Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una aplicación móvil en Android para un sistema de aviso y localización de animales perdidos

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Alejandro Sanjuán Quiles

Tutor: José Luis Poza Luján

Cotutor: Juan Luis Posadas Yagüe

Curso Académico 2014-2015

Resumen

El presente trabajo de fin de grado describe paso a paso el proceso de desarrollo de una aplicación Android que forma parte de un sistema distribuido mayor. La aplicación ofrece a la sociedad la posibilidad de enviar avisos multimedia relativos a la pérdida o avistamiento de animales a un servidor remoto para su posterior cotejamiento.

De esta forma se consigue ayudar a las asociaciones protectoras de animales en la tarea de localizar a los animales perdidos o abandonados para que así puedan ser recogidos, cuidados o dados en adopción.

En este documento se describe el trabajo realizado consistente en: un análisis del mercado actual en aplicaciones de este ámbito, una especificación de los requisitos que la aplicación ha de cumplir, un diseño para la aplicación, su implementación y por último las pruebas realizadas.

Palabras clave: animal, mascota, perdido, abandonado, aviso, aplicación, app, Android, sistema distribuido, servicio web, REST, Java.

Abstract

The present project describes step by step the whole development process made for an Android application, which is part of a greater distributed system. The application provides the society with the means for sending warnings related with the lost or sighting of animals to a remote server for its later collation.

By this method the Animal Protection Societies get some assistance in the task of locating lost or abandoned animals. After that the animals can be gathered up, taken care of, and given a new home.

In this document is described the work carried out consistent in an analysis of the current market in applications of this scope, an specification of the requirements that the application have to meet, a design for the application, and its implementation and finally the tests that have been made.

Keywords: animal, pet, lost, abandoned, warning, application, app, Android, distributed system, web service, REST, Java.

Tabla de contenidos

Índice de figuras

Índice de tablas

1. Introducción

1.1. Contexto y motivación

La pérdida o abandono de animales domésticos es un tema que siempre ha existido y que genera una gran preocupación social. Para luchar contra este hecho existen las asociaciones protectoras de animales que hacen un gran esfuerzo encargándose de recoger, cuidar, y dar un hogar a estos animales en cuanto se tiene noticia de ellos. El problema es que cuando existe un animal perdido o abandonado, esto no siempre llega a las asociaciones no pudiendo así actuar en estos casos.

Hoy en día con las nuevas tecnologías móviles como son los dispositivos Android, se ha conseguido que la sociedad tenga a su disposición en cualquier lugar y momento la posibilidad de tomar todo tipo de información utilizando una gran diversidad de hardware como son las cámaras de fotos, los grabadores de audio o las pantallas táctiles, y transmitirla fácilmente a cualquier lugar del mundo.

De ahí surge la idea de utilizar estas tecnologías móviles para solucionar el problema antes mencionado y poner a disposición del público general de la posibilidad de ayudar a estas asociaciones protectoras de animales en su gran labor y así de esta manera reducir en lo posible el número de animales en la calle.

1.2. Objetivos

El objetivo principal de este trabajo es el de crear una aplicación Android (véase **[1]**, **[34]**, **[35]** y **[36]**) que permita a sus usuarios el envío de avisos para notificar la pérdida de una mascota o el avistamiento de un animal perdido o abandonado. Esto no se limita a la implementación de la aplicación si no a todas las fases del proceso de desarrollo. Durante los diversos capítulos de este trabajo se trataran cada una de estas fases hasta obtener finalmente como resultado esta aplicación.

1.3. Descripción del documento

En el segundo capítulo del presente documento se hará un análisis del mercado actual extrayendo las características destacables en los sistemas actuales y las que son deseables para una aplicación de esta naturaleza. Seguidamente en el tercer capítulo se seleccionarán un subconjunto alcanzable de estas características y se establecerá un contrato sobre qué es lo que la aplicación tendrá que hacer y cómo habrá de hacerlo mediante una especificación de requisitos.

En el capítulo cuarto se partirá de la especificación ya definida para presentar un diseño detallado para la aplicación. En el capítulo quinto del documento se presentará la implementación realizada para la aplicación y será puesta a prueba para comprobar que cumple con lo que es esperado para ella. Finalmente en el sexto capítulo se extraerán conclusiones de todo el trabajo realizado y se hablará de cuáles son los posibles siguientes pasos a tomar.

2. Análisis estratégico

2.1. Introducción

En este primer capítulo se hará un análisis de las aplicaciones existentes en el ámbito del seguimiento de animales perdidos o abandonados para las principales plataformas móviles. Este análisis consistirá en la extracción de las funcionalidades y características presentes en estas aplicaciones y hacer una comparativa para finalmente extraer cuáles son las características indispensables para nuestra aplicación y cuáles son deseables para conseguir que tenga éxito.

2.2. Sistemas similares

Para comenzar este análisis se ha realizado una búsqueda exhaustiva de aplicaciones similares para las dos principales plataformas existentes en el mercado actualmente. Estos son los sistemas que utilizan el sistema operativo Android basado en Linux y los que utilizan el sistema operativo iOS de Apple.

Se ha extraído información en primer lugar de la página de descarga de las aplicaciones. En el caso de Android, se ofrece una gran cantidad de información y esta es accesible además mediante página web. Por otro lado Apple ofrece algo menos de información, pero la existente también se ha tenido en cuenta en este análisis.

En segundo lugar se han bajado las aplicaciones, y se han probado en la medida de lo posible para ver qué es lo que pueden y no pueden hacer. En muchos casos esto no ha sido posible al requerir una cuota para registrarse o porque requiere hardware adicional. En los casos en que la información no se ha podido obtener por un motivo u otro se ha indicado. Seguidamente se presentarán las aplicaciones estudiadas.

2.2.1. Animales Sin Hogar

Esta aplicación (véase **[2]**) dispone de tres paneles:

- Publicaciones
  Permite publicar sobre animales perdidos, animales encontrados, o animales en adopción. Cada una de las publicaciones permite añadir una única foto, texto describiendo el motivo de la publicación o para añadir información general, una ficha en la que se añade el tipo de animal (perdido, encontrado, en adopción), su especie, su sexo, la hora de la publicación, el nombre y teléfono de la persona de contacto, y por último una opción de ubicación con mapa. No permite añadir publicaciones desde dentro de la *app*.
- Mapa
  La aplicación muestra un mapa únicamente de la región cercana a Montevideo, con algunos avistamientos marcados en él. Este mapa es diferente al de la sección anterior.
- Acerca de ASH
  En este panel se ofrece una descripción de la asociación protectora detrás de la aplicación, así como la página web de esta y una dirección de email de contacto.

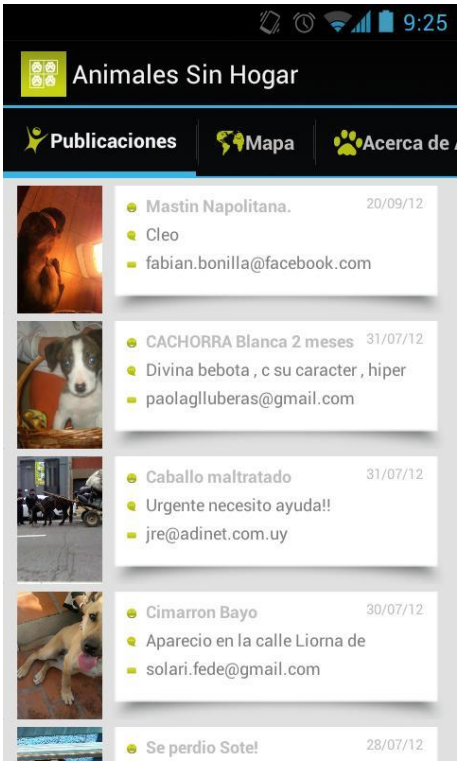

Figura 1 – Animales Sin Hogar (Google Play)

2.2.2. Alerta Animal

Esta aplicación (véase **[3]**) permite al usuario crear una denuncia en situaciones en las que aparecen animales siendo maltratados o abandonados. Dispone de dos partes.

La primera parte nos ofrece información relativa a:

- ¿Qué situaciones son denunciables?
- ¿Quién recibe mi denuncia?
- ¿Puedo preservar el anonimato?
- ¿Cómo puedo informarme sobre la situación denunciada?

A su vez nos ofrecen diferentes enlaces a páginas web y a redes sociales (Facebook y Twitter) propias de la asociación. La navegación se hace dentro de la propia *app*.

La segunda permite hacer una denuncia. Pudiendo añadir texto, una única foto, un video y por último la ubicación.

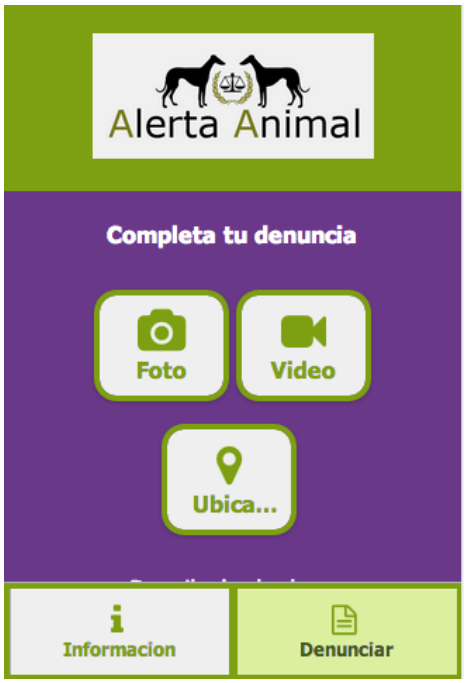

Figura 2 – Alerta Animal (Google Play)

2.2.3. PetFinder.my

Esta aplicación (véase **[4]**) dispone de publicaciones del estilo *post*, en diferentes secciones, con la posibilidad de utilizar filtros en todos los casos. Estas disponen de una foto principal, con información general del animal como por ejemplo de qué animal se trata, si está perdido, en adopción o encontrado o su sexo y edad. También dispone de localización aunque aparentemente sin mostrarla en un mapa, de una galería de fotos y videos, de información de contacto y de la opción de dejar comentarios.

Al entrar a la aplicación se accede directamente a una pantalla con publicaciones relativas a animales en adopción. Pulsando en un botón de menú nos permite acceder a nuevas opciones. Estas son:

- Find a pet: Se trata de la pantalla inicial de adopción.
- Lost & Found: Publicaciones relativas a animales perdidos o encontrados.
- Visual Map: Dice buscar avisos relativos a la posición del usuario pero muestra un mapa exclusivamente de Malasia.
- Favourites: Requiere autenticarse.
- My account: Requiere autenticarse.
- About Us: Dispone de un pequeño texto sobre el portal web de la aplicación, algunos links así como información sobre la *app*.

