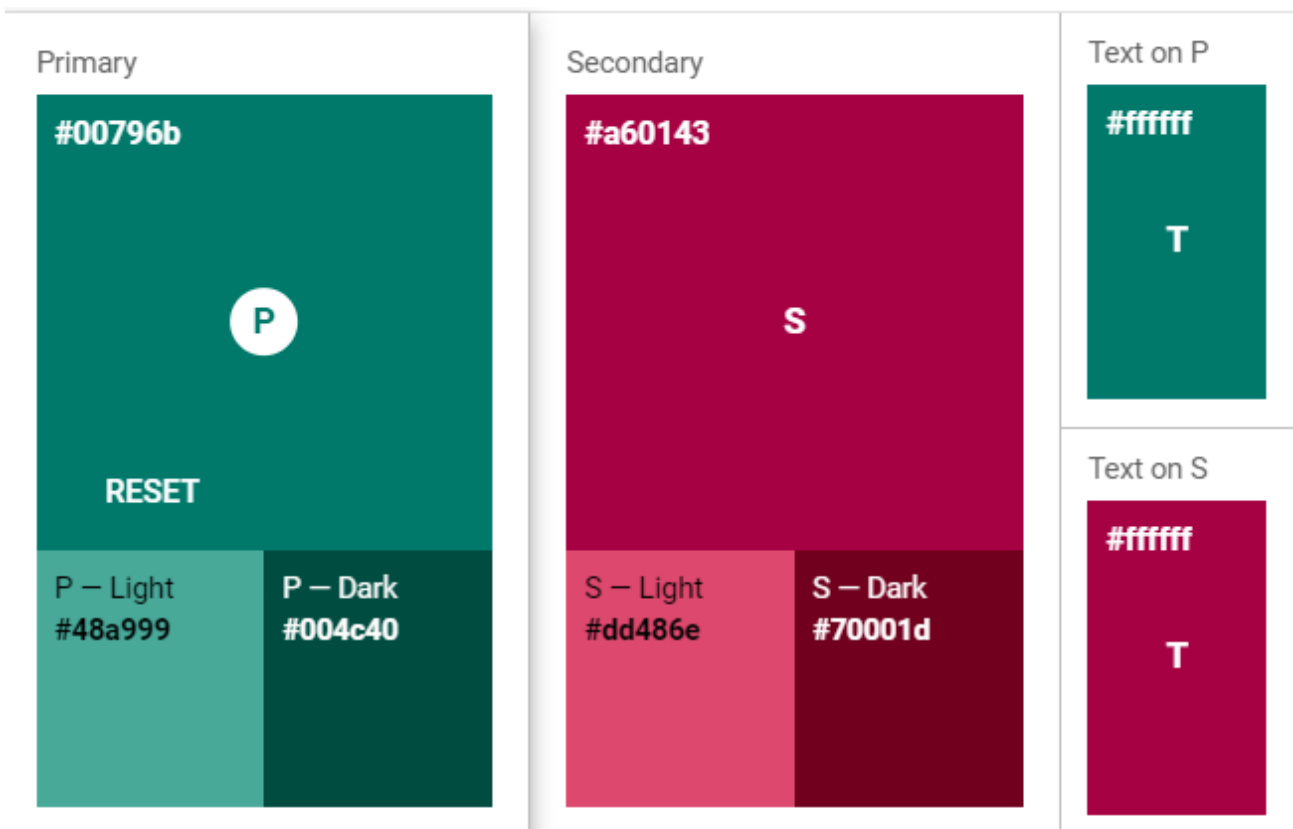
Independiente de sus roles, todos los miembros del equipo han trabajado en la realización del código. Las horas de trabajo en el proyecto se han centrado en los fines de semana o festivos y ocasionalmente en días laborales.

Además, el equipo ha utilizado las siguientes herramientas para organizar y planificar el proyecto:

- **GitHub:** es donde se encuentra el repositorio de nuestro proyecto. El equipo cuenta con diferentes ramas para probar cambios ocasionales y la rama master que es donde se encuentra el código principal.
- **Waffle:** usando la integración entre Waffle y GitHub, el equipo ha trabajado con diferentes “issues” o tareas del proyecto.
- **Slack:** el equipo dispone de un canal propio de Slack donde se han compartido avances en el proyecto con el tutor.
- **WhatsApp:** el equipo también dispone de un grupo de WhatsApp donde los miembros han podido compartir opiniones y preocupaciones acerca del proyecto.

2.3. Descripción del trabajo realizado

En primer lugar, lo que primero se empezó a pensar fueron los colores que se usarían en la aplicación y el número de pantallas necesario para que desempeñara su función. El esquema de colores que resultó fue el siguiente.



La paleta de colores primarios es, por tanto:

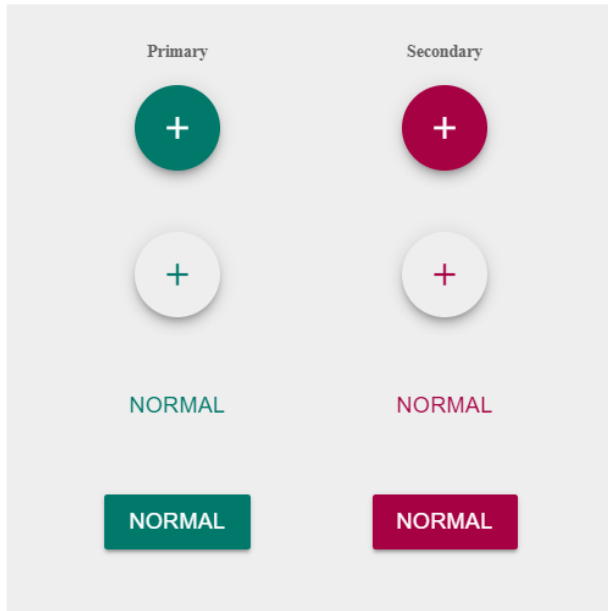
Primary	Aa Large Text	Aa Normal Text
Teal #00796b	White Text min 62% opacity	min 88% opacity
	Black Text min 68% opacity	NOT LEGIBLE ⚠
P – Light	Aa Large Text	Aa Normal Text
Teal #48a999	White Text NOT LEGIBLE ⚠	NOT LEGIBLE ⚠
	Black Text min 50% opacity	min 67% opacity
P – Dark	Aa Large Text	Aa Normal Text
Teal #004c40	White Text min 41% opacity	min 59% opacity
	Black Text NOT LEGIBLE ⚠	NOT LEGIBLE ⚠

Y la de colores secundarios:

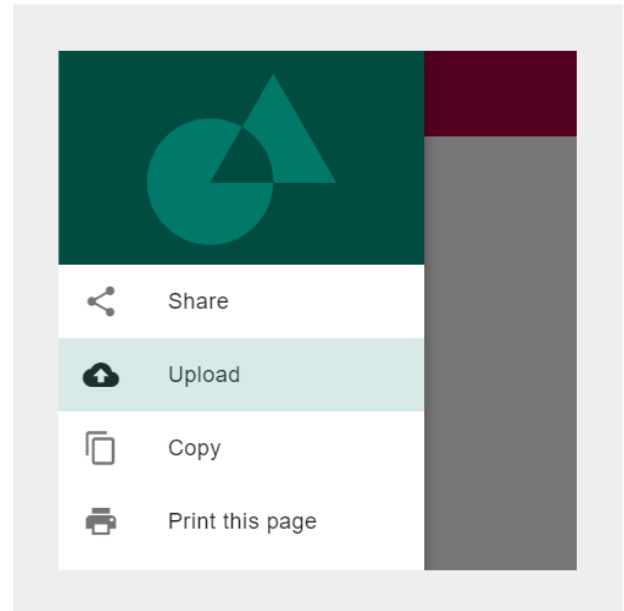
Secondary	Aa Large Text	Aa Normal Text
#a60143	White Text min 55% opacity	min 73% opacity
	Black Text NOT LEGIBLE ⚠	NOT LEGIBLE ⚠
S – Light	Aa Large Text	Aa Normal Text
#dd486e	White Text min 78% opacity	NOT LEGIBLE ⚠
	Black Text min 57% opacity	min 82% opacity
S – Dark	Aa Large Text	Aa Normal Text
#70001d	White Text min 42% opacity	min 56% opacity
	Black Text NOT LEGIBLE ⚠	NOT LEGIBLE ⚠

