



UNIVERSITAT
POLITÀCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una plataforma social Android para propietarios de mascotas

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Iván Espadas García

Tutor: Pedro Valderas Aranda

2016-2017

Resumen

El presente proyecto consiste en el desarrollo y despliegue de una aplicación móvil en el sistema operativo Android que permite a los dueños de mascotas disponer de una red social en la que tener a sus mascotas como protagonistas.

La aplicación permite registrar a la mascota identificándose con el nombre y una foto de ella. Se puede interactuar con otras mascotas, teniéndolas como amigos.

También dispone de una opción de Cruces por la cual se puede publicar y buscar posibles mascotas con las que poder cruzar nuestra mascota.

Otro servicio es el de Perdidos, si desgraciadamente se pierde nuestra mascota podremos publicarlo, usando la ubicación y si alguien lo ve nos lo podrá comunicar.

La aplicación está desarrollada nativamente en Android utilizando el IDE oficial Android Studio. El lenguaje de programación es Java. Se ha utilizado el servicio de Firebase de Google para el registro y la autenticación en la aplicación, además de para guardar todas las imágenes de los usuarios. También se ha creado una base de datos MySQL para guardar toda la información relativa a los usuarios, accediendo a ella a través de Scripts en PHP.

Palabras Clave: Mascota, gato, perro, Android, Java, PHP, MySQL,JSON, phpMyAdmin, Firebase,Google Maps.

Resum

El present projecte consisteix en el desenvolupament i desplegament d'una aplicació mòbil en el sistema operatiu Android que permeteix als amos de les mascotes disposar d'una xarxa social en la qual tindre a les seus mascotes com a protagonistes.

L'aplicació permeteix registrar a la mascota identificantse amb el nom i una imatge d'ella. Es pot interactuar amb altres mascotes ,tenintles com amigues.

També disposa d'una opció d'encreuaments per la qual es pot publicar i buscar possibles mascotes amb les que es pot creuar la nostra mascota.

Altre servici es el de perduts , si desgraciadament es perd la nostra mascota podrem publicar-ho , usant la ubicació i si algú la veu ens ho podrà comunicar.

L'aplicació està desenvolupada nativament en Android utilitzant el IDE oficial Android Studio. El llenguatge de programació es Java. S'ha utilitzat el servici de Firebase de Google per al registre i la autenticació en la aplicació ,a mes de per a guardar totes les imatges del usuaris.També s'ha creat una base de dades MySQL per a guardar tota la informació relativa al usuaris, accedint a ella mitjançant scripts en PHP.

Paraules Clau: Mascota, gat, gos, Android, Java, PHP, MySQL,JSON, phpMyAdmin, Firebase,Google Maps.



Abstract

The present project consists on the development and deployment of a mobile application in the Android operating system that allows pet owners to have a social network in which to have their pets as protagonists.

The application allows registering the pet identifying with the name and a photo of it. You can interact with other pets, having them as friends.

It also has an option of Crosses by which you can publish and search for possible pets with which to cross our pet.

Another service is the Lost, if unfortunately our pet is lost we can publish it, using the location and if someone sees it we can communicate.

The application is natively developed on Android using the official Android Studio IDE. The programming language is Java. The Google Firebase service has been used for registration and authentication in the application, as well as for saving all user images. A MySQL database has also been created to store all the information relative to the users, accessing it through scripts in PHP.

Keywords: Pet, Cat, Dog, Android, Java, PHP, MySQL,JSON, phpMyAdmin, Firebase,Google Maps.



TABLA DE CONTENIDOS

1. Introducción.....	7
2. Estado del arte.....	9
3. Contexto Tecnológico.....	13
4. Arquitectura	17
5. Metodología.....	20
6. Análisis de Necesidades	22
7. Diseño.....	30
8. Implementación	35
9. Conclusiones	49
Apéndice A. Manual del Usuario.....	50
Apéndice B. Bocetos	64
Bibliografía	73



1. INTRODUCCIÓN

1.1. Presentación y Contexto

Lo que se presenta en este Proyecto es una aplicación móvil programada en Android consistente en una Red Social de mascotas.

El contexto actual del auge de internet y los móviles ha impulsado el crecimiento exponencial de las redes sociales. Hoy en día estamos conectados a internet en cualquier momento, compartimos nuestros viajes, cenas, excursiones... todos los momentos que vivimos a lo largo del día, así como las fotos de todo esto.

Nuestras mascotas están también en alguno de esos momentos, por ello necesitamos una red social donde ellas sean las protagonistas.

Así nace MascotaSocial, una red social donde nuestras mascotas son el centro de la red social, las protagonistas de toda la información que hay.

1.2. Objetivos

MascotaSocial nace con la intención de ser una red social única y exclusivamente centrada en nuestras mascotas, con lo que todo lo que veremos en ella es referente a nuestros animales de compañía.

Los objetivos que se pretenden cumplir con el proyecto son:

Generales

- Demostrar la capacidad del alumno para desarrollar un proyecto completo de forma parcialmente autónoma, partiendo de los conocimientos adquiridos en la titulación y de la capacidad de aprendizaje por cuenta propia.
- Capacidad para poder buscar en documentación oficial en de una tecnología nueva.
- Desarrollar una aplicación móvil nativamente en Android.

Específicos

- Ofrecer la capacidad de gestionar cuentas, historias, imágenes, solicitudes y mensajes. Entiéndase como gestionar la capacidad para obtener, crear, modificar, eliminar y/o asociar elementos, según corresponda. Véase el ámbito y los casos de uso.
- Desplegar una base de datos a la que acceder mediante scripts de PHP.
- Manejar esta información bidireccionalmente entre aplicación y base de datos en formato JSON
- Seguir unas líneas de diseño modernas, en este caso Material Design, que den como resultado una interfaz usable e intuitiva.



1.3. Estructura del resto de la memoria

La memoria comienza con la presentación de que consiste el proyecto, así como el contexto actual y los objetivos que se desean cumplir.

Seguidamente se analizan las diferentes opciones que hay en el mercado de aplicaciones similares.

En los dos siguientes capítulos se aborda las tecnologías disponibles actualmente y cuales se han usado. Así como la representación de toda la arquitectura que hay montada para que sea posible usar la aplicación.

Lo siguiente que se especifica es la metodología abordada para crear el proyecto, en lo que se ve el Diseño Centrado en el Usuario, que detallaremos en dicho apartado

A continuación, se ve el análisis de necesidades, proporcionado por el estudio que se debe realizar siguiendo, como acabamos de mencionar el **DCU** (Diseño centrado en el Usuario).

Siguiendo con la memoria, tenemos los dos siguientes capítulos.

En uno veremos el diseño que se ha realizado de la aplicación visualmente y estructuralmente, así como el diseño de la estructura de datos que se maneja.

En el otro ya veremos la implementación, como está hecha la aplicación, y como se han utilizado las tecnologías que se han seleccionado.

Por último, se analizan las conclusiones derivadas de la creación y desarrollo del proyecto.

La estructuración de la memoria sigue un orden tanto cronológico como causal, es decir, cada capítulo depende y parte del anterior. Por lo tanto, permite realizar un seguimiento gradual de la evolución del proyecto.

2. ESTADO DEL ARTE

Antes de comenzar con la definición del proyecto veremos algunas aplicaciones existentes similares a la aplicación que hemos abordado.

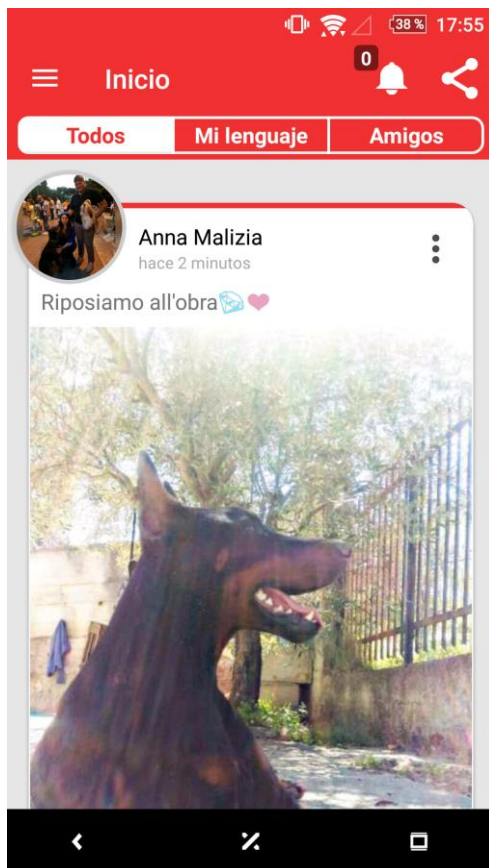


Empezamos por esta aplicación llamada **Dogalize**, que como en el nombre se intuye, tiene la peculiaridad de ser diseñada solo para perros.

En esta aplicación nos podemos encontrar con las opciones de una red social, tales como buscar, añadir amigo y chat.

Además, dispone de un servicio para adoptar perros, para encontrar perros perdidos y para citas de ellos.

Por último, también incluye la opción de poder llamar a un veterinario o un entrenador.



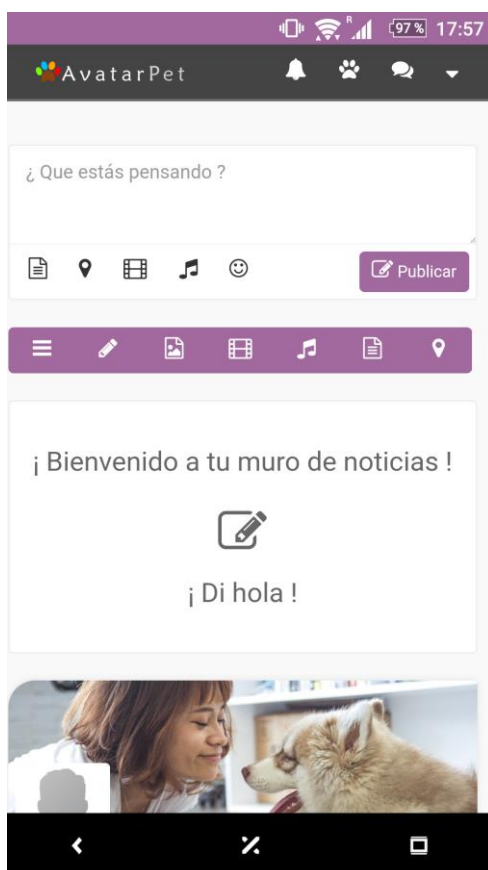


En **AvatarPet** nos encontramos ante una red social al estilo Facebook, pero para mascotas.

En ella podremos registrar a nuestra mascota y podremos compartir contenido a través del muro.

Podremos tener amigos y conversar con ellos.

Además, encontraremos una sección donde podremos registrar y añadir a una mascota que se haya perdido.

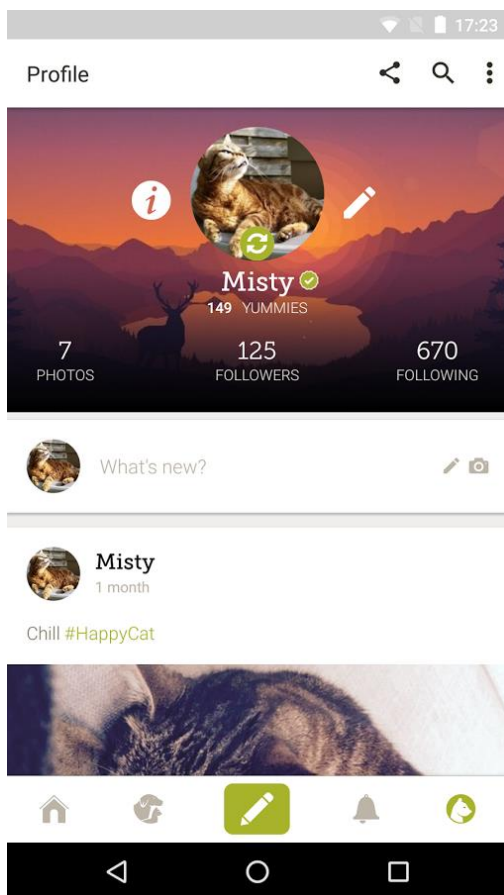




La aplicación de **YummyPets** nos presenta una aplicación muy simple centrada sobre todo en las fotos.

Está la posibilidad de compartir un texto o imagen con nuestros seguidores al igual que nosotros vemos el contenido de los que estamos siguiendo.

Existe también la opción de poder chatear con nuestros seguidores de forma privada.



Por tanto, teniendo en cuenta las plataformas actuales en el mercado, podemos analizar los puntos fuertes y débiles de cada uno, y así poder enfocar Mascota Social para explotar los puntos fuertes y suplir las carencias.