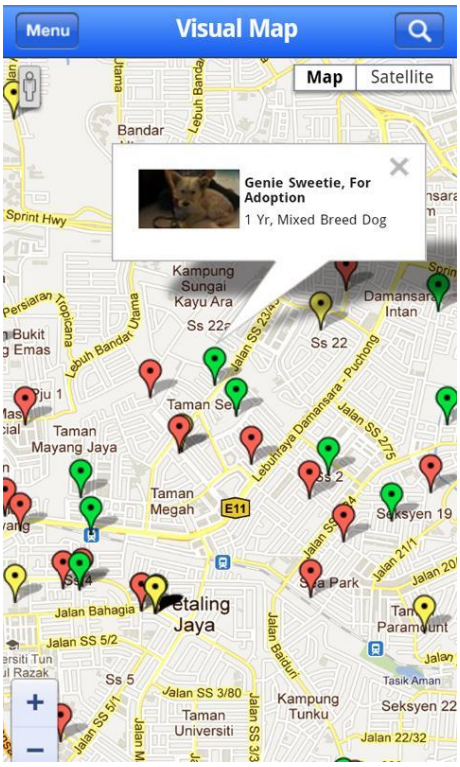

Figura 3 – PetFinder.my (Google Play)

2.2.4. HeLP Lost and Adoptable Pets

Aplicación (véase **[5]**) cuyo objetivo es ayudar a los animales perdidos.

Dispone de cinco secciones:

- My pets: Requiere autenticarse.
- Adoptable Pets: Te lleva a un formulario para filtrar los animales que se pueden adoptar.
- Missing Pets: Igual que la anterior pero para animales perdidos.
- Emergency: Ofrece una lista de diferentes servicios, no solo de emergencia, sino también de otros servicios relativos a mascotas. Permite seleccionar más de una opción, pero al final te lleva a un formulario idéntico en la web de la asociación.
- Snap and Send: Permite enviar fotos de posters de la calle sobre animales perdidos.

Figura 4 – HeLP (Google Play)

2.2.5. Charleston Animal Society

Esta *app* (véase **[6]**) dispone de muchos menús. Algunos de ellos son:

- Donate
- Adopt: Permite ver publicaciones sobre perros y gatos en adopción. Muestra información general del animal así como una foto opcionalmente.
- Events: En nuestro caso no muestra ningún evento.
- Bark Wall: Pequeña red social del estilo de Facebook pero para animales que permite compartir comentarios, ubicaciones y fotos, permitiendo filtrar por recientes o por cercanía.
- Alerts: No muestra alertas.
- Dog parks: Nos muestra en mapa o en lista diferentes parques para perros en Nueva York. No funciona muy bien, y ni siquiera está acabada.
- Submit a photo
- Lost & Found: No funciona.
- Dog Training
- Facebook y Twitter
- Volunteer Program
- Adoption Notes
- About us y About the *app*

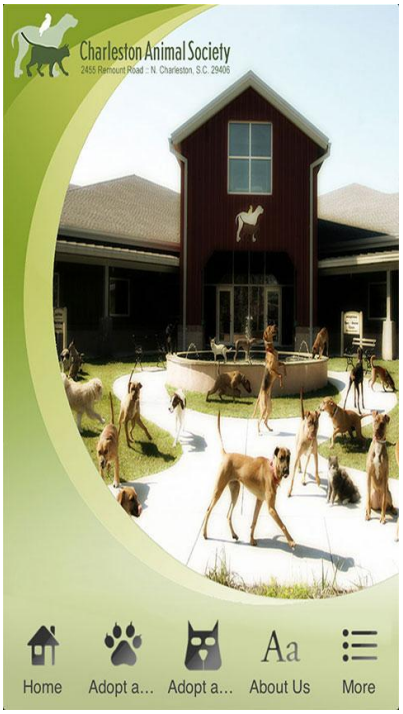

Figura 5 – Charleston Animal Society (Google Play)

2.2.6. Callejerito Help!

Aplicación (véase **[7]**) que tiene como función recoger datos de animales encontrados introducidos por el usuario, así como una foto y su localización y automáticamente hacer un cotejamiento con posibles similitudes en el ámbito mundial. Permite una descripción del animal bastante completa mediante formularios, y no da rodeos, ya que no dispone de un menú como es normal en otras *apps*, si no que directamente se hace la notificación. Solo permite perros y gatos.

Figura 6 – Callejerito Help! (Google Play)

2.2.7. Guau! qué perros

Esta *app* (véase **[8]**) nada más entrar, nos muestra tres menús parecidos, uno para perros en adopción, otro para perros perdidos, y otro para perros encontrados, así como un cuarto menú para la configuración de la *app*.

En los tres primeros nos muestra avisos según el menú en el que se está en ese momento. Además permite publicar un nuevo aviso. A la hora de hacer una publicación, se nos permite introducir información general del animal, una descripción, así como una foto.

En el cuarto menú, nos permite configurar un filtro para los otros tres menús. Permite filtrado por ciudad (España), raza del perro o sexo.

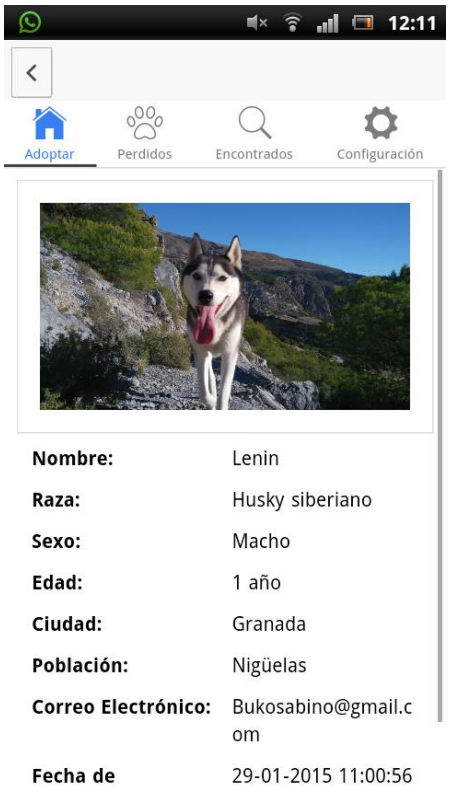

Figura 7 – Guau! Qué perros (Google Play)

2.2.8. Encuéntralos

Aplicación (véase **[9]**) que muestra animales perdidos, filtrados por proximidad al usuario. Para cada animal perdido podemos ver su foto, su descripción e información de contacto, la ubicación en un mapa español sin nombres (pero permite hacer zoom) y por último dejar comentarios.

También si hemos iniciado sesión nos permite algunas otras opciones entre ellas la de subscribirse a las actualizaciones sobre una mascota concreta, o introducir un aviso de pérdida o de encuentro de una mascota. En este caso permite introducir la información de contacto, una foto del animal, una descripción en forma de texto y la localización y fecha del suceso.

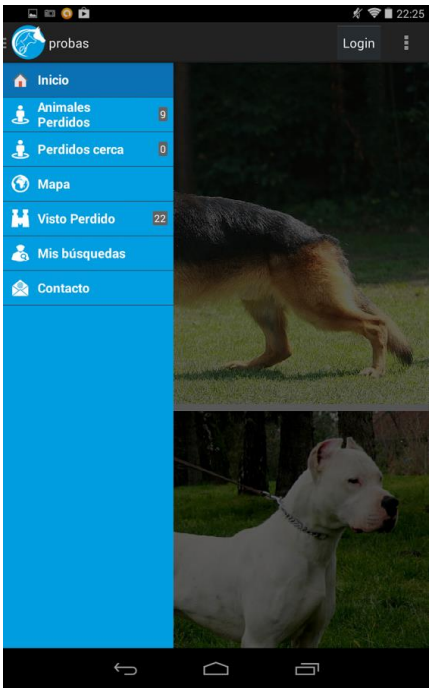

Figura 8 – Encuéntralos (Google Play)

2.2.9. Find My Pet

Esta aplicación (véase **[10]**) requiere estar registrado, permitiéndote hacerlo desde la *app*. Al registrarte dice que tomará información como la IP e ID del dispositivo, además te pide tu posición, email y número de teléfono.

Una vez registrado accedes al menú principal el cual incluye:

- My pets: Permite crear una ficha de tu mascota incluyendo información general y foto. La forma de introducir la información es de tipo formulario.
- I Lost My Pet: Deja poner un aviso indicando que una mascota se ha perdido pudiendo añadir una descripción y una posible recompensa ambos en forma de texto, así como añadir la ubicación. También permite ver alertas.
- Lost Pets Around Me: Muestra un mapa mundial sin ningún aviso en él.
- Lost and Found: Al entrar a esta página sale un mensaje de "No has recibido ninguna alerta aun", refiriéndose a que no se ha puesto ningún aviso relativo a una mascota propia perdida. También permite poner un aviso de animal encontrado pudiendo añadir un comentario de 255 caracteres, la especie, y la ubicación.
- Articles: Este menú permite ver en forma de páginas web dentro de la misma *app* una gran cantidad de artículos sobre el cuidado de las mascotas.

Esta aplicación utiliza una cantidad ingente de publicidad intersticial (*pop-ups*). A pesar de ello funciona ágilmente además de disponer de una interfaz muy cuidada y que el usuario agradecerá.

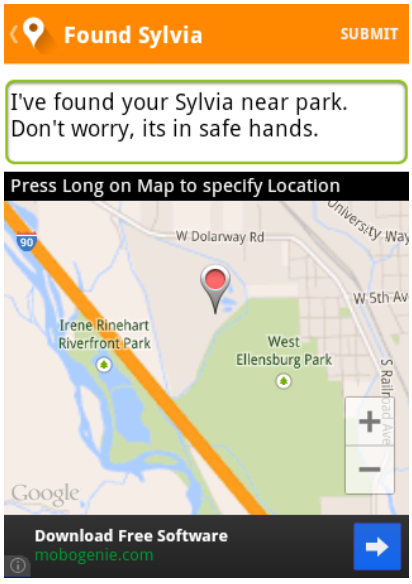

Figura 9 – Found My Pet (Google Play)

2.2.10. Pet Finder

Aplicación (véase **[11]**) que permite, tras hacer *log in* (inicio de sesión) con Facebook, notificar la pérdida o el encuentro de una mascota rellenando un formulario idéntico en los dos casos que permite insertar el nombre de la mascota, la ubicación, la descripción, el email, el teléfono de contacto y una foto.

Por otro lado te permite buscar notificaciones recientes y cercanas, y además, ofrece información sobre diferentes organizaciones de animales de Bogotá.