Usando estos esquemas se puede hacer pruebas para observar como resultarían estos colores aplicados en los elementos de los propios layouts del proyecto. A continuación, se presentarán algunos ejemplos:

Buttons



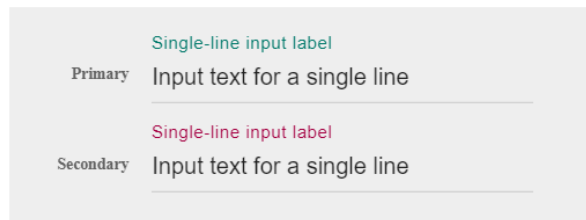
Menu



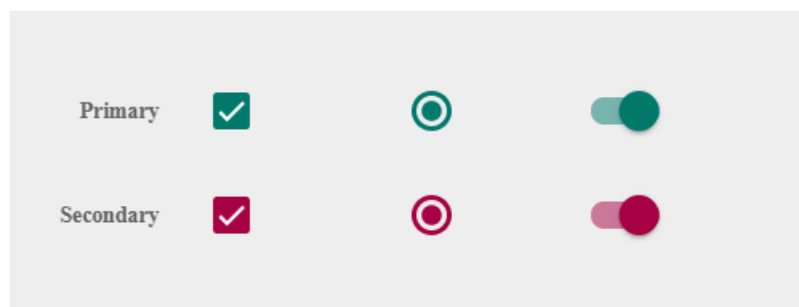
Selection

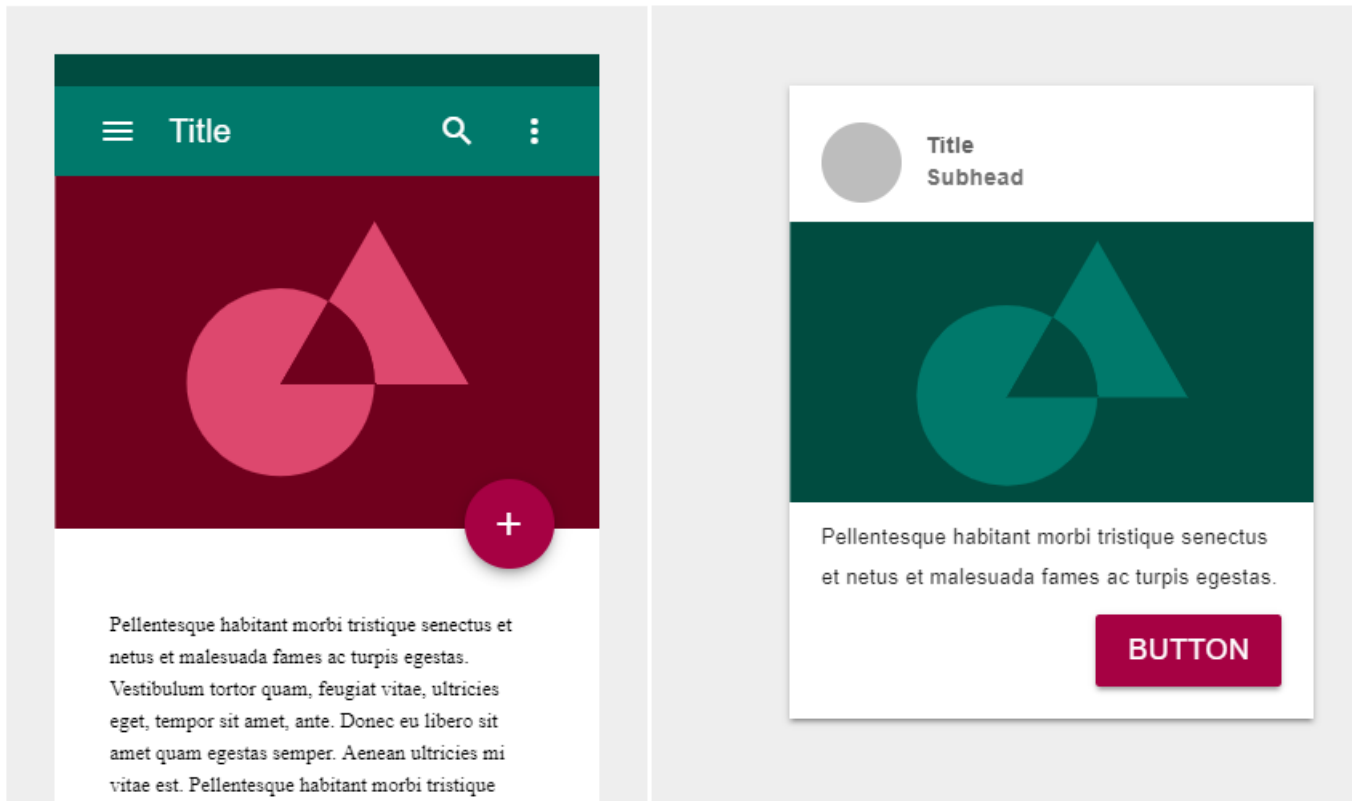


Text Field



Switches and sliders





Desde su inicio, los miembros del equipo decidimos que el mapa sería una pieza fundamental para esta aplicación. Esto se refleja en nuestros mockups (realizados con la herramienta *Moqups*) iniciales. Otro de los elementos que estaban planeados desde el principio era un listado de los diferentes anuncios. Sin embargo, la idea del listado fue descartada antes de que pudiera incluso ser implementada en la propia aplicación. Un ejemplo del listado a implementar sería lo siguiente:



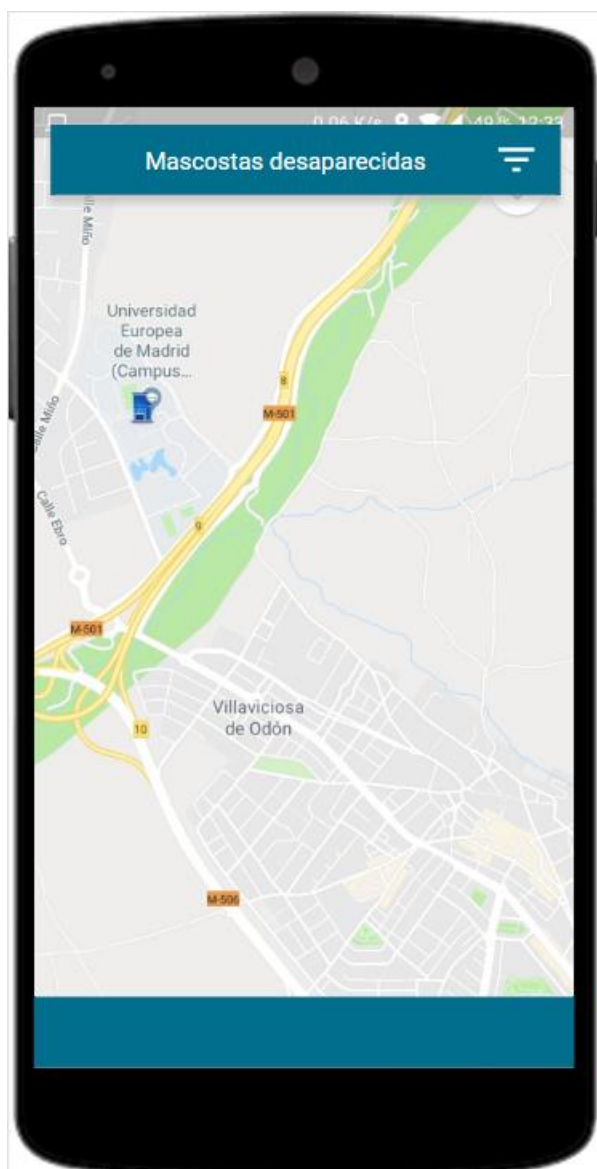


Ilustración 10: Pantalla del mapa en el mockup.