Nombre	Tipo Animal	Amigos	Mensajería	Cruces	Perdidos
Dogalize	Perros	SI	SI	SI	SI
AvatarPet	Varios	SI	SI	NO	SI
YummyPets	Varios	SI	SI	NO	NO
MI APP	Varios	SI	SI	SI	SI



3. CONTEXTO TECNOLÓGICO

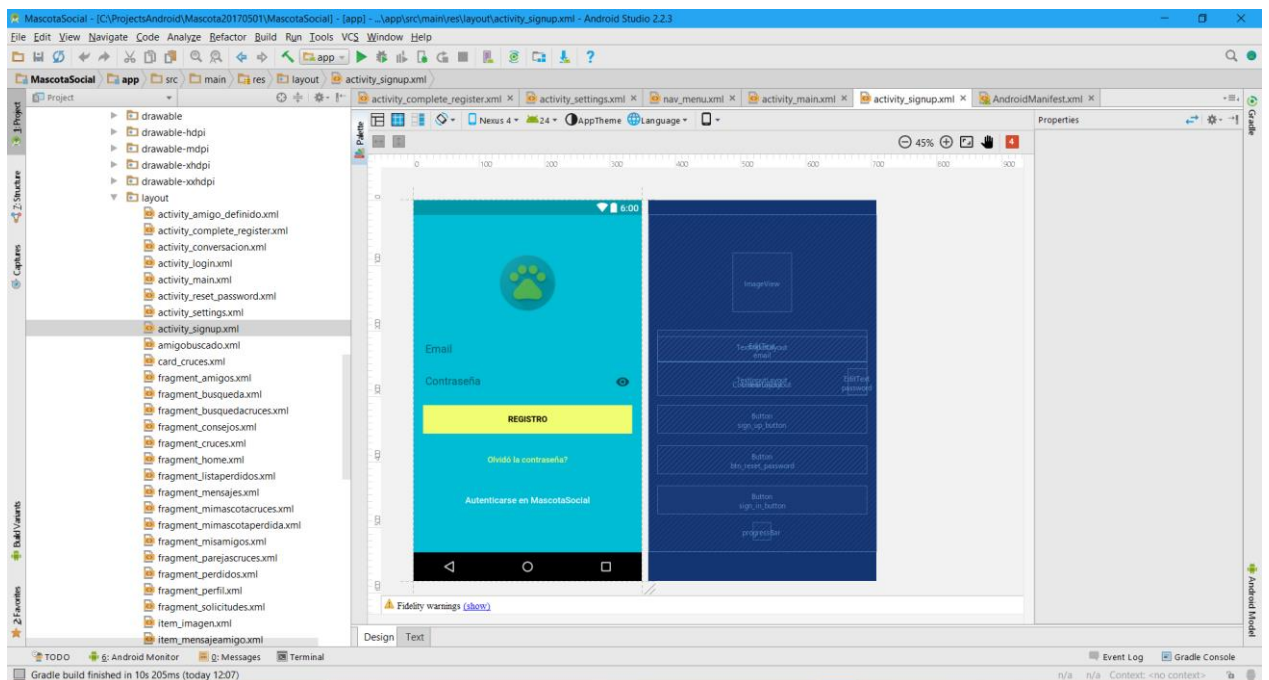


Para el desarrollo nativo de Android se ha usado el IDE oficial **Android Studio**, ya que es el que más ventajas y comodidades ofrezca para el desarrollo en esta plataforma.

Presentado en la conferencia de Google I/O de 2013 , fue lanzado inicialmente en 2014 y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android.

Es gratuito y está disponible para Windows, macOS y Linux.

Se ha utilizado **Android Studio 2.2.3**.





El lenguaje utilizado para programar la aplicación es **Java**.

Apareció en 1995 desarrollado por Sun Microsystems.

Java es un lenguaje de programación orientado a objetos diseñado para poder correr en cualquier maquina sin necesidad de ser recompilado y es uno de los lenguajes más utilizados en la actualidad.



La base de datos utilizada para almacenar la información de la aplicación es **MySQL**.

MySQL es un sistema de gestión de bases de datos relacional creado en 1995 por Oracle.

Es una de las bases de datos más utilizadas en la actualidad, sobre todo en el entorno web.



SQL es el lenguaje de consulta estructurado y declarativo, de acceso a bases de datos relacionados que se ha utilizado para acceder y manipular la información.

Este lenguaje creado en 1974 es hoy en día un estándar y es el lenguaje mas usado para operar con bases de datos.



El lenguaje utilizado para transmitir bidireccionalmente la información entre la base de datos y la aplicación es **PHP**.

PHP es un lenguaje de programación de uso del lado del servidor lanzado en 1995.

Es uno de los lenguajes del lado del servidor más utilizados en estos momentos.



phpMyAdmin ha sido la herramienta web utilizada para administrar la base de datos MySQL.

Esta herramienta creada en 1998 es de las más populares siendo su uso gratuito.



Para la autenticación de los usuarios de la aplicación y para guardar las imágenes se ha utilizado la plataforma **Firebase**.

Firebase es una plataforma de desarrollo móvil en la nube creada por Google en 2011.

Ofrece, entre otras cosas, el servicio de crear y autenticar las cuentas de la aplicación además de almacenamiento para las imágenes.





El servicio de Mapas utilizado es **Google Maps**.

Publicado en el 2005 , los mapas de Google son los más completos y más utilizados en la actualidad.



El formato en el que se envían los datos bidireccionalmente desde la aplicación al php para atacar la base de datos y viceversa es **JSON**.

Apareció en 2005 y se está posicionando como un sustituto para el transporte de datos de XML.



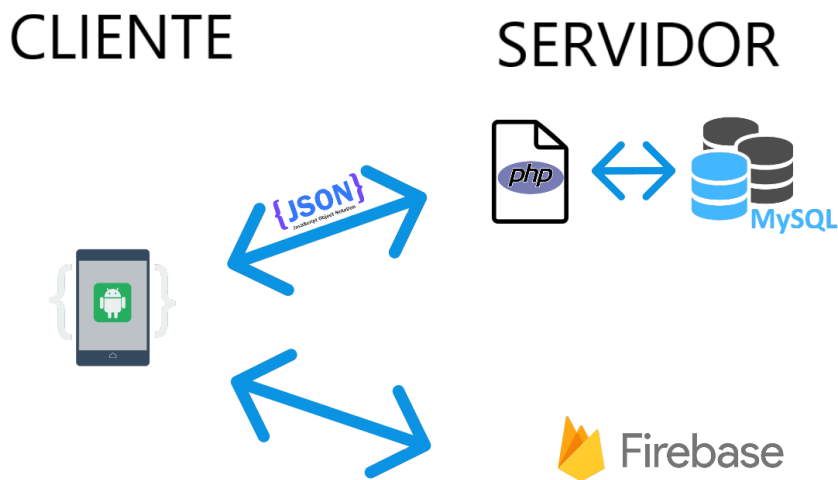
Filezilla ha servido para la transmisión de los archivos entre el ordenador de desarrollo y el servidor de la aplicación.

Lanzada en 2001, Filezilla es una aplicación gratuita y de código abierto.

4. ARQUITECTURA

Ahora se presentará la arquitectura formada para hacer posible este proyecto.

Tenemos una arquitectura **cliente-servidor**. Por un lado, el cliente es la aplicación Android y por otro, el servidor se corresponde con la base de datos MySQL, además del servicio de Firebase, que nos ofrecería la parte de la persistencia.



Cliente

El cliente está formado por la propia aplicación de Mascota Social en Android

Base de datos

Para acceder a toda la información que almacenamos respecto a los usuarios, sus cuentas y demás información tenemos una **base de datos** en MySQL alojada gratuitamente en Hostinger.



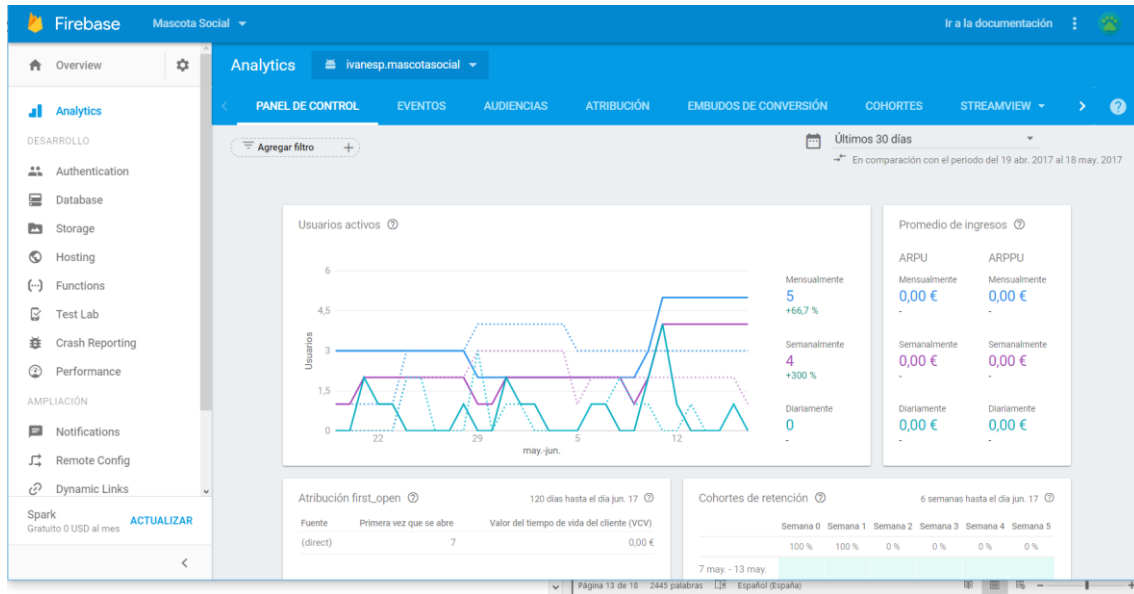
Para comunicarnos con la base de datos y poder consultar, insertar, actualizar y borrar los datos tenemos un **hosting** en el que se almacenan *scripts* en PHP con los que nos comunicamos bidireccionalmente en formato **JSON**.



Firestore

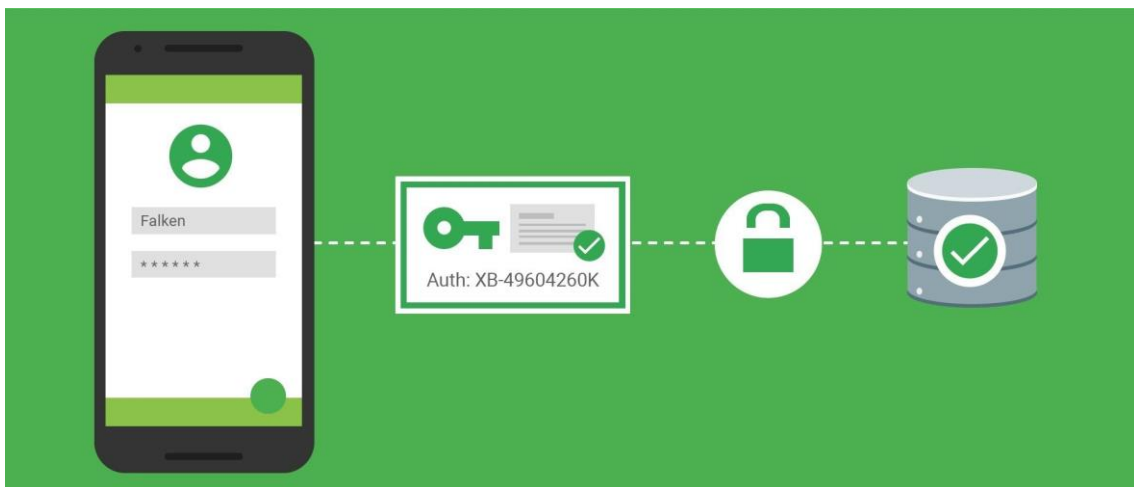
Firestore es una plataforma de desarrollo para aplicaciones multiplataforma apoyada por Google la cual nos ofrece múltiples opciones y servicios.

Para utilizarlo se necesita tener una cuenta de Google y crear un nuevo Proyecto:



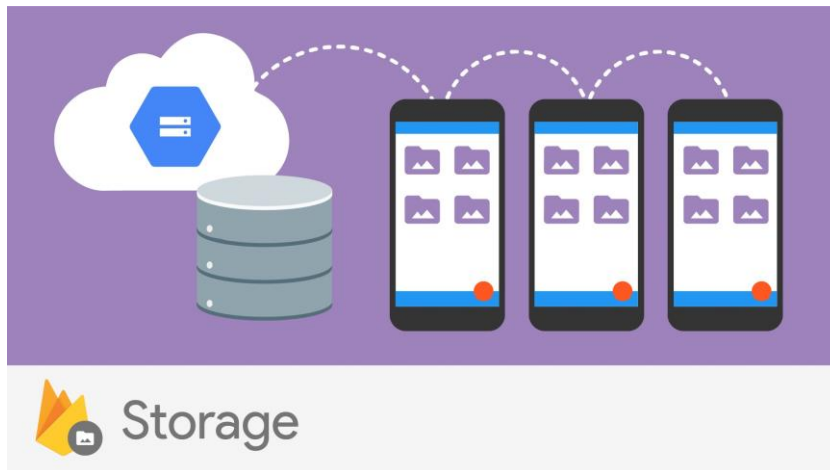
En este caso interesa mencionar:

- **Autenticación** de forma segura:



Con ello tendremos un sistema de registro y autenticado totalmente seguro que se gestiona y almacena en los servidores de Google

- **Almacenamiento de imágenes:**



Se utilizará el almacenamiento de Firebase ya que nos proporciona un servicio rápido y seguro de almacenar y recuperar las imágenes proporcionadas por los usuarios

Otros Servicios

Para el servicio de mapas usamos la API de **Google Maps** en su última versión.

Para utilizarlo es necesario crear una clave API e introducir en el archivo AndroidManifest.xml del proyecto

```
manifest application meta-data
    android:name=".SignupActivity"
    android:screenOrientation="portrait" />
<activity
    android:name=".LoginActivity"
    android:screenOrientation="portrait" />
<activity
    android:name=".CompleteRegisterActivity"
    android:screenOrientation="portrait" />
<activity
    android:name=".SettingsActivity"
    android:label="Ajustes"
    android:parentActivityName=".MainActivity" />
<activity android:name=".ResetPasswordActivity" />
<activity
    android:name=".amigodefinidoActivity"
    android:parentActivityName=".MainActivity"
    android:screenOrientation="portrait"/>
<activity android:name=".ConversacionActivity"
    android:parentActivityName=".MainActivity"
    android:screenOrientation="portrait" />
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyD2K5aay3-A10gJ8gYK1bJAV9wg" />
</application>
</manifest>
```

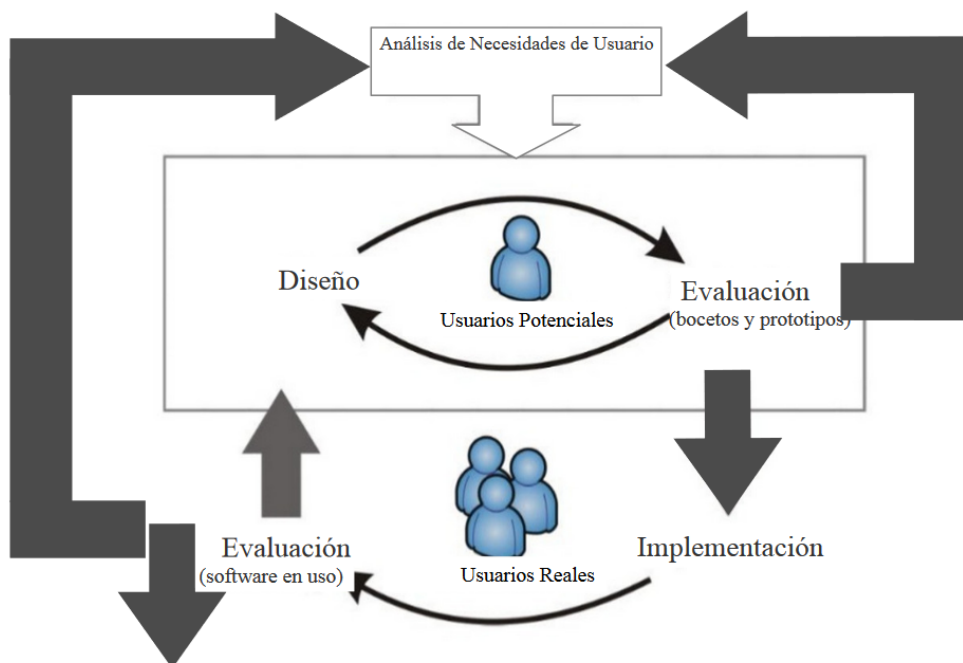


5. METODOLOGÍA

La metodología utilizada para el desarrollo de este proyecto ha sido el **Diseño Centrado en el Usuario(DCU)**.