Figura 10 – Pet Finder (Google Play)

2.2.11. Pet Search

Las diferentes secciones de esta *app* (véase **[12]**) son:

- About us
- Receive Lost Alerts: Permite registrarte para ser avisado cuando un animal es encontrado cerca tuya.
- Animal Shelters
- Found Stray: Rico y bastante extenso formulario para notificar que un animal ha sido encontrado
- Lost a pet: Información sobre qué hacer al perder un animal.
- Información de contacto, de redes sociales, y de videos, así como de promociones.

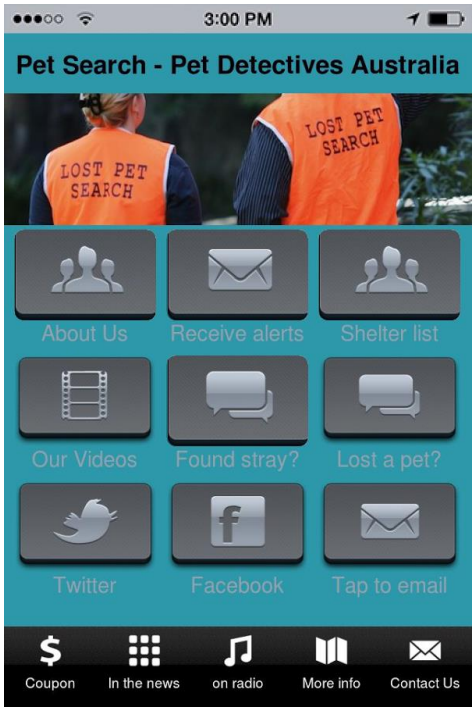

Figura 11 – Pet Search (Google Play)

2.2.12. Back2Gether

Se trata de una aplicación para iPhone que tras un pequeño tutorial, te lleva directamente a un mapa con la opción de ver perros perdidos, encontrados o ambos. Desde aquí puedes ver cualquiera de los avisos o poner uno propio.

En el segundo caso, si se pretende introducir un animal encontrado te pide hacer *log in* con Facebook permitiendo añadir información general del animal, información de contacto, una ubicación en forma geocodificada (muestra al usuario el nombre del lugar más cercano), y la fecha.

Si se pretende introducir un animal perdido, tras introducir la misma información que antes, la aplicación busca si hay coincidencias y si no las hubiese, entonces te pide hacer el *log in* con Facebook para añadir la información de contacto y poder poner el aviso.

2.2.13. Pet Position

En esta aplicación podemos ver y añadir alertas de animales perdidos o encontrados, pudiendo añadir el nombre, la especie, marcas distintivas, foto y temperamento, así como un email. También nos permite añadir mascotas propias para avisar de su pérdida, así como registrarte para encontrar gente con quien pasear el perro.

2.2.14. PetCrumbs

Nada más arrancar esta *app*, nos encontramos con un menú que pone a nuestra disposición 6 opciones:

- Report a Stray Pet: esta opción nos permite mandar un aviso de una mascota encontrada. En una primera pantalla brinda diferentes opciones para subir una foto, y seguidamente permite introducir la localización del animal y una pequeña descripción a modo de texto.
- Report My Pet Missing: requiere estar registrado (de pago o con suscripción de vecindario), por lo que no ha sido probada. Se podría suponer que es como la pantalla anterior pero con quizás algo más de información para introducir.
- Search Stray Pet: Permite buscar animales encontrados utilizando como filtros lo reciente que es el aviso, la localización, pudiendo usar la actual o por si tiene o no foto.
- Search Missing Pet: Es la misma pantalla que la anterior pero nos lleva a otra pestaña para animales perdidos. Por lo demás es igual que la anterior.
- How It Works: Lleva a un video de www.youtube.com explicando cómo usar la *app*.
- My Account: Esta opción permite autenticarse en la *app*. Crear una cuenta requiere de pago o de subscripción de vecindario.

2.2.15. iRescue Pet

Sencilla aplicación de iOS que ofrece al usuario 4 opciones. Estas son:

- Report a sighting: Permite introducir diferente información general de la mascota encontrada en forma de formulario, una foto, y según "How Does it Works?" toma la localización actual automáticamente.
- Search Pets: Esta pantalla nos permite buscar animales reportados filtrando por tipo de animal, localización, y radio tomando como centro la localización.
- How Does it Works?: Muestra una pantalla con un texto explicativo de cómo utilizar la aplicación.
- Spread the Word: Esta opción permite al usuario compartir la aplicación en diferentes redes sociales.

2.2.16. Otras aplicaciones

IcPet

Según Google Play (véase **[13]**), permite poner un aviso al encontrar un animal perdido o abandonado, y/o buscar entre los avisos cercanos a la posición actual alguno que coincida con nuestra mascota si se ha perdido. Requiere registrarse, y no funciona.

Dog Tracker

Esta aplicación funciona con un dispositivo hardware instalado en el animal. Sirve para el rastreo de este, pero no se puede acceder a la aplicación en sí si no se dispone del dispositivo.

2.3. Análisis de los sistemas similares

Una vez vistas las aplicaciones existentes en el mercado, es de gran importancia analizarlas detalladamente para extraer las características disponibles en estas aplicaciones, y hacer una comparativa para ver cuáles son los requisitos mínimos para que la aplicación de este trabajo pueda competir con las existentes y cuáles son las características que pueden hacer que la aplicación destaque entre las demás.

2.3.1. Características generales

En primer lugar se analizarán las características que son independientes de la naturaleza de la aplicación y que por el contrario, son intrínsecas a cualquier tipo de *app* móvil. Por esto mismo, estas características son fácilmente localizables en la página de descarga de cualquier *app* en el *market* correspondiente del sistema operativo.

1. Sistemas operativos (SO)

Para qué sistemas operativos móviles mayoritarios está disponible la aplicación. Se han utilizado dos posibles opciones: Apple iOS (i) y/o Android (A)

2. Modelo de negocio utilizado por la aplicación. (MN)

En este caso quizás la palabra negocio no es del todo adecuada, pero esto se refiere a la forma en que la aplicación se financia. La aplicación puede financiarse mediante publicidad (pu), mediante compra de la *app* (pa), compras dentro de la *app*, también conocido como modelo *Freemium* (fr) o es totalmente gratuita (gr).

3. Fecha de última actualización (FE)

Esto indica si la aplicación sigue siendo mantenida de alguna forma. Con algunas de ellas se verá que hace por lo menos 2 o 3 años de su última actualización, por lo que o la aplicación es perfecta y no necesita ningún cambio, o realmente ya no está siendo mantenida.

4. Número de descargas (ND)

Esta característica es importante junto a la puntuación media, como indicativo de la popularidad de la aplicación, aunque al contrario de la puntuación media, esta depende en gran medida de lo nueva que es la aplicación. Para iOS este dato no es conocido.

5. Puntuación Media y número de puntuaciones individuales (PN)

Otra característica que nos indica cuán popular es la aplicación. Es una media de las puntuaciones individuales. Por lo tanto cuantas más puntuaciones se hayan dado más precisa es. En la tabla 1 mostramos entre paréntesis el número total de puntuaciones individuales. Para iOS este dato no es conocido.

6. Web (WW)

Web de la asociación que soporta la aplicación. En algunos casos veremos que no se ha dado ninguna web si no un email. Por último, también existen casos en los que no parece haber una asociación detrás y en la tabla se muestra la dirección web o email del desarrollador.

App	**SO**	**MN**	**FE**	**ND**	**PN**	**WW**
Animales sin hogar	A	gr	09/12	1K-5K	4.6 (101)	www.animalessinhogar.com.uy
Animales sin hogar	i	gr	09/12	-	4+	www.animalessinhogar.com.uy
Alerta Animal	A	gr	10/14	10K-50K	4.7 (1134)	www.justiciaydefensaanimal.es
PetFinder	A	gr	11/13	10K-50K	4,1 (174)	www.petfinder.my
PetFinder	i	gr	12/13	-	4+	www.petfinder.my
HeLP	A	gr	06/14	1K-5K	4.0 (30)	www.helpinglostpets.com
Charleston Animal Society	A	pu	10/12	100-500	3.6 (7)	support@clientappbuilder.com
Charleston Animal Society	i	pu	11/12	-	4+	www.netgalaxystudios.com
Callejerito Help!	A	gr	05/14	500-1K	5.0 (35)	www.callejerito.com
Guau! Qué perros	A	gr	02/15	1K-5K	4.7 (50)	bukosabino@gmail.com
Encuéntralos	A	gr	08/14	10-50	4.8 (6)	japolo86@gmail.com
Find My Pet	A	pu	09/14	500-1K	4.0 (24)	www.findmypetapp.com
Pet Finder	A	gr	02/15	10-50	5.0 (2)	appscreativity@gmail.com
Pet Search	A	gr	04/14	50-100	5.0 (2)	www.petsearch.com.au
Back2gether	i	gr	11/13	-	4+	www.back2getherapp.com
Pet Position	i	fr y pu	10/14	-	4+	www.sellanapp.com
PetCrumbs	i	fr	09/14	-	4+	Nada
iRescue Pet	i	gr	02/14	-	4+	www.ib-tech.com

Tabla 1 – Comparativa de características generales para los diferentes sistemas similares

Tras los datos mostrados en la tabla 1, se pueden extraer distintos perfiles de aplicaciones. Existe un grupo numeroso de aplicaciones con altas puntuaciones medias, pero en la gran mayoría con muy pocos votos. Estos datos por sí mismos no son significativos. Mirando además el número de descargas, se aprecia que algunas de ellas tienen muy pocas descargas, mientras que otras tienen un número considerable.

En el primer caso, es algo negativo, porque significa que la aplicación no se ha dado a conocer lo suficiente. En contraposición, en el segundo caso no lo es, ya que la falta de puntuaciones puede significar solamente que la gente que ha probado la aplicación no ha aportado con su puntuación. En este caso destaca Guau Qué Perros!, una aplicación actualizada recientemente, con un buen número de descargas, y por lo general buena puntuación media.

En un segundo bloque tendríamos tres aplicaciones que destacan en sus estadísticas. PetFinder, Animales Sin Hogar, y sobre todo la española Alerta Animal. Tanto en el número de descargas, como en la puntuación y en el número de descargas, tienen valores bastante altos. Además las dos primeras tienen versión en los dos sistemas operativos, aunque no se han actualizado desde hace más de un año.

En cuanto a las aplicaciones de iOS, poco se puede decir, ya que todas tienen la misma puntuación media, y el Apple Store no ofrece más estadísticas considerables. Sí que cabe destacar que por lo general, se han actualizado más recientemente que las de Android.