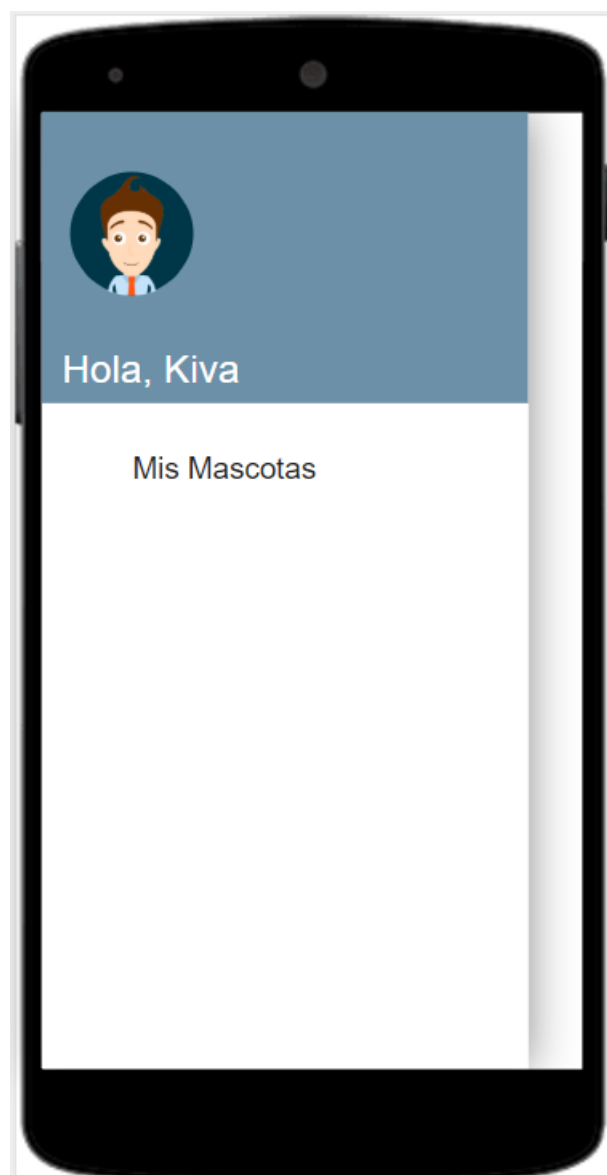


Ilustración 9: Menú de navegación antiguo (mockup).

2.3.1 Inicio de sesión

Con esta idea fija, el equipo comenzó a trabajar con Android Studio en el modo de inicio de sesión que utilizaría la aplicación. Como todo el tema de registro y sesiones de usuarios se haría mediante Firebase, se eligió inicialmente que se hiciera inicialmente usando el método de correo electrónico y contraseña. Más tarde se empezó a plantear la idea de un inicio de sesión a través de una cuenta de google del usuario. Se pantallas sencillas para registrar al usuario para que este proceso resultara más ameno.

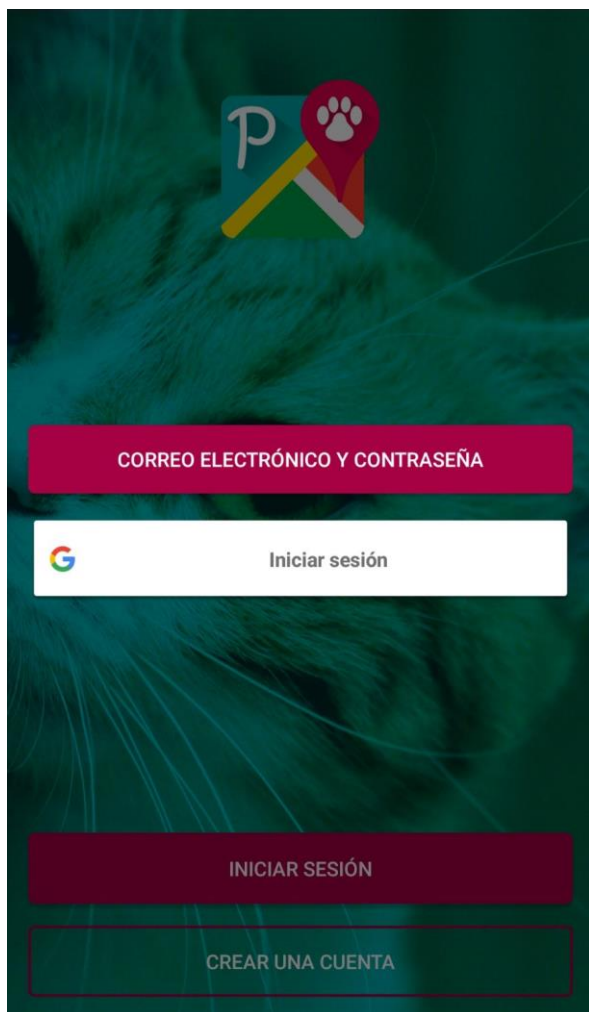


Ilustración 11: Vista de la pantalla de Inicio de una versión intermedia.

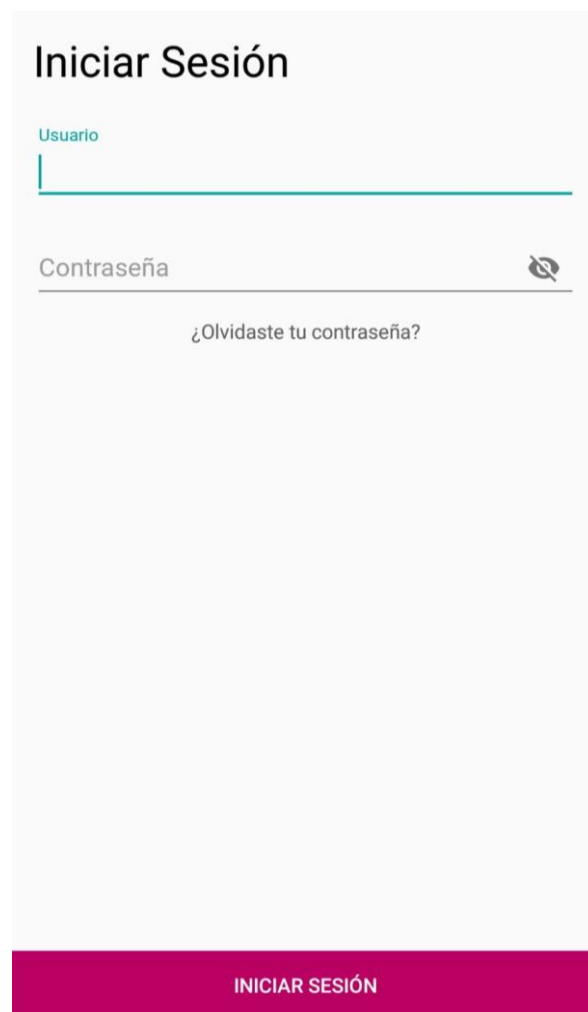



Ilustración 12: Vista de la pantalla de Iniciar Sesión.

Create an account

Name

E-mail

Password 

SEND DATA

Ilustración 14: Pantalla de creación de usuario.

Login Options

Email and Password

or

Google Account

LOG IN

CREATE AN ACCOUNT

Ilustración 13: Pantalla de pantalla de inicio antigua.



Ilustración 15: Versión actual de la pantalla de Inicio.

Una vez el usuario ha sido creado. Se muestra una animación de carga. Mientras esta animación de carga. Si se ha creado el usuario de forma correcta y los datos han sido guardados, la pantalla cambiará a color verde y posteriormente pasará a la pantalla principal. En caso contrario la pantalla cambiará rojo y volverá a la creación de usuario. Las animaciones también aparecen cuando el usuario inicia sesión.

2.3.2 Pantalla principal y Fragment del mapa

El siguiente paso fue trabajar en la pantalla principal. El modelo de menú desplegable vertical fue sustituido por un menú de navegación en la parte de abajo. Las opciones de este menú conectaban con tres fragments (el fragment principal mostraba el mapa, los otros mostrarían el listado de anuncios y el perfil del usuario).