Esta metodología se basa en la idea de situar al usuario en el centro del Diseño, basándose este en sus necesidades, gustos y preferencias para así poder obtener un producto con que el usuario esté cómodo y sea lo que se espera.

Para ello un concepto fundamental es la **usabilidad**. Está relacionado con la facilidad de uso, la rapidez con la que se puede aprender a usar y la eficiencia del producto.



El proceso del DCU tiene 3 fases:

- Análisis de necesidades de Usuario

Consiste en identificar los aspectos relacionados con la creación del proyecto, tales como identificar a las personas a las que se dirige el producto, para que lo usaran y en qué condiciones, con el fin de determinar los objetivos del usuario que deberán satisfacerse.

Existen múltiples técnicas para llevar a cabo esto, tales como entrevistas, cuestionarios, brainstorming, etc

- Diseño

Es la fase en la que, con todos los resultados obtenidos tras realizar la investigación de las preferencias de los usuarios, empezamos a definir el diseño. Hay varias técnicas de poder llevar a cabo esta fase, que según nuestras necesidades y objetivos podremos elegir alguna, como: sketching, storyboards, maquetas digitales, prototipos software, etc

- Evaluación

Una vez llevado a cabo el análisis de las necesidades y haber definido el diseño se deberá poner a cabo en esta fase, la de evaluación.

Se enumeran todos los problemas de usabilidad que se detectan para su corrección hasta que no quede ninguno y se dará por finalizada la fase con la validación del diseño.



6. ANÁLISIS DE NECESIDADES

6.1. Encuestas

A continuación, se expone como se ha hecho y llevado a cabo el estudio de los usuarios potenciales de la aplicación y así conocer sus necesidades, gustos y preferencias para poder basar en ello el desarrollo de la aplicación.

Para ello se hizo una investigación cualitativa, en forma de encuesta, usando los cuestionarios de **Google Forms**, a 10 personas.

Las 10 preguntas que se hacían eran:

1. Sexo
2. Edad
3. Uso de redes sociales
4. Uso de aplicaciones similares
5. Mascota
6. ¿Te gustaría poder compartir fotos?
7. ¿Te gustaría chatear con amigos?
8. ¿Te gustaría tener publicidad de productos para mascotas?
9. ¿Te gustaría tener información acerca de eventos de animales?
10. ¿Te gustaría tener opción de cruzar tu mascota?
11. ¿Te gustaría poder conectar la aplicación a otras redes sociales?
12. ¿Te gustaría el servicio de mascotas perdidas?

Al ser una encuesta online pude ser enviada por correo electrónico a cualquier persona.

Los resultados que se obtuvieron fueron:

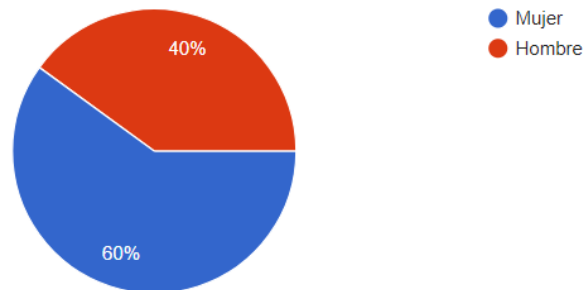


Pregunta 1:

La mayoría de los encuestados fueron mujeres, exactamente un 60%.

Sexo

15 respuestas

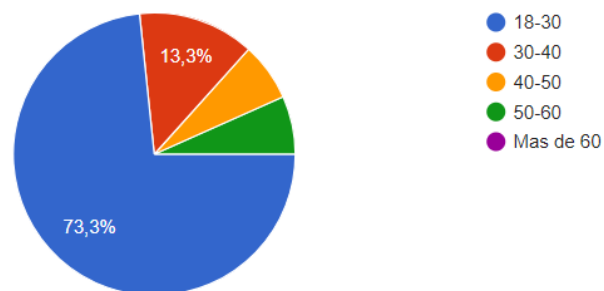


Pregunta 2:

En la edad vemos diferentes tramos de edad encuestados, aunque el mayoritario es el de 18-30 con más de dos tercios, un 73,3%

Edad

15 respuestas

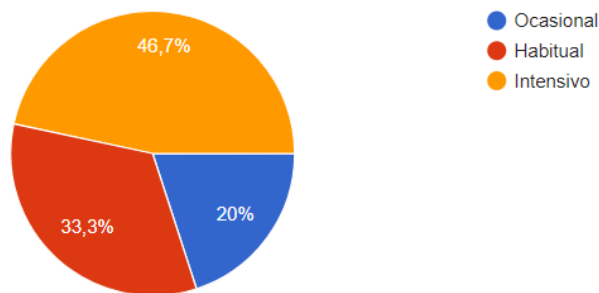


Pregunta 3:

En la pregunta acerca del uso de las redes sociales se puede observar que hay más igualdad entre las respuestas, aunque la ganadora, por no mucho, es el uso intensivo de las redes sociales con un 46,7%

Uso de redes Sociales

15 respuestas

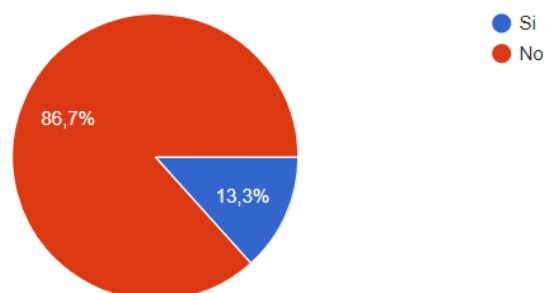


Pregunta 4:

En la cuestión acerca del uso sobre aplicaciones similares a la de este proyecto, se puede observar que prácticamente no hay uso de ellas. 86,7% de uso frente al 13,3% de no uso.

Uso de aplicaciones similares

15 respuestas

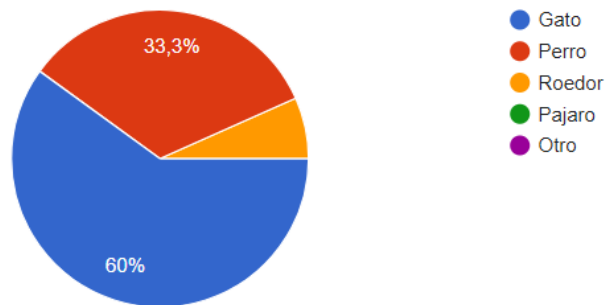


Pregunta 5:

En esta pregunta podemos observar el tipo de público entrevistado con una amplia mayoría que tiene gato, un 60%.

Mascota

15 respuestas

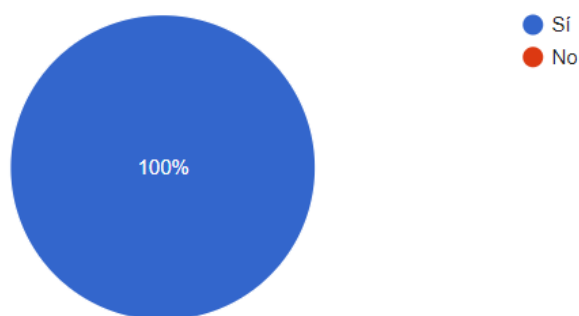


Pregunta 6:

En la pregunta acerca de la opción de poder compartir fotos vemos unanimidad hacia el sí con un 100%.

¿Te gustaria poder compartir fotos?

15 respuestas

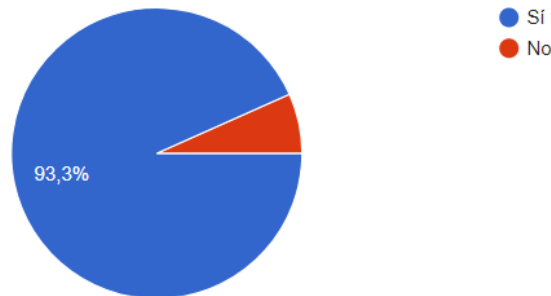


Pregunta 7:

En la opción de poder chatear con amigos solo hay un voto de disconformidad, con un 93,3% a favor de tener esta característica.

¿Te gustaria chatear con amigos?

15 respuestas

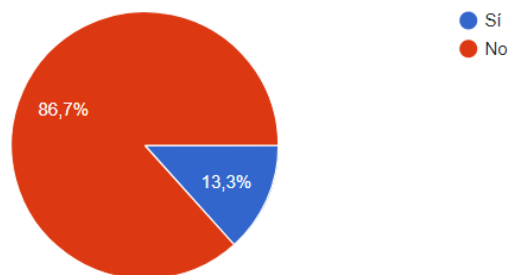


Pregunta 8:

Preguntando acerca de si incluir publicidad de productos necesarios para nuestros animales, hay una respuesta mayoritaria al no, un 86,7%.

¿Te gustaria tener publicidad de productos para mascotas?

15 respuestas

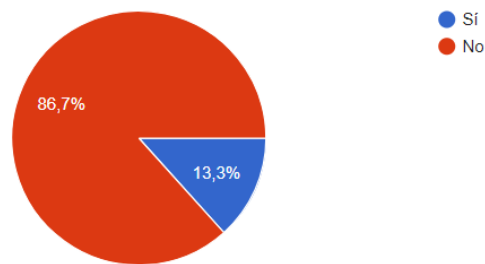


Pregunta 9:

Sobre la opción de poder tener información acerca de eventos sobre animales, vemos que no hay mucho interés, con un 86,7% de respuesta negativa.

¿Te gustaria tener información acerca de eventos de animales?

15 respuestas

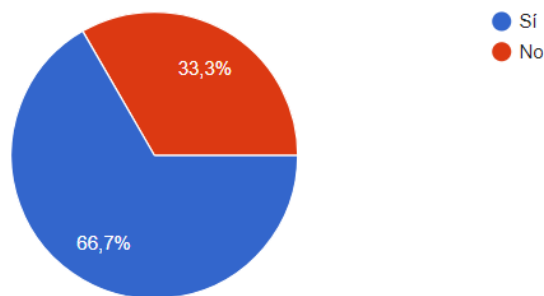


Pregunta 10:

En esta pregunta de si se quiere un servicio para poder encontrar una pareja de cruce para nuestra mascota, encontramos una respuesta positiva con dos tercios , un 66,7%

¿Te gustaria tener opción de cruzar tu mascota?

15 respuestas

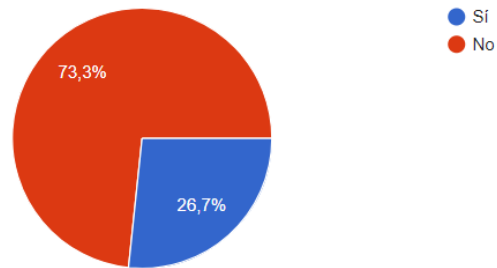


Pregunta 11:

Preguntando si sería interesante poder conectar la aplicación con otras redes sociales ya existentes, hay una mayoría del no, con un 73,3%.

¿Te gustaría poder conectar la aplicación a otras redes sociales?

15 respuestas

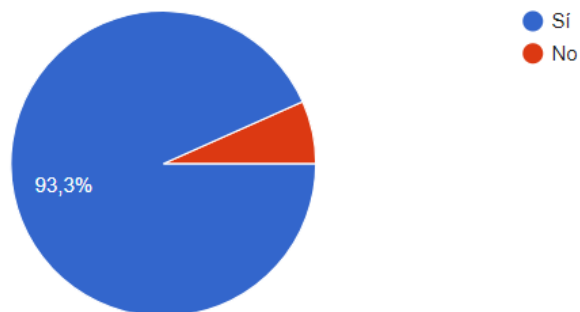


Pregunta 12:

Por último, en la pregunta de si queremos una opción para poder registrar y buscar mascotas perdidas vemos una respuesta casi total del sí, con un 93,3%.

¿Te gustaría el servicio de mascotas perdidas?

15 respuestas



6.2. Persona

Con todos estos datos anteriores es posible realizar un análisis y extraer como resultado una persona tipo.

PERSONA

Nombre: Irina Gómez

Biografía:

- 24 años de edad
- Vive en Valencia
- Vive en un piso de estudiantes con 2 compañeras mas
- Tiene carnet de conducir y un coche
- Le encanta la naturaleza
- Le encantan los animales, en especial los gatos.
- Colabora en alguna protectora de animales
- Tiene varios gatos



Salud:

- Mide 1,75m
- No tiene problemas de salud
- No esta operada de nada
- Cuida la alimentación y le gusta todas las gastronomías
- No fuma, bebe esporádicamente
- Le gusta hacer algo de deporte

Objetivos:

- Dar el máximo en su trabajo para ser una buena profesional
- Le gusta viajar y descubrir culturas nuevas
- Ser feliz con su familia y amigos
- Ver un mundo mejor

Tecnología:

- Dispone de ordenador y móvil
- Le gusta lo último en móviles ya que lo usa mucho
- Le gusta la fotografía y capturar sitios bonitos
- Le gusta compartir fotos de sus animales.