Por último, faltaría resaltar, que aunque bastantes aplicaciones se encuentran soportadas por asociaciones, y que nos ofrecen diversas formas de contacto con ellos, existen unas cuantas en Android, y prácticamente todas las de iOS que no ofrecen nada de esto, y parecen tratarse de aplicaciones soportadas por desarrolladores individuales, o empresas de desarrollo, incluso algunas no ofrecen absolutamente nada, o los enlaces están caídos. Por lo tanto, estas aplicaciones podrían ser menos fiables (dado el tipo de aplicación que buscamos).

2.3.2. Introducción de datos en el sistema

Seguidamente se compararán las diferentes aplicaciones antes descritas desde el punto de vista de las modalidades de entrada que pueden utilizar. Aunque la información dada por una u otra pueda parecer diferente, después del procesamiento de esta podría disponerse exactamente de los mismos datos. En realidad el usar o unas u otras será importante sobre todo para facilitarle al usuario el uso de la *app*.

1. Texto (TX)

Permite introducir texto para describir al animal. Es la forma más simple de introducir información pero requiere que el usuario escriba toda la información que quiera introducir. También puede complementar a otro tipo de modalidad como información adicional.

2. Formulario (FM)

La aplicación permite introducir información de manera más cómoda ofreciendo al usuario valores predefinidos y/o pequeños campos de texto.

3. Localización (LO)

La aplicación toma la localización del dispositivo o una suministrada por el usuario. Esto puede ayudar a cercar el área de búsqueda para un animal perdido y además a la hora de cotejar con avistamientos, descartar los casos en que la distancia entre un suceso y otro sea excesiva.

4. Foto (FT)

Esto se refiere a si la aplicación permite adjuntar una foto al "suceso". Puede extraerse información mediante técnicas de reconocimiento de patrones. También dispondrá de esta característica si permite introducir video.

5. Audio (AU)

Permitiendo que el usuario añada una nota de voz al "suceso", este podría describir al animal de manera muy fácil, aunque como en el caso anterior, esto requiere de cierto procesamiento mediante reconocimiento de patrones.

6. Dibujo virtual (DV)

La aplicación permite hacer un pequeño dibujo del animal utilizando la información introducida mediante otros de los medios anteriores. Esto no es una manera de introducir información en sí si no una forma de hacer el proceso más agradable para el usuario.

7. Ninguna de las anteriores (NI)

La aplicación no permite ninguna de las anteriores formas de introducción de los datos. Pueden existir otras formas diferentes de introducir la información como por ejemplo mediante contacto directo (email, teléfono, correo ordinario, etc.), mediante una página web externa, o como *posts* con texto, enlaces, y demás componentes de edición HTML (como en foros o blogs). También es posible que sea la propia asociación quien la introduzca, o que no haya ninguna manera aparente de introducirla.

A la vista de la tabla 2 vemos que casi todas las aplicaciones ofrecen al usuario ofrecen modos parecidos de introducir nuevos avisos. Disponen por lo general de un formulario para introducir información descriptiva del animal, así como un cuadro de texto para introducir información adicional. Además permiten introducir una foto, y la localización del animal.

Por otro lado vemos que ninguna permite la entrada mediante audio, ni mostrar un dibujo virtual del animal. Estas son características bastante deseables ya que la primera permitiría al usuario introducir la información más fácil a cambio de un mayor procesamiento posterior, y la segunda ayudaría al usuario a hacerse una idea de lo que está introduciendo de manera visual.

App	TX	FM	LO	FT	AU	DV	NI
Animales sin hogar	No	No	No	No	No	No	Sí
Alerta Animal	Sí	No	Sí	Sí	No	No	No
PetFinder	Sí	Sí	Sí	Sí	No	No	No
HeLP	No	Sí	Sí	Sí	No	No	No
Charleston Animal Society	No	No	No	No	No	No	Sí
Callejerito Help!	No	Sí	Sí	Sí	No	No	No
Guau! Qué perros	Sí	Sí	No	Sí	No	No	No
Encuéntralos	Sí	Sí	Sí	Sí	No	No	No
Find My Pet	No	Sí	Sí	Sí	No	No	No
Pet Finder	Sí	Sí	Sí	Sí	No	No	No
Pet Search	Sí	Sí	Sí	No	No	No	No
Back2gether	Sí	Sí	Sí	Sí	No	No	No
Pet Position	Sí	No	Sí	Sí	No	No	No
PetCrumbs	Sí	Sí	Sí	Sí	No	No	No
iRescue Pet	Sí	Sí	Sí	Sí	No	No	No

Tabla 2 – Comparativa de la introducción de datos en el sistema para los diferentes sistemas similares

2.3.3. Rasgos animales

En cuanto a la descripción de un animal, podemos acudir a una gran variedad de rasgos característicos que diferencian a los animales unos de otros. Cuantos más rasgos una aplicación permita introducir, más preciso será su cotejamiento posterior a la hora de buscar coincidencias con otros avistamientos/pérdidas, y mejor resultado podrá ofrecer por tanto la aplicación.

Para poder cuantificar correctamente los rasgos que permite utilizar cada aplicación, vamos a indicar en la tabla 3 el número de posibles valores para ese rasgo concreto. En caso de que se permita introducir texto o el rango de valores quede abierto de otra forma, aparecerá una 'A' de Abierto para ese rasgo y esa aplicación. En el caso contrario de que no quede claro la forma en que se introduce el valor, se indicara 'NS' de No se Sabe. Si aparece un '+' justo después de un valor numérico indicaremos un número mayor que el indicado.

Por último, cabe destacar que por lo general el que se permitan rangos de valores abiertos, es un claro indicativo de que los datos serán posteriormente procesados por un humano, ya que estos campos son bastante difíciles de procesar por un ordenador, y por lo contrario, rangos de valores limitados podrían ser tanto procesados por humanos como automáticamente.

1. Número de especies animales (ES)

Algunas aplicaciones solo sirven para perros perdidos o abandonados. Otras además incluyen gatos. Idealmente deberían abarcar más especies.

2. Raza (RA)

La aplicación permite o no introducir la raza del animal como puede ser “Yorkshire Terrier” para una especie “Perro” o “Siamés” para un “Gato”.

3. Tamaños diferentes (TA)

Normalmente esto podrá ser 3 posibles valores (“pequeño”, “mediano”, “grande”) o ninguno. Esto puede ser subjetivo.

4. Edad (ED)

La edad del animal, puede ser difícil de discernir en un animal encontrado, pero en el caso de una pérdida, sí que será fácil de saber. A pesar de esta incertidumbre, puede utilizarse de manera orientativa ya que por ejemplo sí que podría saberse si el animal es recién nacido o viejo.

5. Sexo (SE)

Por lo general dos posibles valores o ninguno aunque en algunos casos se introduce un tercer valor como puede ser “no se aprecia” o “camada”, aunque en este último caso los otros datos quedan algo ambiguos por tratarse de más de un animal.

6. Temperamento (TM)

Si el animal es “tranquilo”, “nervioso”, “juguetón”, etc. Esto es muy subjetivo porque depende de la persona con la que se encuentre el animal, su estado en ese momento, o si no está acostumbrado a estar fuera de casa.

7. Colores (CL)

Para este rasgo normalmente con valores abiertos, en la tabla 3, se indicará en primer lugar el número de colores primarios, y seguidamente entre paréntesis el número de colores secundarios refiriéndose por ejemplo al color de las manchas, o de las rayas, etc.

8. Forma de las marcas (MA)

La forma de las marcas podrían ser manchas grandes, pequeñas, motitas, etc.

9. Rasgos característicos (RC)

Esto podría referirse a cualquiera de las anteriores, o por otro lado, cualquier otra información adicional que el usuario crea que ayudará a reconocer al animal. Esto esta generalmente relacionado con un campo de texto abierto.

App	ES	RA	TA	ED	SE	TM	CL	MA	RC
Animales sin hogar	5+	0	0	0	2	0	0	0	A
Alerta Animal	0	0	0	0	0	0	0	0	A
PetFinder	8	50+	0	6	2	0	0	0	A
HeLP	9	50+	4	4	2	0	0	0	A
Charleston Animal Society	2	NS	NS	NS	NS	NS	NS	NS	NS
Callejerito Help!	2	0	3	0	3	0	6	6	6
Guau! Qué perros	1	50+	0	A	3	0	0	0	A
Encuéntralos	4	50+	0	0	0	0	0	0	A
Find My Pet	8	A	A	A	2	0	0	0	A
Pet Finder	0	0	0	0	0	0	0	0	A
Pet Search	A	A	0	0	2	0	A	A	A
Back2gether	1	50+	3	3	2	0	0	0	A
Pet Position	A	A	0	0	0	A	0	0	A
PetCrumbs	0	0	0	0	0	0	0	0	A
iRescue Pet	5	A	3	0	0	0	A	A	A

Tabla 3 – Comparativa de los rasgos animales disponibles para los diferentes sistemas similares

2.3.4. Otras características

Además de todas las características vistas anteriormente, aún existen algunas otras que también son interesantes y merece la pena analizar pero no forman parte de ninguno de los grupos anteriores. Estas características se estudian a continuación.

1. Tipo de procesamiento (TP)

Los datos son procesados por personas como por ejemplo en el caso del modelo de *post*, automáticamente (cotejando nuevos sucesos con sucesos anteriores) o automáticamente con supervisión humana para comprobar que el resultado del cotejamiento es correcto. En la tabla 4 se indicara "H" para procesamiento humano, "A" para el automático, "S" para el caso de que este sea supervisado, o "NS" si no se puede saber.

2. Funcionamiento (FU)

Esta característica es bastante subjetiva e indica si la aplicación no funciona nada, funciona mal, o funciona correctamente. Será medida del 1 al 5 siendo "1" que no funciona y "5" que no hay ningún problema con el funcionamiento.

3. Interfaz amistosa (IA)

Es importante que la aplicación tenga una interfaz agradable, que invite al usuario a utilizarla. En la tabla 4 se mide del 1 al 5, siendo el "1" nada agradable y "5" muy agradable.

4. Filtrado (FI)

Cuando existan muchos avisos de animales perdidos o encontrados, es adecuado que exista una opción para filtrar a través de diferentes campos del aviso. Indicaremos con un valor numérico positivo el número de filtros disponibles, mientras que con un "0", la ausencia de filtros.

5. Protección de los datos (PD)

En este tipo de aplicaciones puede existir información sensible como por ejemplo la ubicación de un animal perdido. En la tabla 4 indicaremos con un "No" si existe información sensible no protegida, y por lo tanto está abierta al público y con un "Sí" si por lo contrario, sí que está protegida y solo es accesible por personas de confianza. "NS" indicará que no se conoce.