Ilustración 16: Menú de navegación antiguo.



Ilustración 17: Menú de navegación actual (versión en inglés).

Para poder trabajar con un mapa de Google, el equipo creó una clave de API estándar propia para el proyecto. Se editaron los siguientes campos en el archivo AndroidManifest.xml para poder dar permisos. También se deben dejar claros las versiones de GL y la clave de API para Google Maps:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission
  android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.WHITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<uses-feature
  android:glEsVersion="0x00020000"
  android:required="true" />

<!-- Maps Google API key -->
```

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="@string/google_map_api" />
```

En la clase Java designada al fragment del mapa se implementaron los métodos de la Interfaz *OnMapReadyCallback* de Google y se implementa su método *onMapReady()*. En este método se trabaja con los marcadores, se declara el tipo de mapa y se intenta conseguir la localización del usuario.

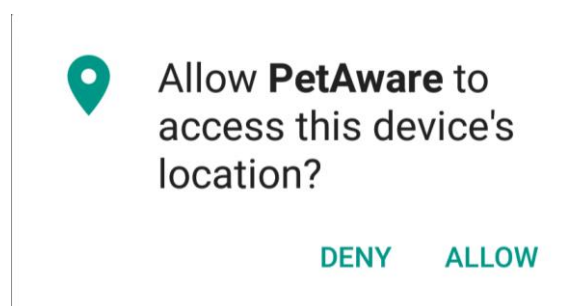
```
@Override
public void onMapReady(GoogleMap googleMap) {
    if (ActivityCompat.checkSelfPermission(getActivity(),
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(getActivity(),
ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        v_vista_error_carga.setVisibility(View.VISIBLE);
        return;
    } else {
        //en caso de que no existan errores en la carga del mapa, el elemento
MapView se hace visible
        mGoogleMap = googleMap;
        mMapView.setVisibility(View.VISIBLE);
        MapsInitializer.initialize(getApplicationContext());

        //Se establece el tipo de mapa
        UiSettings uiSettings = mGoogleMap.getUiSettings();
        mGoogleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);

        //Se llama al método para conseguir la ubicación del usuario
        MiUbucacion();

        //Resto del código del método    }
```

De esta forma se consiguió un fragment con un mapa de Google. Adicionalmente se añadió que saltara un dialog en la HomeActivity que avisara al usuario que la aplicación requería permisos para conocer al usuario. Si el usuario decidiera no dar permisos, o hubiera algún tipo de error a la hora de crear el Mapa.



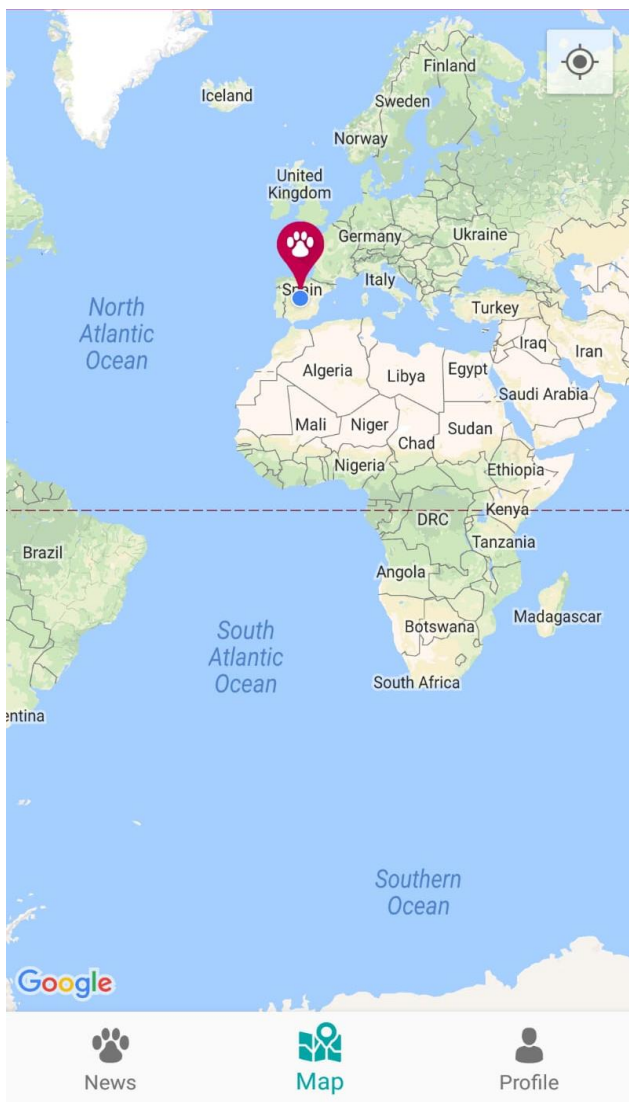


Ilustración 19: Vista del Fragment del mapa.

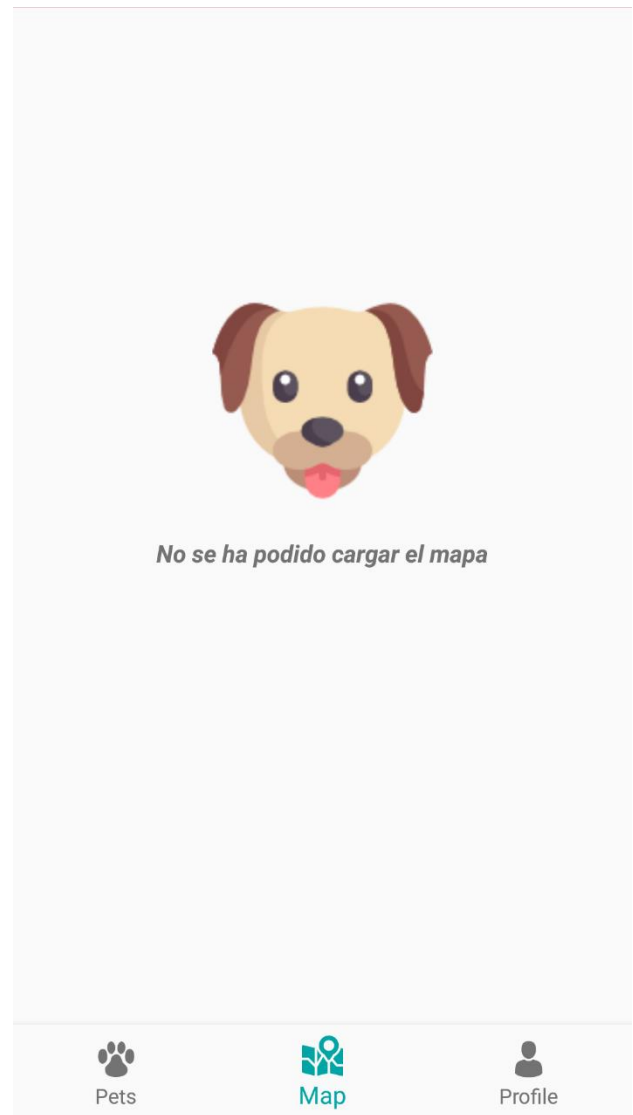


Ilustración 18: Vista del Fragment del mapa con error.