7. DISEÑO

La imagen que acompaña en todo momento la aplicación y la que es el logo:



Pienso que representa intuitivamente que se trata de una aplicación relacionada con los animales.

Para realizar el diseño preliminar de la interfaz de la aplicación he utilizado el programa para hacer *mockups* **Balsamiq Mockups 3**.

Este programa nos permite crear *mockups* de manera muy sencilla, teniendo gran cantidad de controles e iconos.

Pese a ser completa, no disponía de los controles *Material Design* que se han utilizado en la aplicación, por lo utilicé un complemento para ello, llamado **Android Material Design Symbols**.

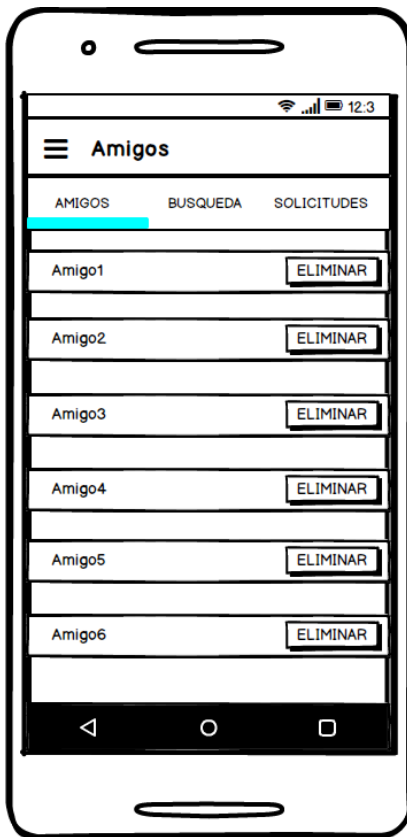
Como he comentado antes, para el diseño he optado por utilizar *Material Design* de Google.

Es un lenguaje de diseño pensado para ser utilizado en cualquier plataforma, pudiendo interoperar entre ellas.

Es la gran apuesta de Google en el diseño de todas sus aplicaciones, ya que es un diseño sencillo, moderno y adaptable a cualquier tamaño de pantalla, formato y forma de utilizarlo, ya se táctil o con ratón y teclado.

Como ejemplo se presentan 3 *mockups* el resto estarán en el Apéndice B dedicado a ello.

Amigos



Esta es la pantalla que vemos al navegar a la sección de amigos.

Tenemos una lista con el nombre de todos ellos y la opción de poder eliminar nuestra amistad.

Cruces



Esta pantalla nos servirá para poder gestionar los cruces.

Podemos observar que tenemos la foto de nuestra mascota junto al tipo, raza, nombre y sexo.

Deslizando el control a activado podremos registrarnos para encontrar cruce insertando un texto de descripción de nuestra mascota.

Perdidos



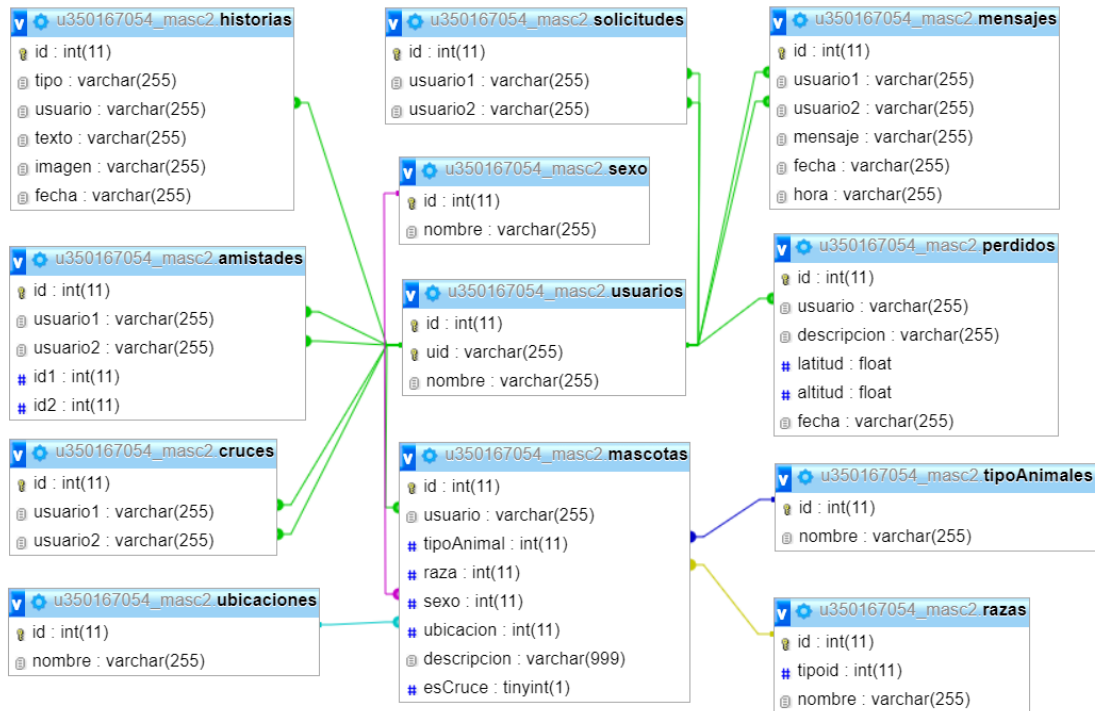
Esta pantalla nos servirá para poder gestionar el anuncio sobre nuestra mascota perdida.

Deslizando el control a activado podremos introducir un texto de información sobre la pérdida de nuestra mascota, así como la ubicación de su pérdida.



Diagrama Base De Datos

A continuación, se muestra el diagrama de la base de datos:



Hay 12 Tablas en total:

- **Usuarios**
La tabla principal de la aplicación donde se almacenan los usuarios y sus datos.
- **Mascotas**
Es la tabla donde se mantiene la información de la mascota de cada usuario.
- **Tipo de Animales**
Tabla donde se almacenan los tipos de Animales.
- **Razas**
Contiene los tipos de razas según el tipo de animal que sea.
- **Sexo**
Almacena los tipos de sexo.
- **Ubicaciones**
Listado de las distintas ubicaciones, en este caso provincias.



- Historias
Tabla donde se almacena las distintas historias de cada usuario.
- Amistades
Tabla que contiene las relaciones de amistad entre los usuarios.
- Cruces
Mantiene los datos acerca de los cruces favoritos de los usuarios.
- Solicitudes
Tabla que contiene las peticiones de amistad entre dos usuarios.
- Mensajes
Contiene los mensajes privados que mantienen dos usuarios.
- Perdidos
Almacena los datos de las mascotas actualmente perdidas.



8. IMPLEMENTACIÓN

La aplicación está compilada con el último SDK (*Software Development Kit*) de Android. La versión 24 correspondiente a la versión de **Android 7 Nougat**

Se ha establecido de versión mínima para poder ejecutar la aplicación el SDK 21 correspondiente a la versión de **Android 5 Lollipop**.

Esto permite utilizar muchas de las últimas tecnologías lanzadas por Google sin problema.

Para probar la aplicación he utilizado la emulación que nos ofrece Android Studio, utilizando un **Nexus 6P** con Nougat como móvil virtualizado.

Además, la aplicación también está probada en móviles físicos como el **OnePlus 3T** con la versión Nougat y un **Sony Xperia Z3** con la versión Marshmallow.

Se ha obtenido una buena experiencia en los 3 dispositivos teniendo diferentes tamaños de pantalla y resolución.

Aparte de las dependencias propias de los servicios de Google Play se han utilizado dos más:

- CircleImageView: Para poder tener imágenes circulares
- MaterialEditText: Estilo del control de introducción con diseño *Material*
- Picasso: Gestión más eficiente y rápida de descargar y mostrar las imágenes.

En Android cada pantalla está formada por dos archivos, para separar:

- Diseño:

Contiene todos los elementos que forman la parte visible de la pantalla.
Es un archivo **xml**.

- Lógica:

Es el que contiene toda la lógica del funcionamiento de la pantalla, cada elemento que tiene que mostrar, que tiene que hacer, etc.
Es un archivo **java**.



Como ejemplo se muestra la pantalla de Ajustes.

El archivo xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        android:theme="@style/ThemeOverlay.AppCompat.Dark"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
    />
    <ScrollView
        android:layout_below="@id/toolbar"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:isScrollContainer="true">
        <RelativeLayout
            xmlns:android="http://schemas.android.com/apk/res/android"
            xmlns:app="http://schemas.android.com/apk/res-auto"
            android:id="@+id/layoutSettings"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_below="@id/toolbar"
            android:layout_margin="12dp">
            <TextView
                android:id="@+id/txtImagenPerfil"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Foto Perfil"
                android:textSize="22dp"
                android:gravity="left"
                android:layout_marginBottom="5dp"
            />
            <ImageView
                android:id="@+id/imagenPerfil"
                android:layout_marginStart="5dp"
                android:layout_width="250dp"
                android:layout_height="250dp"
                android:layout_gravity="center"
                android:layout_marginTop="10dp"
                android:layout_below="@id/txtImagenPerfil"
                android:layout_centerHorizontal="true"/>

            <Button
                android:id="@+id/cambiarImagen"
```



```

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="6dp"
        android:text="Cambiar Imagen"
        android:drawableRight="@drawable/ic_insert_photo"
        android:drawablePadding="10dp"
        android:layout_below="@id/imagenPerfil"
        android:background="#64B5F6"/>
    <com.rengwuxian.materialledittext.MaterialEditText
        android:id="@+id/nombreUsuario"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:inputType="textPersonName"
        android:layout_below="@id/cambiarImagen"/>

    <Spinner android:id="@+id/spinTipoAnimales"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/nombreUsuario"/>
    <Spinner android:id="@+id/spinRaza"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/spinTipoAnimales"
        android:layout_marginBottom="30dp"/>

    </RelativeLayout>
</ScrollView>
<Button
    android:id="@+id/guardarCambios"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dip"
    android:text="Guardar Cambios"
    android:textColor="@android:color/black"
    android:background="@color/bg_login"
    android:layout_alignParentBottom="true"
    />
</RelativeLayout>

```

Como se puede observar cada elemento de la pantalla es un elemento del xml, y el diseño utilizado, *Relative Layout* contiene a todos los elementos.

Se ha utilizado este tipo de diseño, ya que es el más flexible para la colocación de los elementos. Cada elemento se posiciona en una posición relativo a otro elemento, sea el elemento contenedor o cualquiera de sus homólogos.



El archivo java:

```
package ivanesp.mascotasocial;

import android.content.Intent;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.Toast;

/**** MUCHOS MAS IMPORTS ****/

public class SettingsActivity extends AppCompatActivity {
    Toolbar toolbar;
    ActionBar actionBar;
    String uidactual;
    FirebaseStorage storage = FirebaseStorage.getInstance();
    ImageView imagenPerfil;
    private static final int SELECT_PICTURE = 100;
    Uri selectedImageUri;
    private String URL_TIPOANIMALES =
"http://tfgivan2017.esy.es/obtenerTipoAnimales.php";
    private String URL_RAZA = "http://tfgivan2017.esy.es/obtenerRazas.php";
    private String URL_PERFIL =
"http://tfgivan2017.esy.es/obtenerPerfil.php";
    private String URL_UPDATE = "http://tfgivan2017.esy.es/updatePerfil.php";

    private ArrayList<Tipo> tipoAnimales = new ArrayList<Tipo>();
    private ArrayList<Razas> razasList = new ArrayList<Razas>();

    private Spinner spinTipoAnimales, spinRaza;
    int tipoAnimal, raza;
    private String nombreUsuario= "NULL";
    EditText txtnombre;
    Category stipoAnimal, sraza;

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

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowHomeEnabled(true);

        //no abrir teclado automaticamente
        getWindow().setSoftInputMode(
            WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);
    }
}
```



```

descargaImagen();
Button cambiar=(Button) findViewById(R.id.cambiarImagen);
final Button guardar=(Button) findViewById(R.id.guardarCambios);
spinTipoAnimales = (Spinner) findViewById(R.id.spinTipoAnimales);
spinRaza = (Spinner) findViewById(R.id.spinRaza);
txtnombre = (EditText) findViewById(R.id.nombreUsuario);

new GetPerfil().execute();

cambiar.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick (View v){
        openImageChooser();
    }
});
guardar.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick (View v){
        guardar();
    }
});

new GetTipo().execute();
new GetRaza().execute();

}

void openImageChooser() {
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(Intent.createChooser(intent, "Select
Picture"), SELECT_PICTURE);
}
void descargaImagen(){
    final FirebaseUser user =
FirebaseAuth.getInstance().getCurrentUser();

    uidactual=user.getId();

    String userid = user.getId();
    StorageReference storageRef =
storage.getReferenceFromUrl("gs://mascota-
social.appspot.com/photoProfile/"+userid);

    storageRef.getDownloadUrl().addOnSuccessListener(new
OnSuccessListener<Uri>() {
        @Override
        public void onSuccess(Uri uri) {
            Picasso.with(getBaseContext()).load(uri).into((ImageView)
findViewById(R.id.imagenPerfil));
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override

```



```

        public void onFailure(@NonNull Exception exception) {
            // Handle any errors
        }
    });
}
public void onActivityResult(int requestCode, int resultCode, Intent
data) {
    if (resultCode == RESULT_OK) {
        if (requestCode == SELECT_PICTURE) {
            // Get the url from data
            selectedImageUri = data.getData();
            if (null != selectedImageUri) {
                // Get the path from the Uri
                String path = getPathFromURI(selectedImageUri);
            }
        }
    }
    public String getPathFromURI(Uri contentUri) {
        String res = null;
        String[] proj = {MediaStore.Images.Media.DATA};
        Cursor cursor = getContentResolver().query(contentUri, proj, null,
null, null);
        if (cursor.moveToFirst()) {
            int column_index =
cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
            res = cursor.getString(column_index);
        }
        cursor.close();
        return res;
    }

    private void guardar(){
        final FirebaseUser user =
FirebaseAuth.getInstance().getCurrentUser();

        if(selectedImageUri != null){

            Uri image_uri= selectedImageUri;
            StorageReference storageRef =
storage.getReferenceFromUrl("gs://mascota-social.appspot.com");
            StorageReference riversRef =
storageRef.child("photoProfile/"+user.getId());
            UploadTask uploadTask;
            uploadTask = riversRef.putFile(image_uri);