App	**TP**	**FU**	**IA**	**FI**	**PD**
Animales sin hogar	H	2	3	0	No
Alerta Animal	H	5	5	0	Sí
PetFinder	H	3	4	7	No
HeLP	NS	2	4	6	NS
Charleston Animal Society	H	1	3	1	NS
Callejerito Help!	S	4	4	0	Sí
Guau! Qué perros	H	5	5	3	Sí
Encuéntralos	H	4	2	1	No
Find My Pet	NS	3	5	0	No
Pet Finder	H	4	4	2	No
Pet Search	H	3	2	0	Sí
Back2gether	A	5	5	1	No
Pet Position	H	3	3	1	NS
PetCrumbs	H	4	5	3	No
iRescue Pet	H	4	5	3	NS

Tabla 4 – Comparativa de otras características para los diferentes sistemas similares

2.4. Síntesis de características

Una vez vistas las características existentes en las aplicaciones del mercado, lo siguiente es plantearse cómo sería la aplicación ideal buscada en este proyecto.

Esta, debería estar por lo menos disponible en Android, el sistema móvil más utilizado hoy en día. Además debería poder funcionar en sistemas a partir de Android 4.1 conocido como Jelly

Bean, alcanzando así más del 90% de dispositivos Android como se aprecia en la figura 12 (Véase **[14]**). También se ha visto que prácticamente todas las aplicaciones son gratuitas, algo que es normal, dada la naturaleza voluntaria de estas aplicaciones. Así que esta aplicación debería ser también gratuita para que tenga éxito.

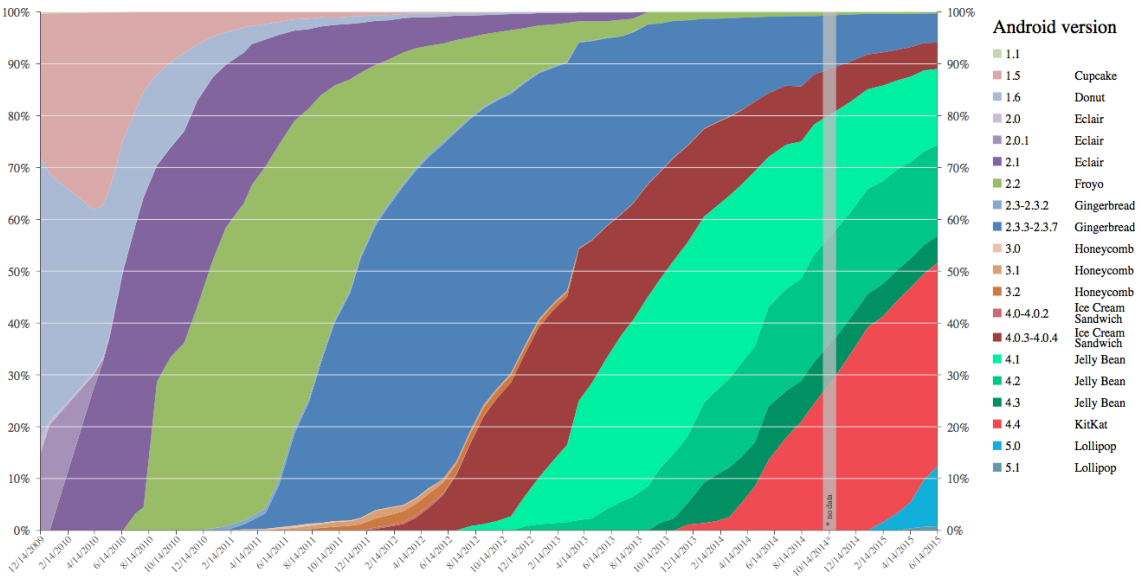

Figura 12 – Gráfica del uso de las diferentes versiones similares de Android a lo largo de los años

En cuanto a la introducción de información, hay que buscar la manera que implique menos interacciones del usuario con el sistema. Por lo tanto es deseable que el usuario pueda introducir la información mediante audio o imágenes con unos pocos toques en la pantalla, y estos puedan ser procesados en un servidor externo para extraer los rasgos del animal. Esto requiere que en algún punto del proceso, haya un humano supervisando que todo salga correcto.

Otra forma posible es que el usuario pueda introducir audio, y este sea procesado por la propia *app*, y posteriormente, un formulario corriente sea rellenado con los datos extraídos. En este caso, el usuario introduciría la información con unos pocos toques (menos que en un formulario corriente) y podría verificar/corregir la información antes de enviarla, ahorrando tiempo de supervisión en el servidor.

Además, la aplicación ideal contaría con el dibujo virtual, función que no aparece en ninguna otra aplicación. Esto se refiere a que el usuario según vaya introduciendo información, un dibujo virtual del animal se va creando para que el usuario pueda verificar lo que el sistema ha captado gráficamente, e incluso, que pueda introducirla o corregirla directamente sobre el dibujo.

Por lo que a los rasgos animales se refiere, lo ideal sería, utilizar la cantidad mínima de rasgos, que consiga identificar correctamente un porcentaje de animales por encima de un umbral (por ejemplo el 90% de los animales), siendo así estos los rasgos más característicos o discriminativos de entre todos los posibles. Si se dispone de una cantidad grande de datos reales, esto se puede averiguar mediante el uso de algoritmos como LDA, PCA, etc. (Véase **[15]**).

Además debe de existir un servidor, implementando servicios web REST, capaz de cotejar los diferentes sucesos en busca de coincidencias entre ellos. Por ejemplo si existe un aviso de un animal perdido, y se da de alta un aviso de que se ha encontrado un animal, el servidor, teniendo en cuenta los rasgos, el momento de los avisos, y la distancia entre los sucesos ha de ser capaz de decir si se trata del mismo animal o no.

Otro factor a tener en cuenta para la aplicación es que esta debería contar con diferentes perfiles de usuario y ofrecer unas funcionalidades u otras dependiendo de la persona que se haya autenticado. Los perfiles de usuario serían el de usuario colaborador, el de usuario corriente y opcionalmente el de usuario no registrado teniendo más privilegios el primero y menos el último.

Un usuario no registrado, en caso de existir, sería capaz de avisar de que un animal se ha encontrado perdido o abandonado. En cambio un usuario corriente además de esto, podría dar de alta un animal perdido. La diferencia entre un caso y otro, es que para un animal encontrado no hace falta necesariamente una dirección de contacto. Simplemente con el aviso y la localización, un usuario colaborador sería capaz de ir a buscar al animal. Estos dos perfiles, al no tener información sobre ellos, no se les permitiría ver la localización de los animales.

Por otro lado, además de lo anterior, un usuario colaborador debe de poder ver todas las incidencias y la ubicación precisa del animal, por si hiciese falta ir a buscarlo. También podría recibir notificaciones al encontrarse avisos nuevos, o cotejamientos en el servidor con resultado positivo.

Por último, esta aplicación tendría una interfaz agradable, nativa de Android, y funcionaría ágilmente (así como el servidor) y con los mínimos bloqueos. El servidor por otro lado, debe estar disponible las 24h y ser escalable para poder atender todas las peticiones. Esto es algo importante si se quiere que la aplicación tenga éxito.

2.5. Conclusiones

En este capítulo se han tomado diferentes aplicaciones móviles existentes para dispositivos Android e iOS en el ámbito del aviso y localización de animales perdidos, y se han revisado cuidadosamente sus características. Una vez hecho esto ha sido posible sintetizar todo esto, en una serie de características deseables para una aplicación ideal de estas características.

Esta aplicación ideal es un objetivo bastante difícil de alcanzar en este trabajo por cuestiones de tiempo y depende en gran medida de un servidor existente en el que confiar. Por ello, definiremos una aplicación con menor complejidad, que se adapte a las limitaciones existentes y que será la aplicación objetivo de este trabajo. En el siguiente capítulo se procederá a especificar según el formato del estándar IEEE-830 los requisitos funcionales y no funcionales para esta aplicación.

3. Especificación de requisitos

3.1. Introducción

En este capítulo se describe la Especificación de Requisitos del Software (ERS) de la aplicación móvil para el sistema operativo Android cuya funcionalidad es la de avisar y localizar animales perdidos o abandonados. Este apartado sigue la estructura definida y las prácticas recomendadas en el estándar IEEE 830-1998. Véase **[16]**.

3.1.1. Propósito

El presente capítulo tiene como propósito definir los requisitos funcionales y no funcionales de la *app* para el sistema de aviso y localización de animales perdidos. Está dirigido al equipo de desarrollo de la aplicación y servirá como punto de partida para el diseño, y posterior implementación de la misma. Además en el caso de que existiese un cliente, permitiría llegar a un acuerdo con este para definir qué es lo que tiene que hacer la aplicación.

3.1.2. Ámbito del sistema

El objetivo del desarrollo de este sistema, es el de facilitar la localización y rescate de animales perdidos o abandonados a las asociaciones protectoras de animales. Esto se conseguirá mediante el envío con la *app* Android por parte de los ciudadanos de avisos al perder un animal, o al encontrar uno perdido. Esta información recogida y almacenada por un servidor, será procesada por este, y cotejada para encontrar posibles coincidencias entre avisos.

3.1.3. Definiciones, acrónimos y abreviaturas

Término	Descripción
HTTP	HyperText Transfer Protocol es un protocolo que permite a aplicaciones ejecutándose en diferentes máquinas intercambiar información. HTTP ofrece métodos para solicitar información (GET), para enviarla (POST), actualizarla (PUT) o eliminarla (DELETE) remotamente (Véase **[17]**)
JSON	JavaScript Object Notation es un lenguaje utilizado para intercambiar objetos con datos. Su función es la de codificar estos objetos en texto plano para su posterior intercambio. Su sintaxis está basada en la de los objetos del lenguaje JavaScript (Véase **[18]**)
Localización	Par de valores consistentes en una latitud y una longitud, que definen una posición en el planeta Tierra. También pueden incluir la altitud para una mayor precisión
REST	Representational State Transfer es una arquitectura concreta para servicios Web que utiliza HTTP como canal para intercambiar mensajes y JSON o XML para codificar el mensaje enviado **[19]**
Servicio Web	Interfaz de un servidor web que permite intercambiar información con este. Esta interfaz ha sido definida mediante un conjunto de protocolos, lenguajes de programación y estándares concretos que establecen las reglas para usarlo. Una máquina interesada en consumir este servicio Web ha de conocer la interfaz y adherirse a estas reglas para poder establecer la comunicación (Véase **[20]**)
XML	eXtensible Markup Language, al igual que JSON es un lenguaje cuya función es la de codificar objetos en forma de texto plano para su posterior intercambio (Véase **[21]** y **[22]**)

Tabla 5 – Definiciones, acrónimos y abreviaturas usados en la especificación de requisitos

3.1.4. Visión general del documento

El punto de especificación de requisitos consta de 3 secciones:

- La primera sección (sección actual) realiza una breve introducción al apartado de especificación de requisitos.
- La segunda sección describe todos aquellos factores que afectan al producto y a sus requisitos. No se describen los requisitos, sino su contexto. Además proporciona una visión muy general del sistema.
- Por último, la tercera sección contiene los requisitos a un nivel de detalle suficiente como para permitir a los diseñadores diseñar un sistema que satisfaga estos requisitos, y que permita al equipo de pruebas planificar y realizar las pruebas que demuestren si el sistema satisface, o no, los requisitos.