Otro tema que se trabajó junto con los mapas fueron los dialog que muestran los anuncios. Como se ha mencionado anteriormente, la información de los anuncios iba a ser contenida en un ListView o RecyclerView en la pantalla de Anuncios (actualmente llamada pantalla de Mascotas). Sin embargo, el equipo decidió mostrar la información del anuncio directamente en el mapa a través del uso de Listeners. A continuación se mostrará los métodos a los que se llama para recoger la información de una mascota perdida al pulsar sobre el marcador en el mapa:

```
public void CogerMarcadores() {

    all_marcadores =
    FirebaseDatabase.getInstance().getReference("marcadores").child("perdidas");
    all_marcadores.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            marcadores = new ArrayList<>();

            for (DataSnapshot dato : dataSnapshot.getChildren()) {
                Marcadores_perdidos mp =
                dato.getValue(Marcadores_perdidos.class);
                marcadores.add(mp);
                Log.v("datosUsuarios",mp.toString());
            }
            MeterMarcadores();
        }

        public void onCancelled(DatabaseError databaseError) {
        }
    });
}

public void MeterMarcadores() {
    mGoogleMap.clear();
    Log.v("ESTO__2", "" + marcadores.size());
    for (int i = 0; i < marcadores.size(); i++) {
        LatLng ubi = new LatLng(marcadores.get(i).getLatitud(),
        marcadores.get(i).getLongitud());

        Marker map_marcador = mGoogleMap.addMarker(new MarkerOptions()
            .position(ubi)

            .icon(BitmapDescriptorFactory.fromResource(R.mipmap.ic_logomaps)));

        map_marcador.setTag(marcadores.get(i).getId_mascota() + "##" +
        marcadores.get(i).getOwner());
        Log.v("ESTO__3", "" +ubi);
        Log.v("ESTO__3", "" +marcadores.get(i).getId_mascota() + "##" +
        marcadores.get(i).getOwner());
    }
}
```

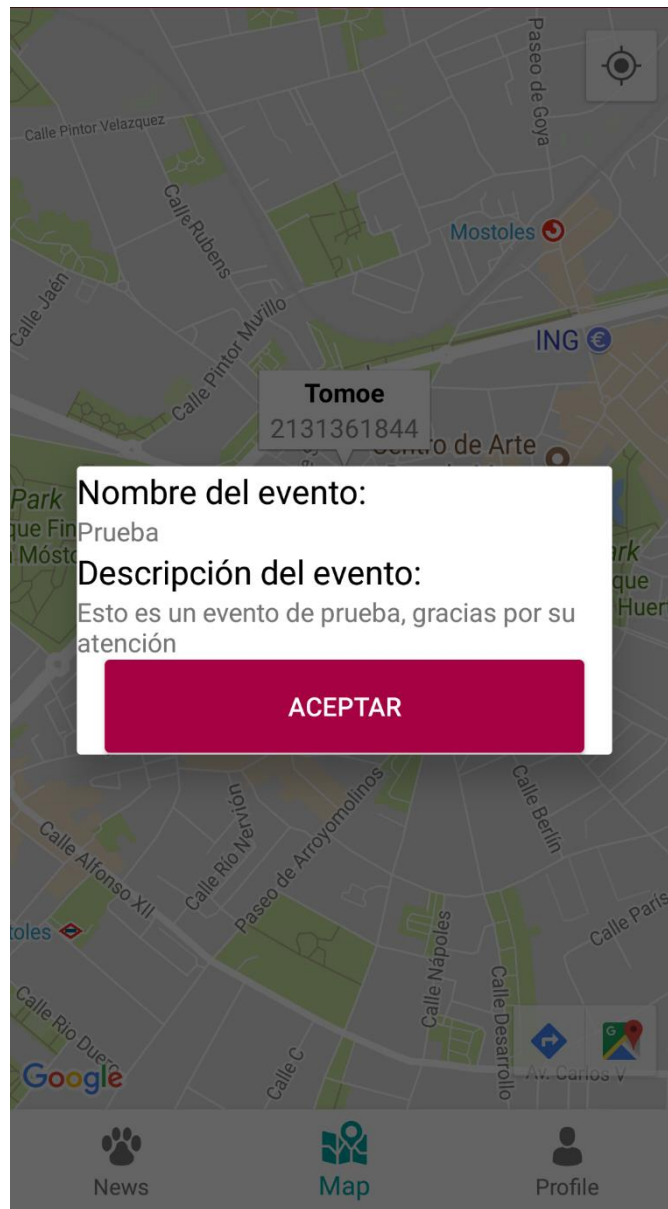


Ilustración 20: Dialog de anuncio antiguo.

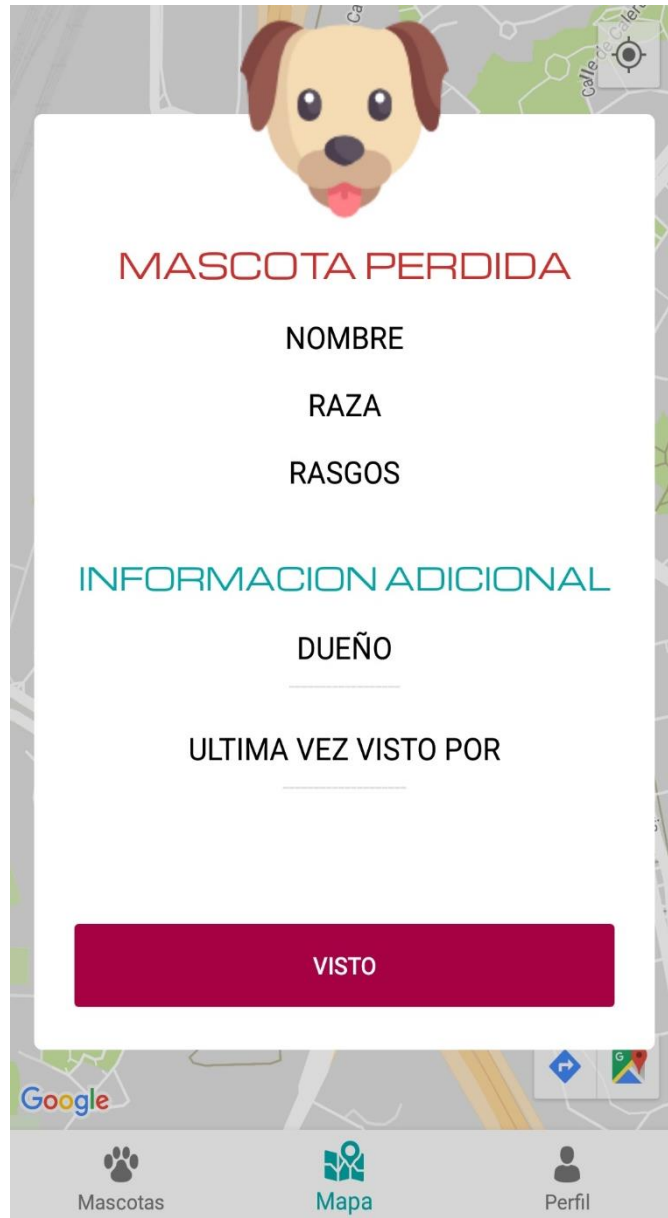


Ilustración 21: Dialog de anuncio actual.

Como se puede observar, en la versión actual se han cambiado la fuente de los títulos a la fuente Michroma.

Si se toca sobre el Fab Button de añadir con el icono de “+” Aparacerán nuevos botones.



Estos botones llevan a diferentes Activities con sus respectivos layouts que permiten al usuario crear anuncios de mascota perdida (rojo), eventos como paseo con mascotas (verde) y crear un marcador en caso de que se haya visto una mascota (azul). Para crear marcadores se deben introducir sus datos, subir una foto (solo para las mascotas perdidas) y posteriormente se debe pulsar el botón *Guardar*. De esta manera los datos quedan guardados.

2.3.3 Pantalla de Perfil

Otra de las pantallas desarrolladas ha sido la pantalla de perfil de usuario. Esta pantalla contiene el nombre y la foto del propio usuario. En esta pantalla también se puede encontrar el botón de cerrar sesión.