            // Register observers to listen for when the download is done or
if it fails
            uploadTask.addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception exception) {

```




```

        // Handle unsuccessful uploads
    }
    }).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            // taskSnapshot.getMetadata() contains file metadata such
            as size, content-type, and download URL.
            Uri downloadUrl = taskSnapshot.getDownloadUrl();
        }
    });

    UserProfileChangeRequest profileUpdates = new
UserProfileChangeRequest.Builder()
        .setPhotoUri(Uri.parse(riversRef.toString()))
        .build();

    user.updateProfile(profileUpdates)
        .addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()) {
                    descargaImagen();
                }
            }
        });
}
stipoAnimal = (Category) spinTipoAnimales.getSelectedItem();
sraza = (Category) spinRaza.getSelectedItem();
nombreUsuario= txtnombre.getText().toString();
new updatePerfil().execute();
}

private class GetPerfil extends AsyncTask<Void, Void, Void> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected Void doInBackground(Void... arg0) {
        ServiceHandler jsonParser = new ServiceHandler();
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("uid",uidactual));
        String json = jsonParser.makeServiceCall(URL_PERFIL,
ServiceHandler.POST,params);
        Log.e("Response: ", "> " + json);

        if (json != null) {
            try {
                JSONObject jsonObj = new JSONObject(json);
                if (jsonObj != null) {
                    JSONArray categories = jsonObj

```



```

        .getJSONArray("animalPerfil");
        JSONObject catObj = (JSONObject) categories.get(0);
        tipoAnimal = catObj.getInt("tipoAnimal");
        raza = catObj.getInt("raza");
        nombreUsuario = catObj.getString("nombre");
    }

    } catch (JSONException e) {
        Log.e("ERROR: ", "> " + e.getMessage());
        e.printStackTrace();
    }
} else {
    Log.e("JSON Data", "Didn't receive any data from server!");
}
return null;
}

@Override
protected void onPostExecute(Void result) {
    super.onPostExecute(result);
    txtnombre.setText(nombreUsuario);
}

}

private class GetTipo extends AsyncTask<Void, Void, Void> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected Void doInBackground(Void... arg0) {
        ServiceHandler jsonParser = new ServiceHandler();
        String json = jsonParser.makeServiceCall(URL_TIPOANIMALES,
ServiceHandler.POST);

        if (json != null) {
            try {
                JSONObject jsonObj = new JSONObject(json);
                if (jsonObj != null) {
                    JSONArray tipoAnimales = jsonObj
                        .getJSONArray("tipoAnimales");

                    for (int i = 0; i < tipoAnimales.length(); i++) {
                        JSONObject catObj = (JSONObject)
tipoAnimales.get(i);

                        Tipo tipo = new Tipo(catObj.getInt("id"),
                            catObj.getString("nombre"));
                        tipoAnimales.add(tipo);
                    }
                }
            }
        }
    }
}

```



```

        } catch (JSONException e) {
            e.printStackTrace();
        }

        } else {
            Log.e("JSON Data", "Didn't receive any data from server!");
        }

        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        super.onPostExecute(result);
        populateSpinner();
    }
}

private void populateSpinner() {

    ArrayList<Tipo> animalesList = new ArrayList<>();

    for (int i = 0; i < tipoAnimales.size(); i++) {
        animalesList.add(new Tipo(tipoAnimales.get(i).getId(),
tipoAnimales.get(i).getNombre()));
    }

    //fill data in spinner
    ArrayAdapter<Tipo> adapter = new ArrayAdapter<Tipo>(getBaseContext(),
android.R.layout.simple_spinner_dropdown_item, animalesList);
    spinTipoAnimales.setAdapter(adapter);
    spinTipoAnimales.setSelection(tipoAnimal-1);
}

private class GetRaza extends AsyncTask<Void, Void, Void> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected Void doInBackground(Void... arg0) {
        ServiceHandler jsonParser = new ServiceHandler();
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new
BasicNameValuePair("tipoAnimal",String.valueOf(tipoAnimal)));
        String json = jsonParser.makeServiceCall(URL_RAZA,
ServiceHandler.POST,params);

        if (json != null) {

```



```

        try {
            JSONObject jsonObj = new JSONObject(json);
            if (jsonObj != null) {
                JSONArray razas = jsonObj
                    .getJSONArray("razas");

                for (int i = 0; i < categories.length(); i++) {
                    JSONObject catObj = (JSONObject) razas.get(i);
                    Raza raz = new Raza(catObj.getInt("id"),
                        catObj.getString("nombre"));
                    razasList.add(cat);
                }
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }

    } else {
        Log.e("JSON Data", "Didn't receive any data from server!");
    }

    return null;
}

@Override
protected void onPostExecute(Void result) {
    super.onPostExecute(result);
    populateSpinnerRaza();
}

}

private void populateSpinnerRaza() {

    ArrayList<Raza> razaList = new ArrayList<Raza>();

    for (int i = 0; i < razasList.size(); i++) {
        razaList.add(new Raza(razasList.get(i).getId(),
razasList.get(i).getNombre()));
    }

    //fill data in spinner
    ArrayAdapter<Raza> adapter = new ArrayAdapter<Raza>(getBaseContext(),
android.R.layout.simple_spinner_dropdown_item, razaList);
    spinRaza.setAdapter(adapter);
    spinRaza.setSelection(razasList.size()-1);

}

private class updatePerfil extends AsyncTask<Void, Void, Void> {

    @Override

```



```

protected void onPreExecute() {
    super.onPreExecute();
}

@Override
protected void doInBackground(Void... arg0) {
    ServiceHandler jsonParser = new ServiceHandler();
    List<NameValuePair> params = new ArrayList<NameValuePair>();
    params.add(new BasicNameValuePair("uid",uidactual));
    params.add(new BasicNameValuePair("nombre",nombreUsuario));
    params.add(new
BasicNameValuePair("tipoAnimal",String.valueOf(stipoAnimal.getId())));
    params.add(new
BasicNameValuePair("raza",String.valueOf(sraza.getId())));

    String json = jsonParser.makeServiceCall(URL_UPDATE,
ServiceHandler.POST,params);
    Log.e("Response: ", "> " + json);

    if (json != null) {
        try {
            JSONObject jsonObj = new JSONObject(json);
            int success = jsonObj.getInt("success");
            Log.e("result",String.valueOf(success));

        } catch (JSONException e) {
            Log.e("ERROR: ", "> " + e.getMessage());
            e.printStackTrace();
        }
    } else {
        Log.e("JSON Data", "Didn't receive any data from server!");
    }
    return null;
}

@Override
protected void onPostExecute(Void result) {
    super.onPostExecute(result);
    Toast.makeText(getBaseContext(),"Perfil
actualizado",Toast.LENGTH_SHORT).show();
}
}
}

```



Como observamos el archivo java contiene toda la lógica de la pantalla para cargar los datos que se deben mostrar y para definir el comportamiento de cada elemento.

Como se comentó anteriormente en la parte cliente, en la aplicación, hacemos uso del servidor a dos sitios diferentes: a la base de datos en MySQL a través de ficheros PHP y al servicio Firebase.

El servicio Firebase, por su lado se utiliza con:

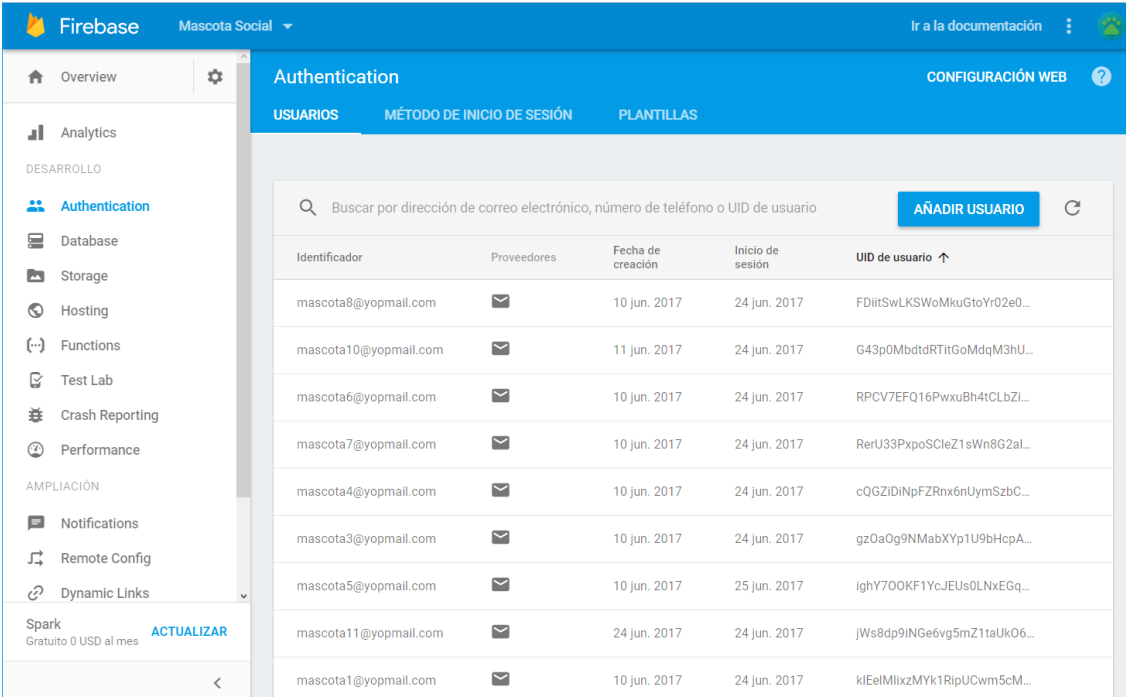
```
final FirebaseAuth user = FirebaseAuth.getInstance().getCurrentUser();
```

Teniendo la instancia de *user* podremos acceder a todos los datos de ese usuario.

La conexión a la base de datos a través de los scripts de PHP se realiza con una comunicación con JSON:

```
String json =  
jsonParser.makeServiceCall(URL_PERFIL, ServiceHandler.POST, params);
```

La configuración al servicio Firebase se hace a través de la web, pudiendo administrar las cuentas de usuarios y el almacenamiento, así como la seguridad.



The screenshot shows the Firebase Authentication console for a project named 'Mascota Social'. The 'USUARIOS' (Users) tab is selected, displaying a table of users. The table has columns for 'Identificador', 'Proveedores', 'Fecha de creación', 'Inicio de sesión', and 'UID de usuario'. There are 10 users listed, all with email providers and creation dates in June 2017.

Identificador	Proveedores	Fecha de creación	Inicio de sesión	UID de usuario ↑
mascota8@yopmail.com	✉	10 jun. 2017	24 jun. 2017	FDiitSwLKSWoMkuGtoYr02e0...
mascota10@yopmail.com	✉	11 jun. 2017	24 jun. 2017	G43p0MbdtdRTitGoMdqM3hU...
mascota6@yopmail.com	✉	10 jun. 2017	24 jun. 2017	RPCV7EFQ16PwxuBh4tCLbZl...
mascota7@yopmail.com	✉	10 jun. 2017	24 jun. 2017	RerU33Pxp0ScleZ1sWn8G2al...
mascota4@yopmail.com	✉	10 jun. 2017	24 jun. 2017	cQGZiDInpFZRnx6nUymSzbC...
mascota3@yopmail.com	✉	10 jun. 2017	24 jun. 2017	gz0a0g9NMabXYp1U9bHcpA...
mascota5@yopmail.com	✉	10 jun. 2017	25 jun. 2017	IghY700KF1YcJEUs0LNxEGq...
mascota11@yopmail.com	✉	24 jun. 2017	24 jun. 2017	JWs8dp9INGe6vg5mZ1taUkO6...
mascota1@yopmail.com	✉	10 jun. 2017	24 jun. 2017	kIEeIMlixzMYk1RipUCwm5cM...

Script PHP

Como se ha comentado anteriormente para acceder a la base de datos y poder realizar operaciones de consulta, actualización inserción y borrado se utilizan Scripts de PHP que contienen las sentencias en SQL.

Un ejemplo, como el de insertar una Mascota en la aplicación:

```
/* Obtener el archivo de configuracion de la base de datos */
require_once('db_config.php');

/* Instanciar la clase mysqli */
$mysqli = new mysqli($hostname, $username,$password, $database);

/* Controlar un posible error de conexión */
if ($mysqli -> connect_errno)
{
    die( "Fallo la conexión a MySQL: (" . $mysqli -> mysqli_connect_errno()
        . ") " . $mysqli -> mysqli_connect_error());
}
else
{
    $response = array();
    /* Controlar que nos vienen todos los parametros que se necesitan */
    if ( isset($_POST['usuario']) && isset($_POST['tipoAnimal']) &&
        isset($_POST['raza']) && isset($_POST['ubicacion']) && isset($_POST['sexo']))
    {
        $usuario = $_POST['usuario'];
        $tipoAnimal = $_POST['tipoAnimal'];
        $raza = $_POST['raza'];
        $ubicacion = $_POST['ubicacion'];
        $sexo = $_POST['sexo'];

        //MySQLi
        if ($stmt = $mysqli->prepare("INSERT INTO Mascotas
(`id`,`usuario`,`tipoAnimal`,`raza`,`ubicacion`,`sexo`)
VALUES(NULL,?,?,?,?,?,?)") )
        {
            /* Ligar parámetros para marcadores */
            $stmt->bind_param("iiii",
$usuario,$tipoAnimal,$raza,$ubicacion,$sexo);
            /* Ejecutar la consulta */
            $stmt->execute();
            /* Cerrar sentencia */
            $stmt->close();

            /* Inserción con éxito */
            $response["exito"] = 1;
            $response["mensaje"] = "La Inserción se ha realizado con
exito.";
            echo json_encode($response);
        }
    }
}
```