3.2. Descripción general

Antes de detallar los requisitos del sistema, es importante hacer una descripción del sistema de alto nivel y analizar aquellos factores que afectan al producto y a sus requisitos. Esta sección incluye: perspectiva del producto, funciones del producto, características de los usuarios, restricciones y suposiciones.

3.2.1. Perspectiva del producto

La aplicación móvil objetivo de este proyecto forma parte de un sistema mayor, el cual incluye una aplicación servidor que implementará una interfaz de servicios Web REST. Estos permitirán el intercambio de información con la aplicación móvil mediante ficheros JSON o XML. La información enviada, que puede incluir archivos multimedia como grabaciones de audio o imágenes, en la mayoría de los casos serán avisos de animales perdidos o encontrados, aunque podrá tratarse de otra información necesaria para el funcionamiento de la aplicación.

Una vez enviada la información al servidor esta podrá ser guardada en una base de datos, y más tarde ser procesada. A su vez el servidor responderá a la aplicación móvil de la misma forma avisando al usuario a través del dispositivo móvil de si todo ha salido correctamente o no. En la figura 13 extraída de **[23]** se muestra la visión global del producto, y las interacciones entre la aplicación móvil Android y el servidor.

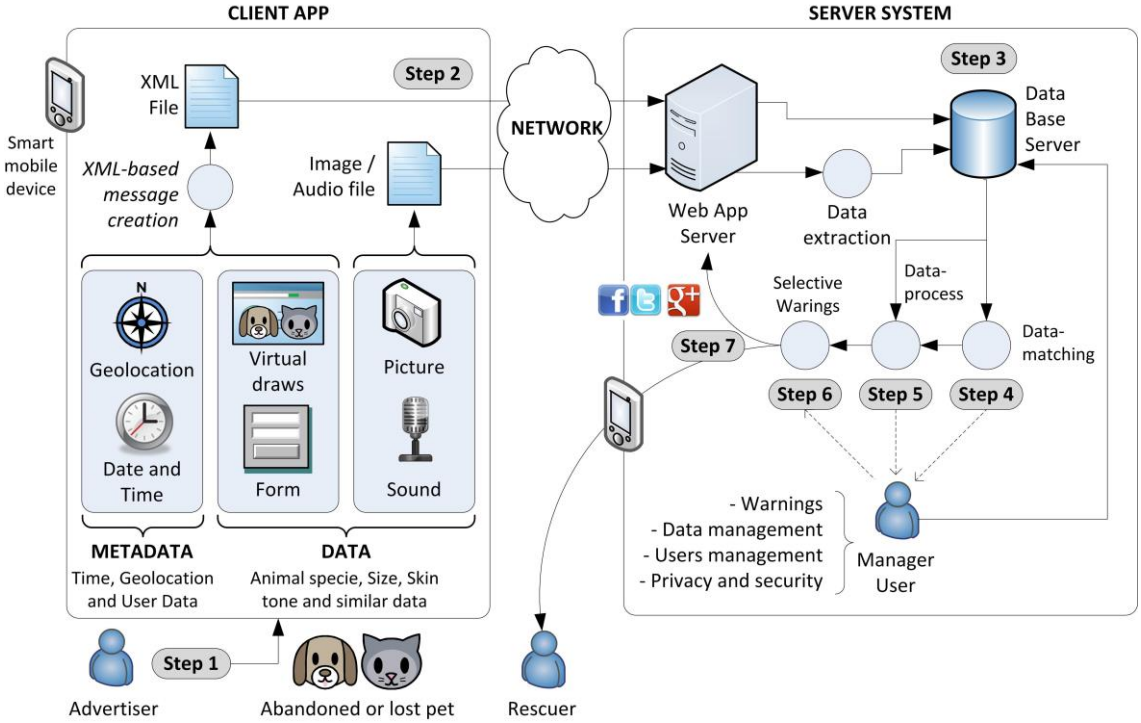

Figura 13 – Perspectiva del producto

3.2.2. Funciones del producto (de la aplicación)

La aplicación móvil permitirá a los usuarios:

- Iniciar sesión en la aplicación.
- Enviar un aviso pudiendo incluir información del animal, una imagen de este, una grabación de audio describiendo al animal y metadatos (fecha, hora, localización del dispositivo, y nombre de usuario).
- Configurar la aplicación.

Esto se muestra gráficamente en el diagrama de casos de uso de la figura 14.

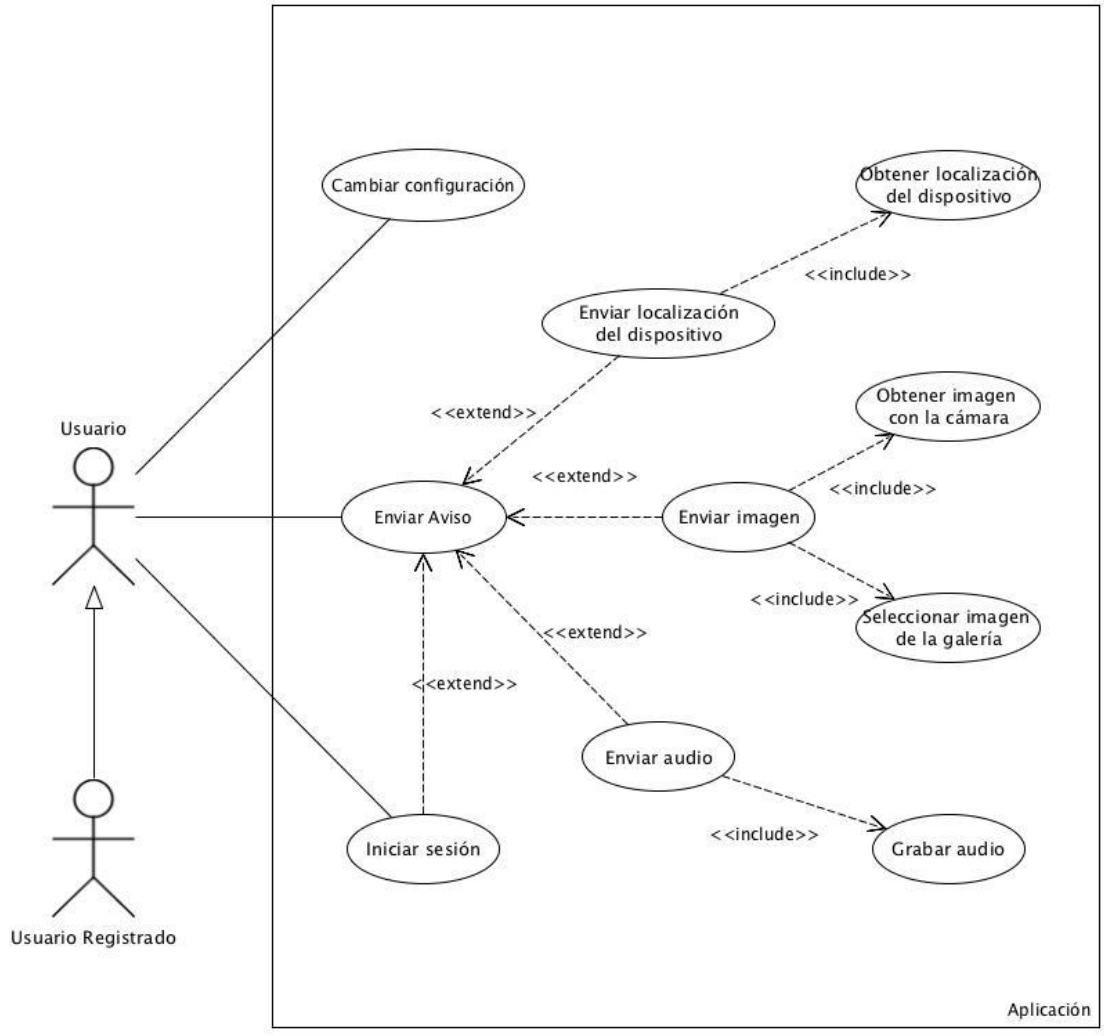

Figura 14 – Diagrama de casos de uso para la aplicación

3.2.3. Características de los usuarios

Para el uso de la aplicación ningún requisito especial es necesario en el usuario. Cualquiera dispuesto a ayudar puede utilizar la aplicación. Este perfil de usuario está asociado al actor *Advertiser* en la figura 13.

3.2.4. Restricciones

- La comunicación con el servidor se realizará mediante el protocolo HTTP.
- La aplicación mandará peticiones al servidor mediante la operación GET indicando los detalles de la petición como parámetros en la URL o mediante la operación POST añadiendo un documento JSON o XML en el cuerpo de esta.
- El servidor responderá a las peticiones enviando al dispositivo documentos JSON o XML.
- El dispositivo sobre el que corra la aplicación deberá de ser Android y dispondrá de Internet.

3.2.5. Suposiciones y dependencias

- Se asumirá que existe un servidor implementando mediante servicios web una interfaz que permita que el servidor realice las tareas que se le atribuyen anteriormente.
- Que esta interfaz sea conocida por el desarrollador de la aplicación.
- Que esta interfaz sea accesible mediante una IP y puertos fijos también conocidos por el desarrollador de la aplicación.
- Que la interfaz cumplirá las tres primeras restricciones indicados en la subsección de restricciones.
- Se cumplirán en general las restricciones indicadas anteriormente.
- El usuario de la aplicación es capaz de utilizar un dispositivo Android.

3.3. Requisitos específicos

En esta sección se detalla en profundidad los requisitos que la aplicación objetivo tendrá que cumplir cara a, por un lado, otros sistemas con los que se relaciona, y por otro los usuarios que utilicen la aplicación. En el segundo caso refiriéndose a lo que la aplicación hará (requisitos funcionales), además de cómo lo hará (no funcionales). Más tarde en el capítulo cuarto de este trabajo, se presentará el diseño para la aplicación siguiendo las directrices marcadas por esta especificación.

3.3.1. Interfaces externas

- Interfaz de usuario: deberá ser fácil de usar, agradable a la vista e intuitiva. Es importante que el usuario realice el mínimo número de interacciones posible. Se utilizará la pantalla como principal forma de interacción, aunque no quedará limitada a esta, si utilizando otros medios de interacción se facilita la tarea al usuario.