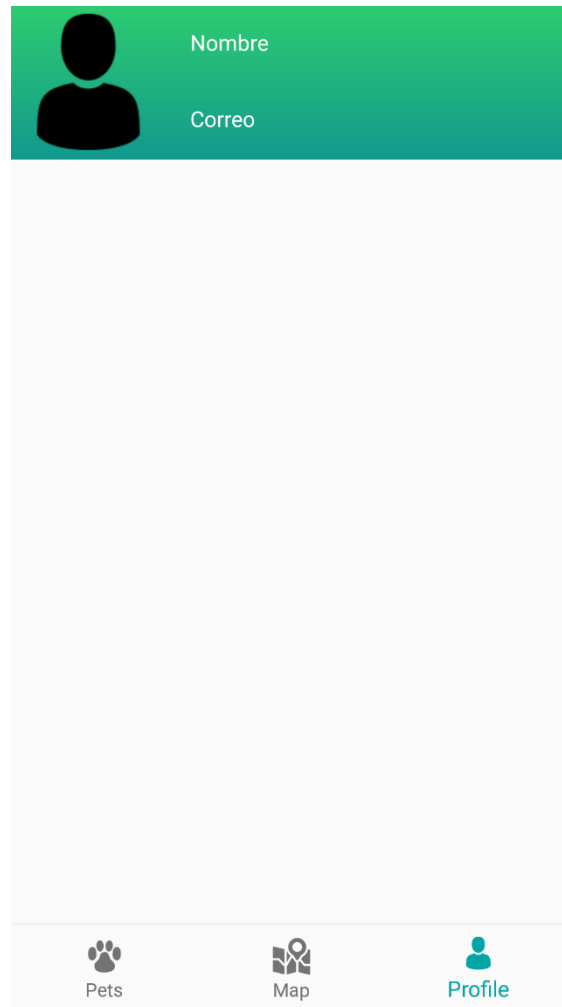


Ilustración 22: Versión inicial de la pantalla de perfil.

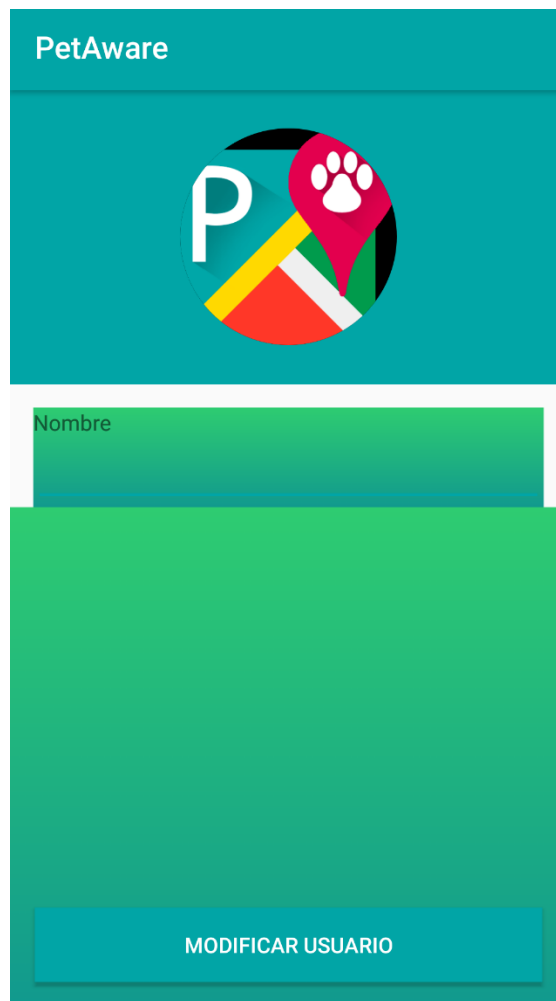


Ilustración 23: Pantalla de modificar perfil.

2.3.4 Pantalla de Mascotas

El último fragment incorporado se corresponde con la pantalla de mascotas. En esta pantalla el usuario puede de dar de alta sus diferentes mascotas. Introduciendo una foto y sus datos. Las mascotas dadas de alta se podrán seleccionar como perdidas a la hora de crear un anuncio de mascota perdida.

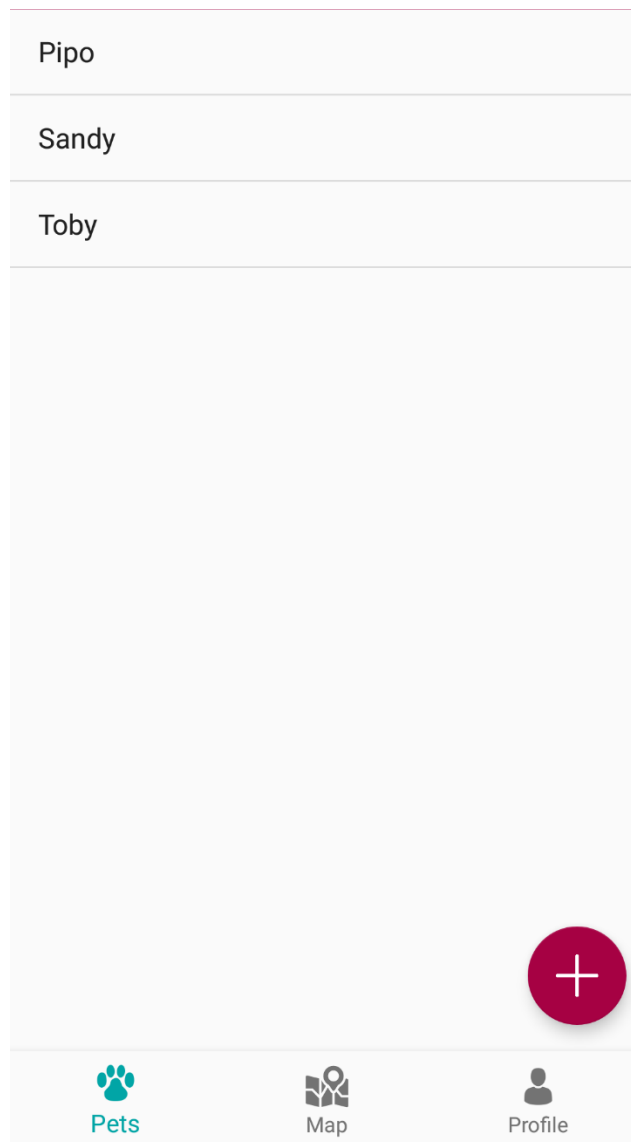


Ilustración 25: Pantalla de mascotas.

Pet Registry



Name

Race

Characteristics

SAVE

Ilustración 24: Dialog de registro de mascota

2.4. Resultados y validación

Hasta el momento, la aplicación ha funcionado sin problemas mayores. Se han realizado pruebas en dispositivos con Android 7.0 y 6.0 para poder ver la compatibilidad de la aplicación entre diferentes versiones. Hasta el momento no ha surgido problema alguno a la hora de usar la aplicación.

La aplicación tarda cerca de 1 segundo con 200 milisegundos en arrancar. También hay un leve problema a la hora de recuperar datos de Firebase. Estos no se recuperan de manera inmediata y hay que esperar un poco (unas décimas de segundo, en realidad) para que se muestren los datos.

3. CONCLUSIONES

Después de partir con la idea inicial del mockup en intentar plasmar dicha idea en la propia aplicación, se han cumplido hasta ahora los objetivos que el equipo tenía marcados para este proyecto. Se comenzó realizando los métodos de inicio de sesión y las pantallas de registro, inicio de sesión y todo lo relacionado con la gestión de usuarios. Una vez terminado se empezó con el verdadero *core* del proyecto: la pantalla principal. Esta pantalla es sin lugar a dudas la más importante de toda la aplicación, ya que en ella se encuentran los tres fragments principales (mapa, mascotas y perfil). De estos tres, el más problemático, pero también más importante es el fragment del mapa.

Este fragment fue complicado en diferentes sentidos. Por un lado, ninguno de los miembros del equipo se había enfrentado anteriormente con un mapa de Google a nivel de código y por otro hubo algunos problemas con la clave de API. Sin embargo, se consiguieron solucionar todos los problemas que surgieron y el equipo adquirió conocimientos de cómo trabajar con mapas.

La aplicación en su estado actual es relativamente similar al mockup inicial, ya que el equipo ha intentado mantenerse lo más fiel posible a la idea representada en el mockup. Hay algunos cambios entre ellos la desaparición del listado de anuncios, ya que ahora mismo la información de los anuncios se puede ver directamente en el mapa. El listado de anuncios es ahora un listado de mascotas, donde los usuarios pueden dar de alta sus mascotas (lo cual estaba también en los planes iniciales). También se hicieron algunos cambios menores en la paleta de colores.

Además de todo esto, las pruebas realizadas en diferentes versiones de Android ha sido satisfactoria.