```

        else
        {
            /* Fallo al insertar */
            $response["exito"] = 0;
            $response["mensaje"] = "Error al ejecutar la sentencia
preparada".$mysqli->error;
            echo json_encode($response);
        }
    }
    else
    {
        /* Falta algun parametro */
        $response["exito"] = 0;
        $response["mensaje"] = "Falta algun parametro";
        echo json_encode($response);
    }
}
/* Cerrar la conexión */
$mysqli -> close();

```

Tabla Historias de la base de datos:

```

CREATE TABLE IF NOT EXISTS `historias` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `tipo` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `usuario` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `texto` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `imagen` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `fecha` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=56 ;

```



9. CONCLUSIONES

El objetivo de este proyecto era desarrollar una aplicación móvil para Android partiendo de una idea y diseñándola siguiendo la metodología de diseño centrado en el usuario.

Se ha presentado una aplicación de una red social basado en mascotas, con todas las funciones que se propusieron en la fase inicial de diseño, partiendo de las entrevistas hechas.

Valoro muy positivamente la experiencia de haber creado una aplicación desde cero.

Además, en una plataforma en la que nunca había trabajado ni conocía la forma de programación.

La motivación que me ha llevado a desarrollar en la plataforma Android es querer tener conocimiento de la plataforma móvil más utilizada actualmente ya que pienso será de gran utilidad para un posible futuro cercano en el que pueda trabajar en ella , a parte también de satisfacción personal al conocer cómo funciona la plataforma favorita y que tanto uso en el día a día.

Lo más difícil y lo que más tiempo me ha llevado es precisamente eso, aprender a programar en una nueva plataforma que no conocía y conocer todas sus particularidades y la forma de trabajar.

Saber cómo montar toda la arquitectura alrededor de la aplicación requiere que primero se conozca cómo funciona Android y saber todo lo que se necesita y valorar la mejor opción.

Aunque afortunadamente el lenguaje de programación utilizado, Java , ha sido de gran ayuda ya que en este lenguaje si tenía mucha experiencia y conocimiento sobre él.



APÉNDICE A. MANUAL DEL USUARIO

Para empezar a usar Mascota Social lo primero que tenemos que hacer es instalar el .apk que contiene la aplicación.

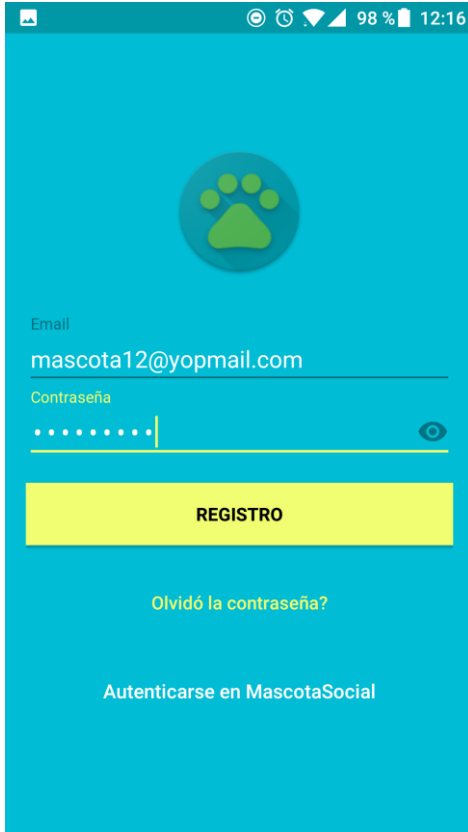
Una vez instalada nada más abrir nos sale la pantalla de login:




En esta pantalla tendremos las opciones de:

- **Autenticación:** Introduciendo el email y la contraseña se podrá entrar a la aplicación para empezar a utilizarla.
- **Registro:** Si no estamos registrados, a través de esta opción pasaremos a la pantalla de Registro.
- **Restablecer contraseña:** Podremos restablecer la contraseña si no la recordamos.

Si seleccionamos la opción de Registro, podremos ver la siguiente pantalla, en la que podremos:



12:16 98% 12:16



Email
mascota12@yopmail.com

Contraseña
.....

REGISTRO

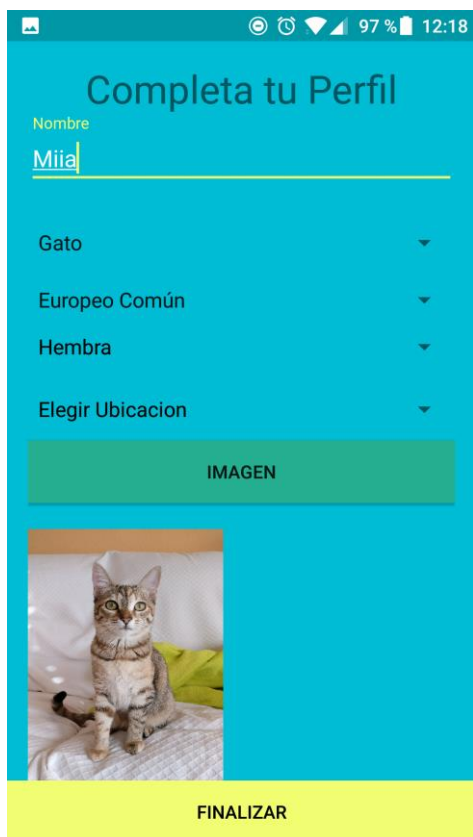
[Olvidó la contraseña?](#)

[Autenticarse en MascotaSocial](#)

- Registro: Introduciendo el email y la contraseña podremos pasar a la siguiente pantalla para completar el registro.
- Autenticarse: Podremos volver a la pantalla de iniciar sesión en la aplicación.
- Restablecer contraseña: Podremos restablecer la contraseña si no la recordamos.

Si seleccionamos la opción de registrarnos pasaremos a esta pantalla, la cual nos pide completar nuestro perfil.

Nos pedirá que completemos los siguientes datos:



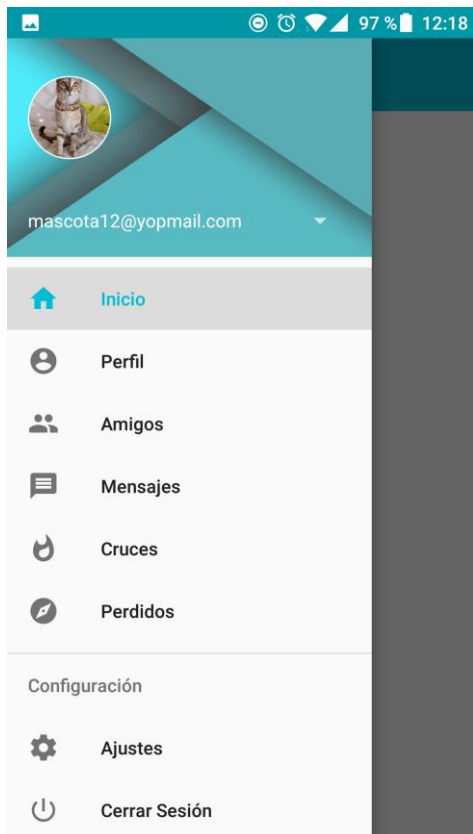
- Nombre: El nombre de nuestra mascota que queremos usar en la aplicación
- Tipo: El tipo de animal que estamos registrando.
- Raza: La raza de nuestro animal dependiendo del tipo.
- Sexo: El sexo de nuestra mascota
- Ubicación: El lugar donde se encuentra nuestro animal.
- Imagen: La imagen del animal que queremos asociar al perfil
-

La única opción que nos encontramos en esta pantalla es:

- Finalizar: Para terminar nuestro registro, y nos mandará a la pantalla principal de la aplicación ya autenticados con la cuenta que acabamos de crear.

Este es el menú desplegable de la aplicación una vez hemos iniciado la sesión.

Podemos ver las siguientes opciones de menú:



- Inicio: Es la pantalla principal de la aplicación y ahí se mostrarán las historias de nuestros amigos.
- Perfil: La pantalla donde podremos gestionar nuestro perfil subiendo nuestras propias historias en forma de texto e imágenes.
- Amigos: La sección donde podremos ver a nuestros amigos, buscar y gestionar las solicitudes de amistad.
- Mensajes: El apartado donde podremos acceder a todas las conversaciones que tengamos con nuestros amigos.
- Cruces: La opción de poder registrar a nuestra mascota para buscar cruce y poder ver a otras mascotas que también buscan.