- Interfaz hardware: la aplicación será capaz de interactuar con el sistema operativo de Android para acceder al hardware necesario para su correcto funcionamiento. El dispositivo deberá contar con el hardware necesario para acceder a Internet, grabar archivos de audio, tomar imágenes, y obtener la localización del dispositivo.

- Interfaz de comunicaciones: la aplicación ha de poder de intercambiar mensajes con el servidor mediante servicios Web REST.

3.3.2. Requisitos funcionales

Identificador	RF01
Nombre	Iniciar sesión
Descripción	La aplicación permitirá autenticarse al usuario mediante el envío de las credenciales de este al servidor, el cual se encargará en último momento de la autenticación y responderá a la aplicación afirmativamente si las credenciales son correctas o negativamente si no lo son. La aplicación ha de ser capaz de diferenciar entre los dos casos
Precondiciones	Que el usuario tenga credenciales (nombre de usuario y contraseña) para iniciar sesión, y que no esté autenticado ya
Postcondiciones	Los avisos que se envíen una vez autenticados estarán asociados al usuario

Tabla 6 – Requisito funcional: Iniciar sesión

Identificador	RF02
Nombre	Obtener imagen con la cámara
Descripción	La aplicación permitirá tomar imágenes con la cámara del dispositivo
Precondiciones	Ninguna
Postcondiciones	La imagen se guardará y quedará accesible para su posterior uso en la aplicación

Tabla 7 – Requisito funcional: Obtener imagen con la cámara

Identificador	RF03
Nombre	Seleccionar imagen de la galería
Descripción	El usuario podrá seleccionar una imagen existente en el dispositivo
Precondiciones	Ninguna
Postcondiciones	La imagen quedará accesible para su posterior uso en la aplicación

Tabla 8 – Requisito funcional: Seleccionar imagen de la galería

Identificador	RF04
Nombre	Enviar imagen
Descripción	La aplicación permitirá enviar la imagen anteriormente seleccionada al servidor
Precondiciones	RF03 o RF04
Postcondiciones	La imagen llegará integra al servidor

Tabla 9 – Requisito funcional: Enviar imagen

Identificador	RF05
Nombre	Grabar audio
Descripción	La aplicación podrá grabar un archivo de audio mediante la aplicación
Precondiciones	Ninguna
Postcondiciones	El archivo de audio podrá ser accedido por la aplicación para su uso

Tabla 10 – Requisito funcional: Grabar audio

Identificador	RF06
Nombre	Enviar audio
Descripción	La aplicación permitirá el envío de un archivo de audio al servidor
Precondiciones	RF05
Postcondiciones	El archivo de audio llegará integro al servidor

Tabla 11 – Requisito funcional: Enviar audio

Identificador	RF07
Nombre	Obtener localización del dispositivo
Descripción	La aplicación ha de ser capaz de obtener las coordenadas del lugar donde el dispositivo se encuentra, existiendo consentimiento por parte del usuario. Además la aplicación tendrá que poder mostrar la localización obtenida al usuario mediante un mapa interactivo.
Precondiciones	Que el usuario haya dado su consentimiento a la aplicación para obtener la localización
Postcondiciones	La localización será obtenida y estará lista para su envío

Tabla 12 – Requisito funcional: Obtener localización del dispositivo

Identificador	RF08
Nombre	Enviar localización del dispositivo
Descripción	La aplicación permitirá el envío de la localización obtenida al servidor
Precondiciones	RF07
Postcondiciones	La localización llegará al servidor

Tabla 13 – Requisito funcional: Enviar localización del dispositivo

Identificador	RF09
Nombre	Enviar aviso
Descripción	La aplicación ha de ofrecer la capacidad de enviar un aviso ya sea de un animal perdido o de un animal encontrado. Este aviso ha de contener los rasgos del animal, la localización del dispositivo, la fecha y la hora (pudiéndose obtener también en el servidor) y la información del usuario si este se encuentra autenticado. La información ha de poder introducirse mediante un formulario, mediante una imagen y mediante una grabación de audio. En cuanto los rasgos del animal, se tienen que poder definir la especie, el color del animal, la talla y su temperamento.
Precondiciones	RF01, RF04, RF06, RF08 (opcionales)
Postcondiciones	El envío del aviso independientemente de la información que incluya llegará al servidor

Tabla 14 – Requisito funcional: Enviar aviso

Identificador	RF10
Nombre	Cambiar configuración
Descripción	El usuario será capaz de configurar la aplicación de diferentes maneras
Precondiciones	Ninguna
Postcondiciones	Los cambios en la configuración realizados se mantendrán hasta que se vuelvan a cambiar, o se borren los datos de la aplicación. Los parámetros configurables no se especifican, siendo la única restricción que exista una motivo para su elección

Tabla 15 – Requisito funcional: Cambiar configuración

3.3.3. Requisitos no funcionales

Identificador	RNF01
Tipo	Fiabilidad
Nombre	Gestión de errores
Descripción	La aplicación ha de poder recuperarse de los errores siempre y evitar que se la aplicación se cierre inesperadamente siempre que sea posible, ofreciendo al usuario mensajes de error e información de lo qué ha ocurrido

Tabla 16 – Requisito no funcional: Gestión de errores

Identificador	RNF02
Tipo	Fiabilidad
Nombre	Evitar pérdidas de información
Descripción	La *app* tendrá que ser capaz de mantener la información introducida por el usuario y poder recuperarse de posibles cambios sin perder esta. Estos posibles cambios incluyen entre otros el cambio de orientación del dispositivo, o llamadas entrantes durante el uso de la aplicación

Tabla 17 – Requisito no funcional: Evitar pérdidas de información

Identificador	RNF03
Tipo	Portabilidad
Nombre	Adaptación a diferentes dispositivos
Descripción	La *app* ha de poder funcionar correctamente en diferentes dispositivos Android, abarcando el máximo número de versiones del sistema operativo. Las interfaces además han de poder adaptarse a diferentes tamaños de pantalla

Tabla 18 – Requisito no funcional: Adaptación a diferentes dispositivos

Identificador	RNF04
Tipo	Integridad
Nombre	Integridad de la información enviada
Descripción	La información enviada al servidor ha de llegar correctamente al servidor, teniendo especial cuidado en los archivos multimedia, de manera que el servidor sea capaz de procesarlos

Tabla 19 – Requisito no funcional: Integridad de la información enviada

4. Diseño del sistema

4.1. Introducción

En el capítulo anterior se ha establecido cómo ha de ser la aplicación objetivo de este trabajo. Se ha indicado en forma de requisitos funcionales y no funcionales qué es lo que tiene que hacer y de qué modo ha de hacerlo. A partir de este punto, el siguiente paso es diseñar la aplicación para poder así formar una base sobre la que implementar la aplicación.

En este capítulo se mostrará el diseño realizado para la aplicación. En el apartado segundo se mostrará el diseño global para el sistema, estableciendo una serie de componentes que formarán la aplicación y mostrando las interacciones entre estos.

Seguidamente en el apartado tercero se presentará un diseño en detalle para el cual se ha adoptado un enfoque por capas, la de presentación, la de negocio y la de persistencia. Cada una de estas capas se ha diseñado por separado mediante diversos diagramas que serán explicados en detalle.

4.2. Diseño global

La aplicación Android de este proyecto, para poder funcionar tendrá que ser capaz de interactuar con diferentes componentes externos a la aplicación. Algunos formarán parte del sistema mientras que otros serán de terceras partes, principalmente de Google.

El principal componente externo con el que interactuará la aplicación será un servidor propio que se encargará de recoger y procesar toda la información que el usuario introduzca a la *app* mediante su interfaz gráfica. En este momento se define un componente interno de la app que se encargará de esta interacción al que se ha llamado **DataManager**, cuya función principal será la de establecer la comunicación con el servidor y gestionar los posibles errores.

Otro componente que se usará exhaustivamente en la aplicación será el hardware del dispositivo. Para ello se confiará en la API de Android la cual se encargará de mapear nuestras necesidades en llamadas al sistema operativo. Algunas características hardware usadas son el sistema de almacenamiento, la cámara, el micrófono y fundamentalmente el acceso a Internet. El acceso al hardware por lo general es trivial y no se ha definido ningún componente especifico excepto para la grabación de audio que requiere una mayor configuración y gestión de errores (**GrabadorAudio**).

También se ha decidido añadir un componente que centralice la configuración de la *app* de manera que en cualquier momento se encuentre disponible el valor actual de cualquier parámetro de configuración. Cuando cualquier otro componente necesite escribir o leer un valor de configuración accederá a través de este nuevo componente al que se ha llamado **ConfigurationManager**. Este componente en adición se encargará de que estos valores se escriban en un archivo de preferencias en el dispositivo para mantener los valores después de que la *app* se cierre.

Además utilizaremos dos APIs de Google, una para obtener la localización del dispositivo, y otra para mostrarla en un mapa al usuario. La primera se trata de Google Play Services, que proporciona una serie de herramientas al desarrollador para obtener la localización del dispositivo, así como para gestionar y tratar de recuperarse de posibles errores en este proceso. Todas las llamadas a la API de Google Play Services se encontrarán en un componente al que se ha llamado **GooglePlayManager**.

La segunda API de Google se trata de la API de Google Maps. Esta nos ofrece un componente visual con el mapa mundial de Google Maps. Sobre este componente se marcará la situación actual del dispositivo para mostrarla al usuario. Dado que toda la funcionalidad de esta API se centra en el componente visual, las llamadas a esta API se harán todas desde la interfaz. Todos estos componentes descritos, y las interacciones entre ellos se muestran en el diagrama de componentes en la figura 15.