3.1. Innovación

Uno de los aspectos más novedosos de la aplicación es la agilización del proceso de búsqueda de mascotas. Mucha gente utiliza redes sociales para compartir este tipo de cosas, y el hecho de que exista una aplicación destinada a este tipo de situaciones puede ser de gran ayuda para los propietarios que se encuentren en casos similares.

También está el elemento que hace que no toda la aplicación esté destinada a mascotas perdidas: el hecho de que se puedan crear eventos diferentes a los anuncios de mascotas perdidas, como quedadas de propietarios con sus mascotas. Esto le da un elemento de red social, ya que crea relaciones entre los usuarios.

3.2. Trabajo futuro

Las siguientes funcionalidades quedan pendientes para las siguientes versiones:

- Hacer que los usuarios puedan contactar entre sí a través de mensajes de texto.
- Mejorar la carga de datos de Firebase.

4. BIBLIOGRAFÍA Y WEBGRAFÍA

- Android Studio: Guía del usuario. (2018) <https://developer.android.com/studio/>. Fecha de consulta: mayo 28, 2018 de <https://developer.android.com/studio/intro/?hl=es-419>
- MONTERO MIGUEL, ROBERTO. (2015). Guía Práctica: Java 8. Madrid: Ediciones Anaya Multimedia.
- Google Maps Platform: Android SDK. (2018) <https://developers.google.com/maps/?hl=ES>. Fecha de consulta: mayo 29, 2018 de <https://developers.google.com/maps/documentation/android-sdk/intro?hl=ES>
- Firebase: Documentación para Android. (2018) <https://developers.google.com/maps/?hl=ES>. Fecha de consulta: mayo 29, 2018 de <https://firebase.google.com/docs/android/setup?hl=es-419>

5. ANEXOS

5.1 Archivo AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.team.a404.a404team">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.WHITE_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_logo_app"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_logo_app"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <!-- Maps Google API -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_api-key" />
        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
        <!-- Maps Google API -->

        <activity
            android:name=".SplashScreen"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".Zona_lobby.LoginActivity"
            android:screenOrientation="portrait"
            android:windowSoftInputMode="stateVisible|adjustResize" />
        <activity
            android:name=".Zona_lobby.RegistroActivity"
            android:screenOrientation="portrait"
            android:windowSoftInputMode="stateVisible|adjustResize" />
        <activity
            android:name=".Zona_lobby.DetallesPerfilActivity"
            android:screenOrientation="portrait"
            android:windowSoftInputMode="stateVisible|adjustResize" />
    </application>
</manifest>
```



```

        <activity
            android:name=".VentanasEstado.ActivitySuccess"
            android:screenOrientation="portrait" />
        <activity
            android:name=".Zona_lobby.lobby"
            android:screenOrientation="portrait" />
        <activity
            android:name=".HomeActivities.HomeActivity"
            android:label="@string/title_activity_main_map"
            android:screenOrientation="portrait" />
        <activity
            android:name=".HomeActivities.PerfilUsuario.PerfilInfo"
            android:label="@string/title_activity_perfil_info"
            android:theme="@style/AppTheme.NoActionBar" />

        <activity
            android:name=".HomeActivities.PerfilUsuario.PerfilUsuario"></activity>
    </application>

</manifest>

```

5.2 Códigos fuente.

5.2.1. HomeActivity.java

```

package com.team.a404.a404team.HomeActivities;

import android.Manifest;
import android.app.AlertDialog;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.RequiresApi;
import android.support.design.widget.BottomNavigationView;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.FragmentManager;
import android.support.v7.app.AppCompatActivity;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;

import com.team.a404.a404team.R;

import static android.Manifest.permission.ACCESS_COARSE_LOCATION;

public class HomeActivity extends AppCompatActivity {

    AlertDialog alertDialog;
    private static int STORAGE_PERMISSION_CODE = 2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_home);
        getSupportActionBar().hide();

        BottomNavigationView navigation = (BottomNavigationView)

```

```

findViewById(R.id.navigation);

navigation.setNavigationItemSelectedListener(mOnNavigationItemSelectedListener);

        navigation.setSelectedItemId(R.id.navigation_map);

    }

    private BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
        = new BottomNavigationView.OnNavigationItemSelectedListener() {

        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {

            switch (item.getItemId()) {
                case R.id.navigation_news:
                    IrAnuncios();
                    return true;
                case R.id.navigation_map:

getWindow().setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_IN_SCREEN,
WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);
                    IrMaps();
                    return true;
                case R.id.navigation_profile:
                    IrPerfil();
                    return true;
            }
            return true;
        }
    };

    public void IrAnuncios() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            getWindow().getDecorView().setSystemUiVisibility(
                View.SYSTEM_UI_FLAG_LAYOUT_STABLE);
        }

getWindow().setStatusBarColor(android.graphics.Color.parseColor("#a60143"));
        FragmentManager manager = getSupportFragmentManager();
        manager.beginTransaction().replace(R.id.container, new
MascotasFragment()).commit();
    }

    public void IrMaps() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            getWindow().getDecorView().setSystemUiVisibility(
                View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
        }

getWindow().setStatusBarColor(android.graphics.Color.parseColor("#25000000"));
        FragmentManager manager = getSupportFragmentManager();
        requestStoragePermission();
        manager.beginTransaction().replace(R.id.container, new
MapaFragment()).commit();
    }

    public void IrPerfil() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            getWindow().getDecorView().setSystemUiVisibility(

```

```

        View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);

getWindow().setStatusBarColor(android.graphics.Color.parseColor("#25000000"));
    }
    FragmentManager manager = getSupportFragmentManager();
    manager.beginTransaction().replace(R.id.container, new
PerfilFragment()).commit();
    }

    public void requestStoragePermission() {

        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(this, ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.ACCESS_FINE_LOCATION)) {
                ActivityCompat.requestPermissions(HomeActivity.this, new
String[] {Manifest.permission.ACCESS_FINE_LOCATION}, STORAGE_PERMISSION_CODE);
            } else {
                ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.ACCESS_FINE_LOCATION}, STORAGE_PERMISSION_CODE);
            }
        }
        return;
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
        if (requestCode == STORAGE_PERMISSION_CODE) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                gohome();
            } else {

            }
        }
    }

    public void gohome(){
        Thread t = new Thread(){
            @RequiresApi(api = Build.VERSION_CODES.JELLY_BEAN)
            @Override
            public void run() {
                try {
                    sleep(500);
                } catch (Exception e){

                } finally {
                    IrMaps();
                }
            }
        };
        t.start();
    }
}

```

5.2.2. MapaFragment.java

```

package com.team.a404.a404team.HomeActivities;

import android.Manifest;
import android.app.Dialog;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.ColorDrawable;
import android.location.Location;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.FloatingActionButton;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.FrameLayout;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapView;
import com.google.android.gms.maps.MapsInitializer;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.UiSettings;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.mikhaellopez.circularimageview.CircularImageView;
import com.team.a404.a404team.Datos.AnuncioInformation;
import com.team.a404.a404team.Datos.DB_Datos_Mascotas;
import com.team.a404.a404team.Datos.DB_Datos_Perfil;
import com.team.a404.a404team.Datos.Marcadores_perdidos;
import com.team.a404.a404team.R;

import java.util.ArrayList;

import static android.Manifest.permission.ACCESS_COARSE_LOCATION;

public class MapaFragment extends Fragment implements OnMapReadyCallback {

    private GoogleMap mGoogleMap;
    private MapView mMapView;
    private View mView;
    private LinearLayout v_vista_error_carga;
    private double longitud, latitud;
    private LatLng actual;

```

```

        private boolean contador;
        private TextView v_mascota_nombre, v_mascota_raza, v_mascota_rasgos,
v_usuario_owner;
        private CircularImageView v_foto_mascota;
        private Button aceptar;
        private DatabaseReference all_marcadores;
        private ArrayList<Marcadores_perdidos> marcadores = new
ArrayList<Marcadores_perdidos>();
        FloatingActionButton FAB,nuevoAnuncio;

        private String id,owner ;

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
            mView = inflater.inflate(R.layout.fragment_mapa, container, false);
            return mView;