- Perdidos: La sección donde podremos registrar la pérdida de nuestra mascota y poder ver las mascotas actualmente perdidas.
- Ajustes: En esta pantalla podremos cambiar nuestros datos de perfil.
- Cerrar Sesión: Podremos cerrar la sesión y volver a la pantalla de inicio de sesión.

Esta es la pantalla principal de la aplicación.

En ella se muestran las historias de nuestros amigos que puede ser de dos tipos:

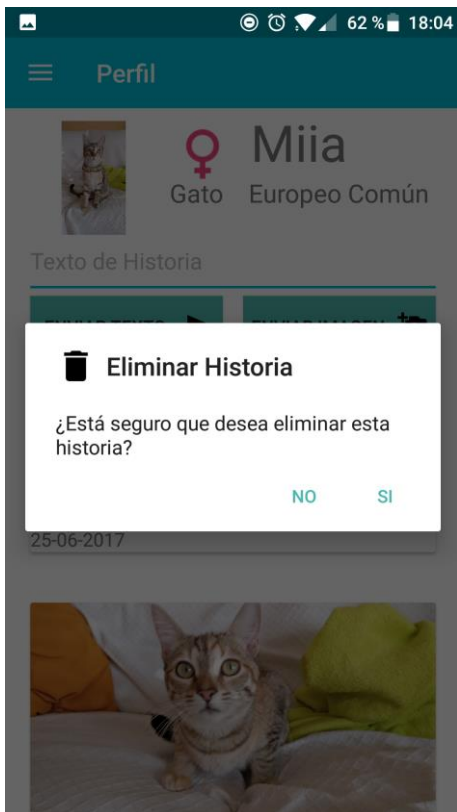


- Texto: Se muestra una tarjeta con el texto que ha introducido nuestro amigo, con su nombre y la fecha en la que lo hizo.
- Imagen: Se muestra una tarjeta con la imagen que subió nuestro amigo junto con su nombre y la fecha de publicación.
- Las historias aparecen ordenadas por orden cronológico descendente.



En la pantalla de nuestro perfil aparecerá nuestra información relativa al perfil del animal.

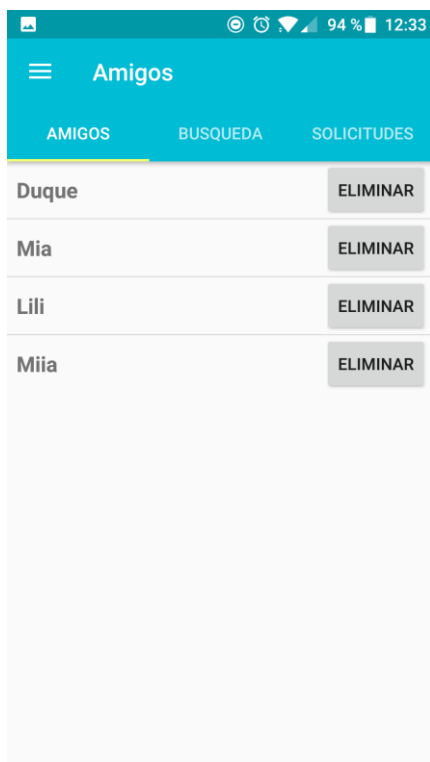
- En primer lugar, tenemos la foto de nuestro perfil con el nombre del animal y el sexo. También podemos ver el tipo de animal que es y la raza a la que pertenece.
- En segundo lugar, tenemos una caja de texto donde podremos introducir nuestra historia en forma de texto y a continuación pulsar el botón de enviar para publicar. También tenemos la opción de enviar una imagen a través del botón dedicado a ello.
- En último lugar tenemos un listado de las historias que hemos ido publicando.



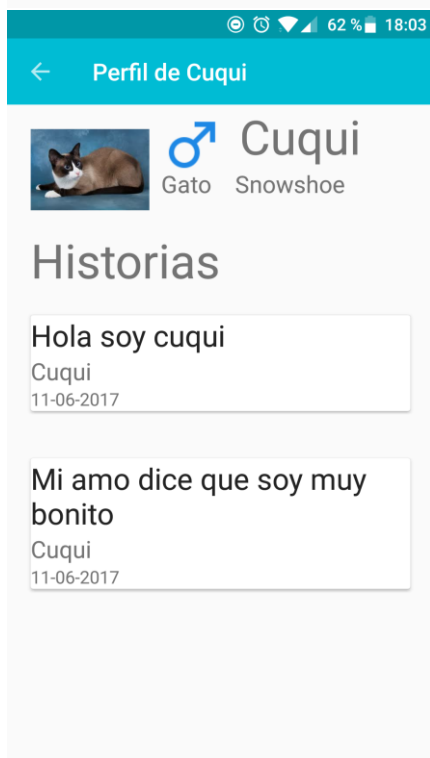
Si mantenemos pulsado en una de nuestras historias saldrá un menú contextual en el cual podremos borrar alguna de nuestras historias publicadas.



En la sección de Amigos encontramos 3 pestañas: Amigos, Búsqueda y Solicitudes.

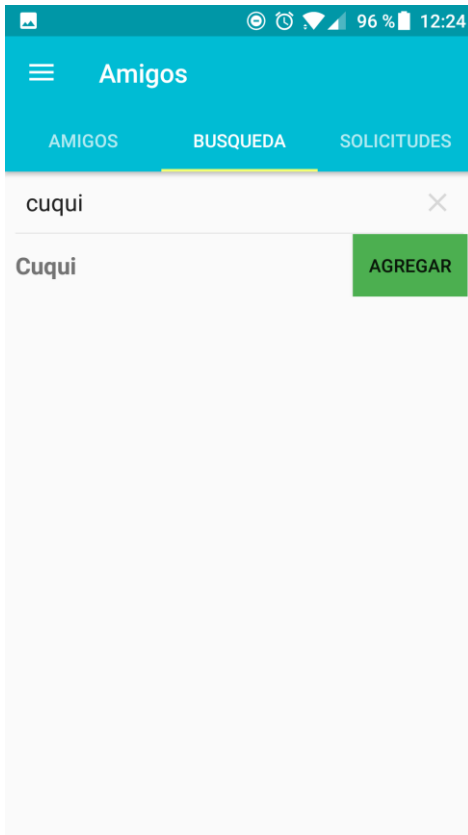


- En la primera pestaña, la de amigos podemos ver el listado de amigos que actualmente tenemos.
- Podremos eliminar de nuestra lista de amigos y así nuestra amistad.
- Si pulsamos en cualquier nombre se nos llevará a la pantalla del perfil de nuestro amigo.

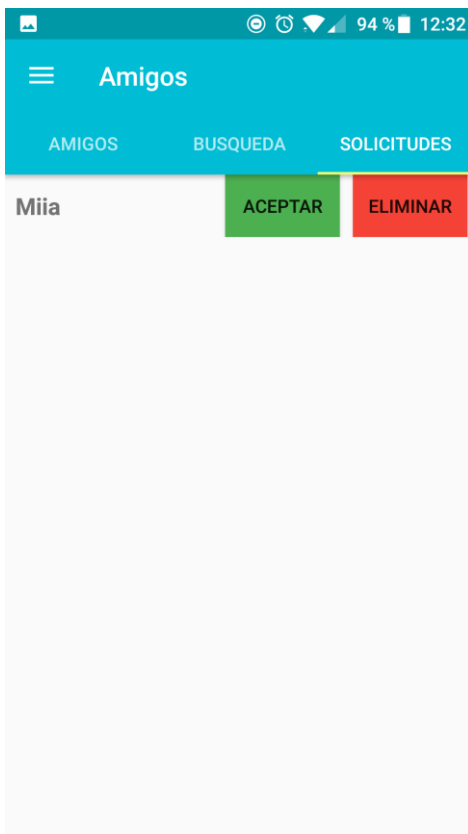


- En su perfil podremos observar su foto, así como su sexo y nombre; el tipo de animal y la raza de la que es.
- Y además las historias que él ha publicado.



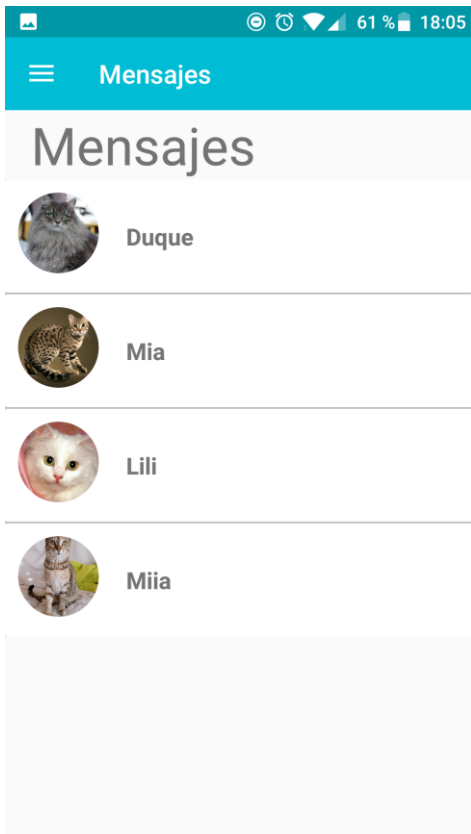


- En la pestaña de búsquedas tenemos una caja de texto donde podremos insertar el nombre de nuestro amigo y abajo nos saldrá un listado con las coincidencias en el cual tendremos un botón para enviarle una solicitud de amistad.



- En la tercera y última pestaña podremos ver las solicitudes de amistad pendientes.
- En ellas podremos aceptarlas y entonces pasaremos a ser amigos, o rechazar su solicitud y borrar la petición de amistad.





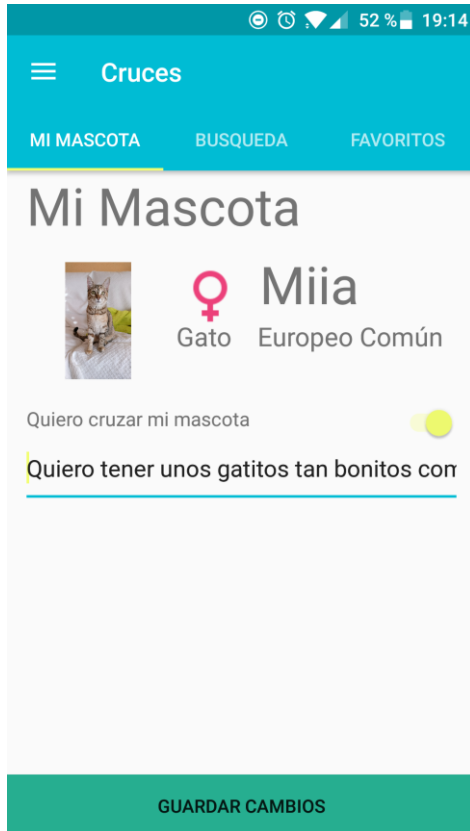
- En esta pantalla podremos observar una lista con los nombres y fotos de los perfiles de los animales amigos con los que podemos tener una conversación.
- Si pulsamos en cualquiera de ellos podremos ir a la pantalla con la conversación.



- Esta es la pantalla de la conversación con un amigo.
- En ella tenemos arriba los mensajes que estamos intercambiando. De gris y a la izquierda los de nuestro amigo y en azul y a la derecha los nuestros.
- Abajo tenemos una caja de texto donde introducir el mensaje que queremos enviar y un botón para ello.

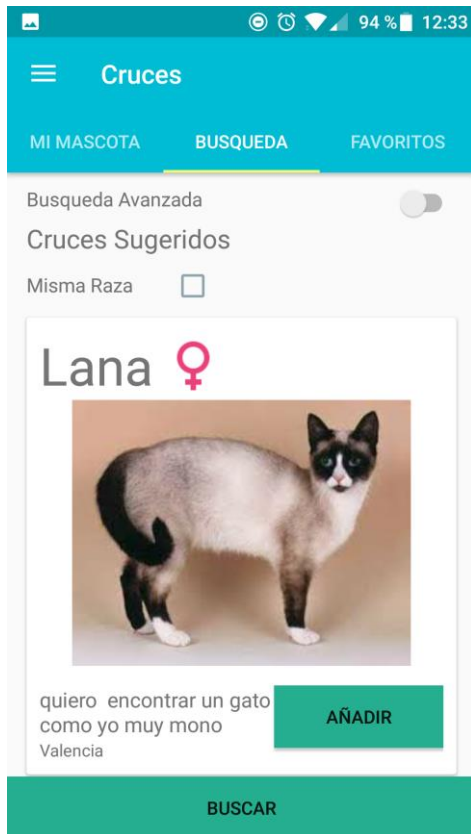


En la sección de cruces nos encontraremos 3 pestaña diferentes: Mi Mascota, Búsqueda y Favoritos.

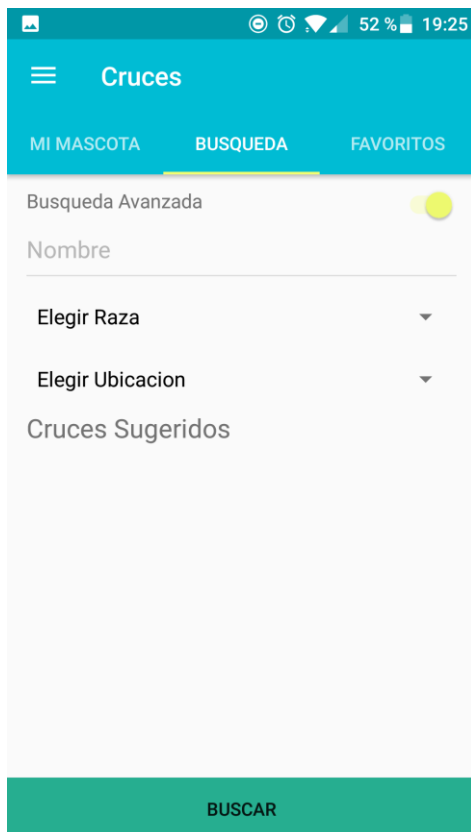


- En la primera pestaña podremos ver un resumen del perfil de nuestra mascota.
- Tendremos un deslizable el cual podremos activar si queremos buscar un cruce o desactivar si queremos borrarlos de la búsqueda de cruces.
- Si lo activamos nos aparecerá una caja de texto y un botón. En la caja de texto podremos introducir la descripción que queremos que aparezca cuando vean nuestro perfil de cruce y el botón de guardar cambios actualizará estos datos.

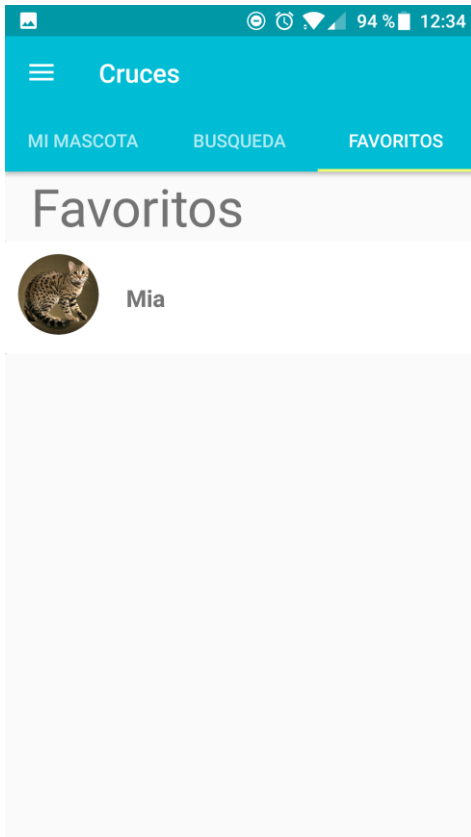




- En la segunda pestaña podremos buscar los perfiles de cruce que queramos.
- Por defecto la aplicación nos muestra unos cruces sugeridos basados en nuestros datos de perfil.
- Busca animales del mismo tipo que el nuestro con sexo opuesto y en la misma provincia.
- Tendremos una casilla para marcar si lo que queremos es que nos muestre solo los animales de la misma raza que la nuestra.
- Cuando busquemos se nos mostrará una tarjeta con la información del perfil: Nombre, sexo, descripción y ubicación.
- Con el botón de añadir podremos marcar como favorito este cruce.

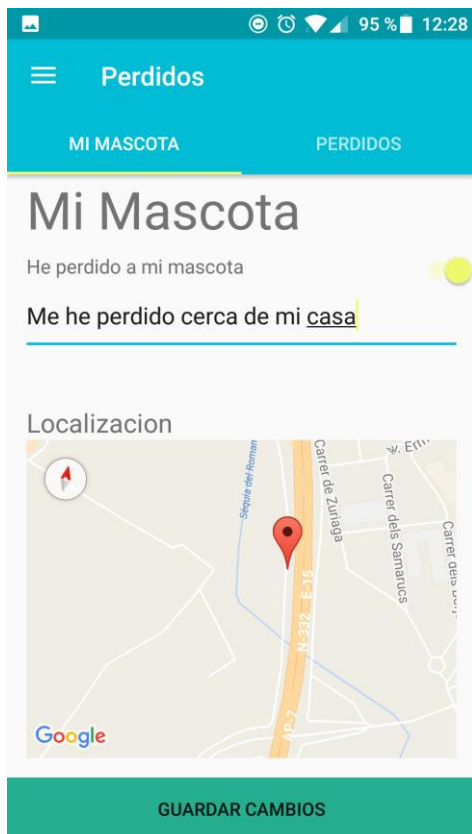


- Si habilitamos el deslizable que pone búsqueda avanzada podremos buscar según nuestras preferencias.
- Podremos filtrar los resultados por el nombre de la mascota, su raza y su ubicación.
- Con el botón buscar actualizaremos los resultados de la búsqueda con los filtros seleccionados.

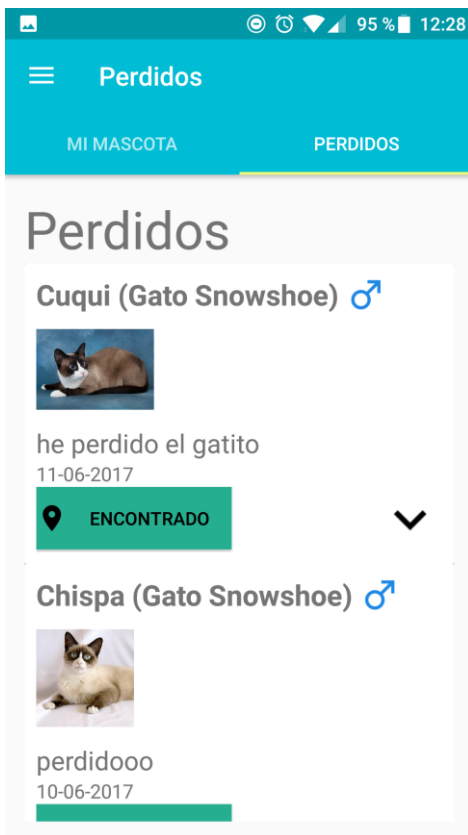


- En la última pestaña de la sección de cruces tendremos el listado de los perfiles favoritos.
- Estos son los que hemos añadido en la pestaña de búsquedas.
- Con ello, podremos ponernos en contacto con el usuario y poder concretar todos los detalles.