        }

        @Override
        public void onViewCreated(View view, Bundle savedInstanceState) {
            super.onViewCreated(view, savedInstanceState);

            mMapView = (MapView) mView.findViewById(R.id.map);
            v_vista_error_carga = (LinearLayout)
mView.findViewById(R.id.vista_error_carga);

            mMapView.setVisibility(View.INVISIBLE);
            contador = true;

            if (mMapView != null) {
                mMapView.onCreate(null);
                mMapView.onResume();
                mMapView.getMapAsync(this);
            }
            userLocationFAB();
            nuevoAnuncio =
(FloatingActionButton)mView.findViewById(R.id.nuevoMarcador);
            nuevoAnuncio.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    if (mGoogleMap.getMyLocation() != null) { // Check to ensure
coordinates aren't null, probably a better way of doing this...
                        LatLng actual = new
LatLng(mGoogleMap.getMyLocation().getLatitude(),
mGoogleMap.getMyLocation().getLongitude());

                        mGoogleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(actual, 16));
                        CrearMarcador(actual);
                    }

                }
            });

        }

        @Override
        public void onMapReady(GoogleMap googleMap) {
            if (ActivityCompat.checkSelfPermission(getActivity(),
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED

```

```

    && ActivityCompat.checkSelfPermission(getActivity(), ACCESS_COARSE_LOCATION) !=
    PackageManager.PERMISSION_GRANTED) {
        v_vista_error_carga.setVisibility(View.VISIBLE);
        return;
    } else {
        mGoogleMap = googleMap;
        mMapView.setVisibility(View.VISIBLE);
        MapsInitializer.initialize(getApplicationContext());

        UiSettings uiSettings = mGoogleMap.getUiSettings();
        mGoogleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);

        MiUbucacion();

        /** /INFO DE LOS MARCADORES **/

        mGoogleMap.setOnMarkerClickListener(new
        GoogleMap.OnMarkerClickListener() {
            @Override
            public boolean onMarkerClick(Marker marker) {
                LatLng posicionClick = marker.getPosition();

                String tagAll = (String) marker.getTag();
                String[] info = tagAll.split("###");
                id = info[0];
                owner = info[1];

                final Dialog dialog = new Dialog(getApplicationContext(),
                R.style.Theme_Dialog_Translucent);
                dialog.setContentview(R.layout.dialog_anuncio);
                dialog.setCanceledOnTouchOutside(true);
                dialog.getWindow().setBackgroundDrawable(new
                ColorDrawable(android.graphics.Color.parseColor("#25000000")));
                dialog.show();

                FrameLayout v_pantalla = (FrameLayout)
                dialog.findViewById(R.id.anuncio_pantalla);
                LinearLayout v_contenido_ventana = (LinearLayout)
                dialog.findViewById(R.id.contenido_ventana);

                v_pantalla.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        //dialog.hide();
                    }
                });
                v_mascota_nombre = (TextView)
                dialog.findViewById(R.id.mascota_nombre);
                v_mascota_raza = (TextView)
                dialog.findViewById(R.id.mascota_raza);
                v_mascota_rasgos = (TextView)
                dialog.findViewById(R.id.mascota_rasgos);
                v_usuario_owner = (TextView)
                dialog.findViewById(R.id.usuario_owner);
                v_foto_mascota = (CircularImageView)
                dialog.findViewById(R.id.foto_mascota);

                DatabaseReference info_mascota =
                FirebaseDatabase.getInstance().getReference("usuarios").child(owner).child("mas
                cotas").child(id);
                info_mascota.addListenerForSingleValueEvent(new

```

```

ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        //ArrayList<DB_Datos_Mascotas> d_mascota = new
        ArrayList<DB_Datos_Mascotas>();

        DB_Datos_Mascotas d_mascota =
dataSnapshot.getValue(DB_Datos_Mascotas.class);

        //d_mascota.add(nuevo);
        v_mascota_nombre.setText(d_mascota.getNombre());
        v_mascota_raza.setText(d_mascota.getRaza());
        v_mascota_rasgos.setText(d_mascota.getRasgos());
        CogerFotoMascota();
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {}
});
DatabaseReference info_owner =
FirebaseDatabase.getInstance().getReference("usuarios").child(owner);
info_owner.addListenerForSingleValueEvent(new
ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        DB_Datos_Perfil d_perfil =
dataSnapshot.getValue(DB_Datos_Perfil.class);

        v_usuario_owner.setText(d_perfil.getNombre());
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {}
});
return false;
}
});
mGoogleMap.setMyLocationEnabled(true);
mGoogleMap.getUiSettings().setCompassEnabled(false);
mGoogleMap.getUiSettings().setMyLocationButtonEnabled(false);
}
}

// -----

private void CogerFotoMascota() {
    StorageReference stor =
FirebaseStorage.getInstance().getReference().child("images/" + owner.toString()
+ "/userphoto.jpg"); //Cambiar a foto de la mascota (Sergio)
    final long ONE_MEGABYTE = 1024 * 1024;
    stor.getBytes(ONE_MEGABYTE).addOnSuccessListener(new
OnSuccessListener<byte[]>() {
        @Override
        public void onSuccess(byte[] bytes) {
            Bitmap bmp = BitmapFactory.decodeByteArray(bytes, 0,
bytes.length);
            v_foto_mascota.setImageBitmap(bmp);
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override

```

```

        public void onFailure(@NonNull Exception exception) {
            v_foto_mascota.setImageResource(R.drawable.logo_petaware);
        }
    });
}

private void userLocationFAB() {
    FAB = (FloatingActionButton) mView.findViewById(R.id.myLocationButton);
    FAB.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            if (mGoogleMap.getMyLocation() != null) { // Check to ensure
coordinates aren't null, probably a better way of doing this...
                LatLng actual = new
LatLng(mGoogleMap.getMyLocation().getLatitude(),
mGoogleMap.getMyLocation().getLongitude());

mGoogleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(actual, 16));
            }

        });
}

private void CrearMarcador(LatLng actual) {

}

/**
 * CREAR MARCADORES EN EL MAPA
 */

public void CogerMarcadores() {

    all_marcadores =
FirebaseDatabase.getInstance().getReference("marcadores").child("perdidas");
    all_marcadores.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            marcadores = new ArrayList<>();

            for (DataSnapshot dato : dataSnapshot.getChildren()) {
                Marcadores_perdidos mp =
dato.getValue(Marcadores_perdidos.class);
                marcadores.add(mp);
                Log.v("datosUsuarios",mp.toString());
            }
            MeterMarcadores();
        }

        public void onCancelled(DatabaseError databaseError) {

        }
    });
}

public void MeterMarcadores() {
    mGoogleMap.clear();
    Log.v("ESTO_2", "" + marcadores.size());
    for (int i = 0; i < marcadores.size(); i++) {
        LatLng ubi = new LatLng(marcadores.get(i).getLatitud(),
marcadores.get(i).getLongitud());

```



```

        Marker map_marcador = mGoogleMap.addMarker(new MarkerOptions()
            .position(ubi)

        .icon(BitmapDescriptorFactory.fromResource(R.mipmap.ic_logomaps)));

        map_marcador.setTag(marcadores.get(i).getId_mascota() + "##" +
marcadores.get(i).getOwner());
        Log.v("ESTO_3", "" +ubi);
        Log.v("ESTO_3", "" +marcadores.get(i).getId_mascota() + "##" +
marcadores.get(i).getOwner());
    }

}

/**
 * METODO PARA MOVER EL MAPA A TU UBICACION
 */

public void MiUbucacion() {

    mGoogleMap.setOnMyLocationChangeListener(new
GoogleMap.OnMyLocationChangeListener() {
        @Override
        public void onMyLocationChange(Location location) {
            longitud = location.getLongitude();
            latitud = location.getLatitude();
            Log.v("lat", String.valueOf(latitud));
            //setLatLng(location.getLatitude(),location.getLongitude());
            LatLng actual = new LatLng(latitud, longitud);

            if (contador) {
                CogerMarcadores();

mGoogleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(actual, 15));
                contador = false;
            }

            // mMap.setInfoWindowAdapter();
        }
    });
}

}

```