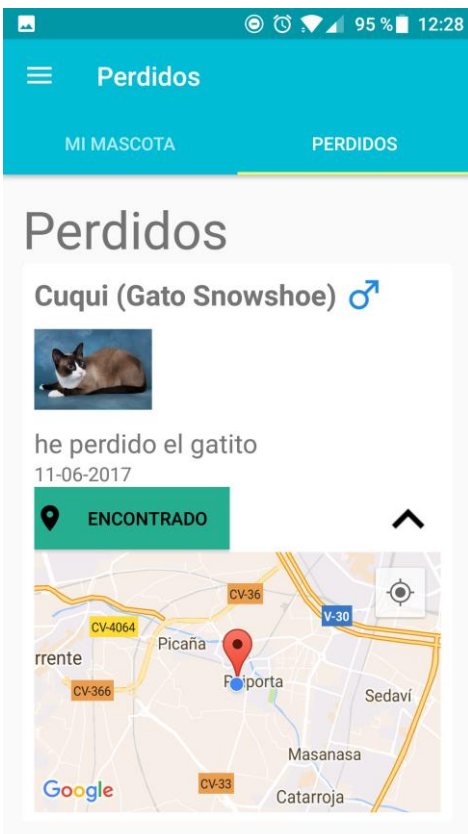
En la sección de Perdidos podremos encontrar dos pestañas: Mi Mascota y Perdidos



- En la primera pestaña podremos registrar la pérdida de nuestra mascota activando el deslizable de he perdido a mi mascota.
- Aparecerá una caja de texto en la que podremos introducir cualquier información relevante acerca de la pérdida de nuestra mascota.
- También aparecerá un mapa con nuestra ubicación para saber en que lugar se ha perdido nuestra mascota.
- Cuando hayamos encontrado a nuestro animal podremos desactivar el deslizable y así borrar de la lista de perdidos a nuestra mascota.



- En la segunda pestaña aparecerá el listado de las mascotas actualmente perdidas.
- Hay una tarjeta por cada animal que muestra una foto del animal, así como su nombre, raza, sexo, la descripción, y la fecha en la que se ha perdido.
- Hay también un botón en el caso de que lo hayamos encontrado y así avisar al usuario
- Vemos una flecha a la derecha del botón en la que si pulsamos se expandirá la tarjeta.



- Cuando expandamos la tarjeta se nos mostrará un mini mapa con la ubicación de donde se perdió el animal.
- Con ello podremos abrir la aplicación de Google Maps y saber la ubicación exacta.
- Pulsando otra vez en la flecha la tarjeta se contraerá.



APÉNDICE B. BOCETOS



Autenticación

La pantalla que vemos nada más abrir la aplicación.



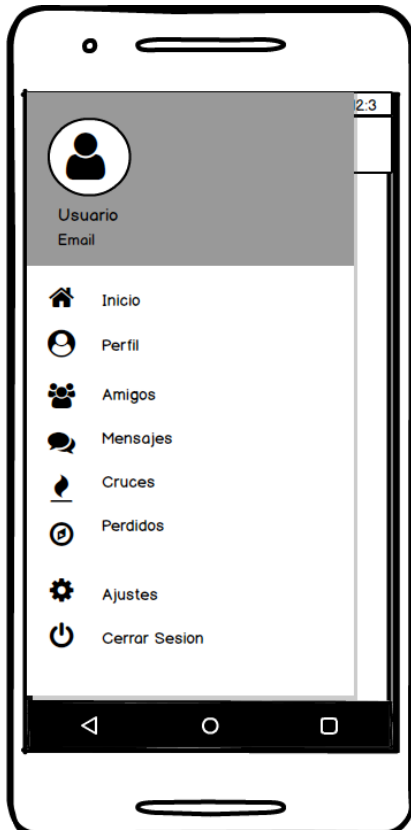
Registro

La pantalla que veremos si seleccionamos la opción de registro en la aplicación.



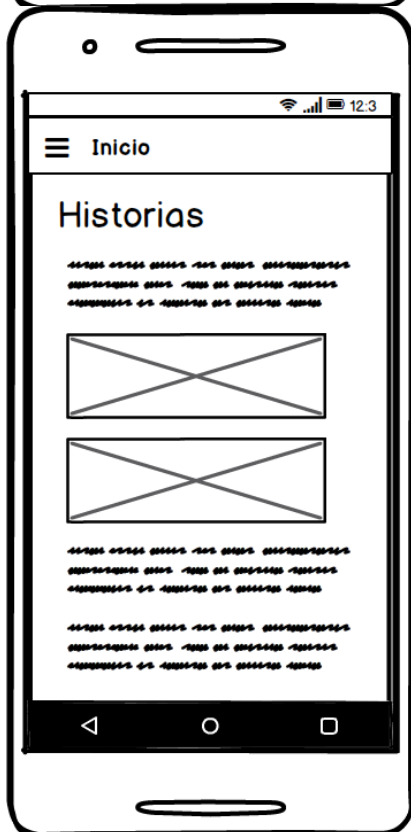
Menú

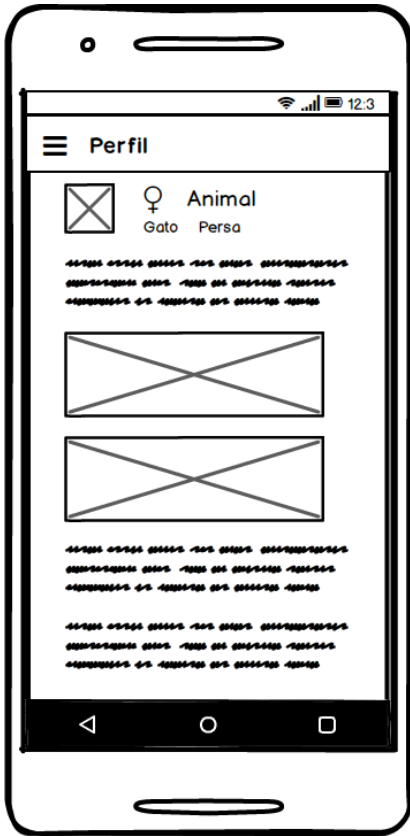
Este es el menú deslizable de la aplicación.



Historias

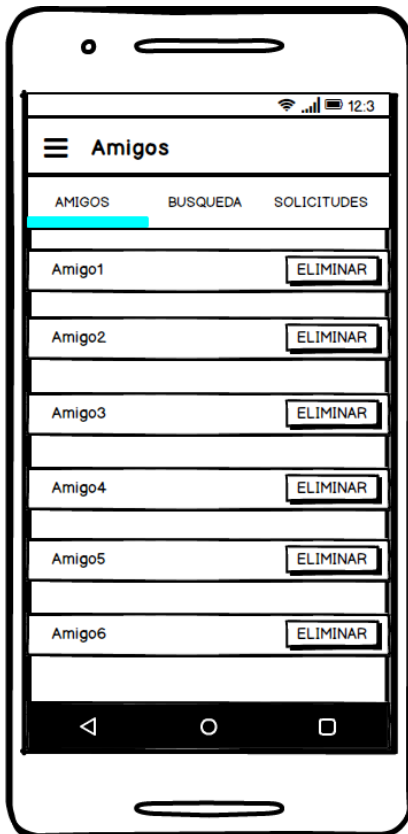
Esta es la pantalla principal que vemos al abrir la aplicación si estamos autenticados.





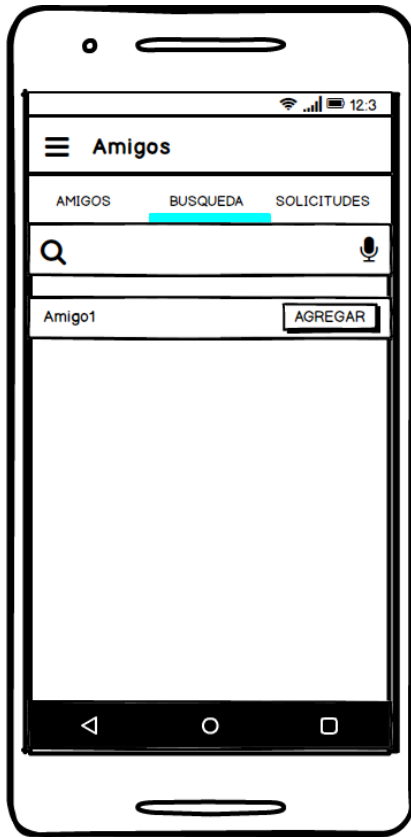
Perfil

Esta es la pantalla donde podremos ver los datos de nuestro perfil y publicar historias.



Amigos

Esta pantalla nos sirve para ver el listado de nuestros amigos



Búsqueda de amigos

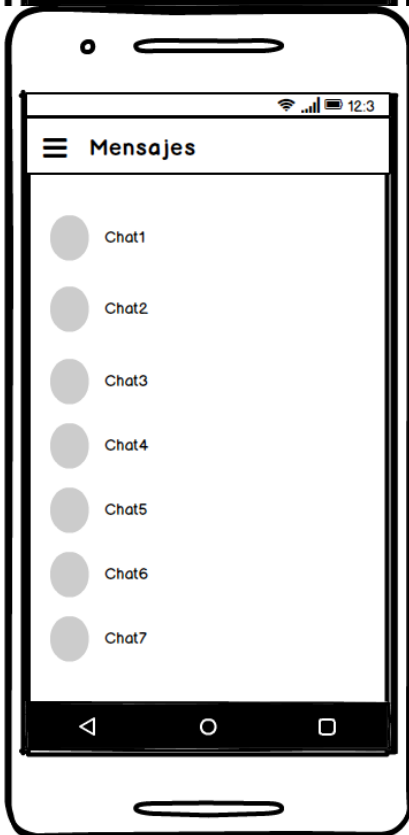
En esta pantalla podremos buscar perfiles y agregarles como amigos.





Solicitudes

Esta pantalla nos servirá para poder gestionar las solicitudes pendientes de amistad que tengamos, pudiendo aceptarlas o rechazarlas.



Mensajes

En esta pantalla nos podremos encontrar un listado de nuestros amigos para poder acceder a la conversación que mantengamos con ellos.





Cruces

En esta pantalla podremos registrar a nuestra mascota para la búsqueda de un cruce.



Búsqueda de Cruces

En esta pantalla podremos buscar cruces para nuestra mascota, sugiriéndonos en primer momento la aplicación unos.





Cruces Favoritos

En esta pantalla tendremos nuestros cruces favoritos agregados desde la pantalla de búsqueda.



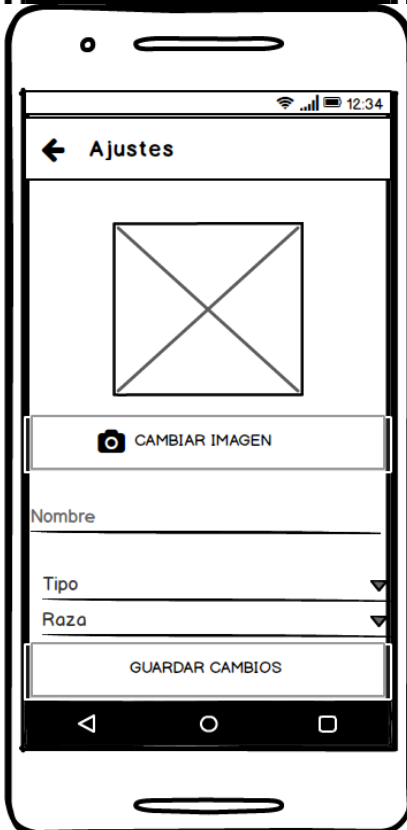
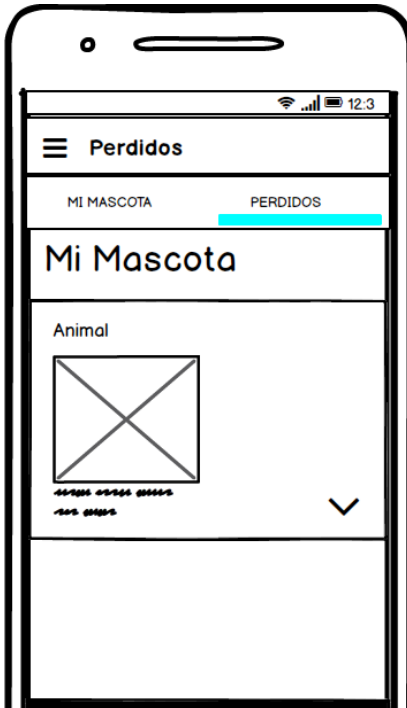
Perdidos

Esta pantalla nos servirá para poder gestionar el anuncio sobre nuestra mascota perdida.

Deslizando el control a activado podremos introducir un texto de información sobre la pérdida de nuestra mascota, así como la ubicación de su pérdida.

Mascotas Perdidas

Esta pantalla nos permitirá ver las mascotas perdidas actualmente, teniendo la información de ellas y de su última ubicación.



Ajustes

Esta pantalla nos permitirá ver y modificar los datos de nuestro usuario.

Completar Registro

Esta pantalla nos permitirá introducir todos los datos necesarios para registrarnos en la aplicación.



The image shows a smartphone screen with the title "Completar Perfil". The screen contains the following elements from top to bottom: a status bar with signal, Wi-Fi, and battery icons and the time 12:34; a text input field labeled "Nombre"; four dropdown menus labeled "Tipo de Animal", "Raza", "Sexo", and "Ubicacion"; a button labeled "IMAGEN" above a placeholder box with an 'X' inside; and a final button labeled "FINALIZAR". The phone's navigation bar with back, home, and recent apps icons is visible at the bottom.

BIBLIOGRAFÍA

- (1) UPV. *Apuntes DCU de 3º Grado Ingeniería Informática.*
- (2) GÓMEZ, SALVADOR. *Curso de Programación en Android* <<http://www.sgoliver.net/blog/curso-de-programacion-android/indice-de-contenidos>> [Consulta: 10 de febrero de 2017]
- (3) VIDEUMCORP. *Navigation Drawer con Fragments* <<https://desarrollador-android.com/material-design/desarrollo-material-design/pautas-desarrollo/navigation-drawer-con-fragments/>> [Consulta: 20 de febrero de 2017]
- (4) ANDROIDHIVE. <<http://www.androidhive.info/2012/05/how-to-connect-android-with-php-mysql/>> [Consulta: 22 de febrero de 2017]
- (5) GOOGLE. <<https://firebase.google.com>> [Consulta: 22 de febrero de 2017]
- (6) STACKOVERFLOW. <<https://stackoverflow.com>> [Consulta: 1 de marzo de 2017]
- (7) GOOGLE. <<https://developers.google.com/maps/documentation/android-api/?hl=es-419>> [Consulta: 15 de marzo de 2017]
- (8) CODERSFOLDER. <<http://codersfolder.com/2016/08/simple-crud-with-php-mysql/>> [Consulta: 15 de marzo de 2017]
- (9) PHP. <<http://php.net/>> [Consulta: 15 de marzo de 2017]