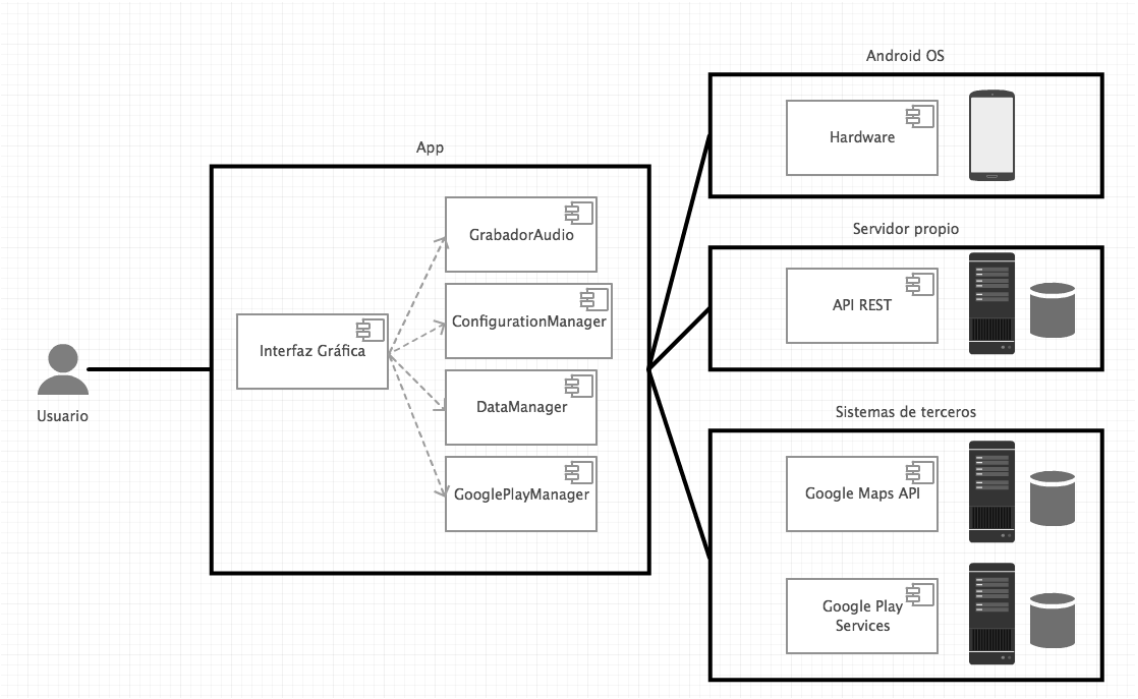

Figura 15 – Diagrama de componentes de la aplicación

4.3. Diseño por capas

Como se ha indicado al principio de este capítulo, se ha optado por un diseño basado en capas. En este caso, la capa de interfaz será la encargada de interactuar con el usuario, la segunda, la capa de negocio incluirá la lógica interna de la aplicación y la capa de persistencia se encargará de interactuar directamente con los diferentes orígenes de datos utilizados en la aplicación.

4.3.1. Presentación

Para diseñar la capa de presentación, se ha decidido crear una relación de prototipos para las diferentes interfaces de la *app* o *mock-ups*. Estos servirán de guía a la hora de definir en las interfaces dentro de la aplicación, aunque es posible que no se puedan seguir al pie de la letra pudiendo aparecer diversas limitaciones a la hora de adaptarse al entorno Android. La figura 16 muestra estos *mock-ups*.

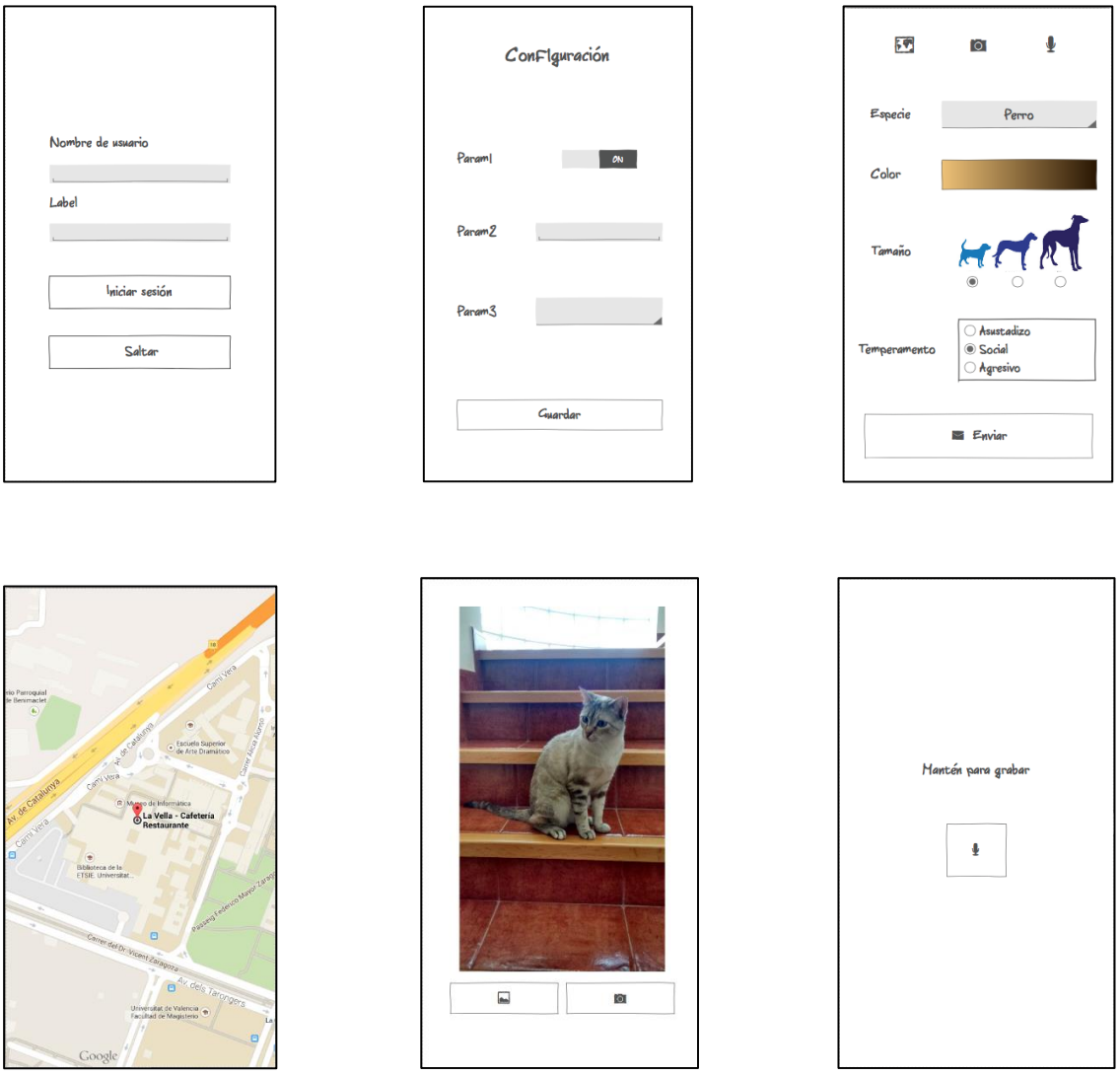

Figura 16 – Prototipos o *Mock-Ups* diseñados para la aplicación

El primer prototipo (arriba a la izquierda) muestra una pantalla en la que el usuario puede autenticarse o continuar usando la aplicación de manera anónima. Esta será la pantalla con la que arranque la aplicación. Seguidamente arriba a la derecha se muestra cómo será la interfaz para configurar la *app*. Por último las últimas cuatro pantallas serán las utilizadas para introducir y mostrar la información relativa al animal.

La de arriba a la derecha permite introducir los rasgos del animal de manera manual. Estos rasgos se han definido anteriormente en el capítulo de especificación de requisitos. Esta pantalla se modificará dinámicamente según se vayan modificando los diferentes parámetros para mostrar los valores introducidos de forma visual. Esta pantalla ofrecerá acceso a las otras tres:

La de abajo a la izquierda mostrará la ubicación obtenida por el dispositivo si hubiese una. La de abajo en el centro permitirá introducir la imagen del animal. La última, abajo a la derecha, será la que permita al usuario describir al animal mediante la grabación de un archivo de audio.

4.3.2. Negocio

Seguidamente se presenta el diseño hecho para la capa de negocio. Cabe destacar que este diseño se ha separado en dos visiones, la visión estática, y la dinámica. La primera de ellas en forma de diagrama de clases muestra las diferentes entidades que forman parte de la aplicación con sus propiedades y métodos. Por otro lado la segunda mediante diagramas de secuencia se refleja la principal lógica interna de la aplicación.

Visión estática

Se han definido dos entidades para la aplicación Aviso y Usuario. En el diagrama de clases de la figura 17 se muestra la relación entre estas clases, sus propiedades y sus métodos de persistencia. A la hora de implementar, cada entidad se dividirá en dos clases. Las propiedades y métodos constructores, consultores y modificadores formarán parte de la clase de negocio de la entidad **[24]**, y sus métodos de persistencia de su Objeto de Acceso a Datos (clase DAO) **[25]**.

Los métodos constructores, modificadores, consultores, y métodos internos se han omitido por su trivialidad y en la implementación se podrán definir los que sean necesarios. Además los campos de tipo, especie, talla, color y temperamento codificarán en forma de entero los posibles valores para estos campos. La codificación utilizada se indica en la tabla 20.

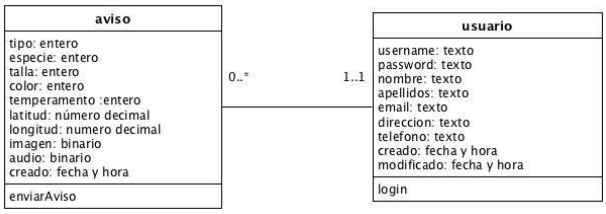

Figura 17 – Diagrama de clases de la aplicación

Campo	Código entero	Significado
Tipo	0	Perdido
	1	Encontrado
Especie	0	Perro
	1	Gato
Talla	0	Pequeño
	1	Mediano
	2	Grande
Color	0	Claro
	1	Punto medio
	2	Oscuro
Temperamento	0	Tímido
	1	Social
	2	Agresivo

Tabla 20 – Significado de los posibles valores enteros para cada rasgo del animal

Visión dinámica

Para la visión dinámica de la capa de negocio se han diseñado diferentes diagramas de secuencia. En estos se muestra la mayoría de la lógica interna de la aplicación aunque con el objetivo de mantener los diagramas simples y legibles solo se han incluido los casos básicos en los que no hay ningún contratiempo. A la hora de implementar habrá que contemplar más casos y gestionar los posibles problemas y errores que puedan surgir.

Cabe destacar que al terminar cada interacción con la aplicación en los diagramas se manda un mensaje a la interfaz llamado *render*. Este es un método ficticio que se ha creado para referirse a cualquier tipo de retroalimentación sobre los cambios realizados que haya que mostrar por la interfaz, desde posibles mensajes informativos o de error, hasta pasar a una nueva pantalla. Cuando a este método se le pasa un parámetro, se indica que la retroalimentación dependerá del valor de este parámetro.

El primer diagrama de secuencia en la figura 18 se muestra el comportamiento interno de la aplicación para el *log in* o inicio de sesión. Seguidamente en las figuras 19 a 22 se muestran los diagramas de secuencia de las diferentes versiones de envío de aviso que se podrán hacer en la aplicación. Estos diagramas de secuencia se podrán combinar en función de la diferente información que haya introducido el usuario o la propia aplicación en el caso de la localización.

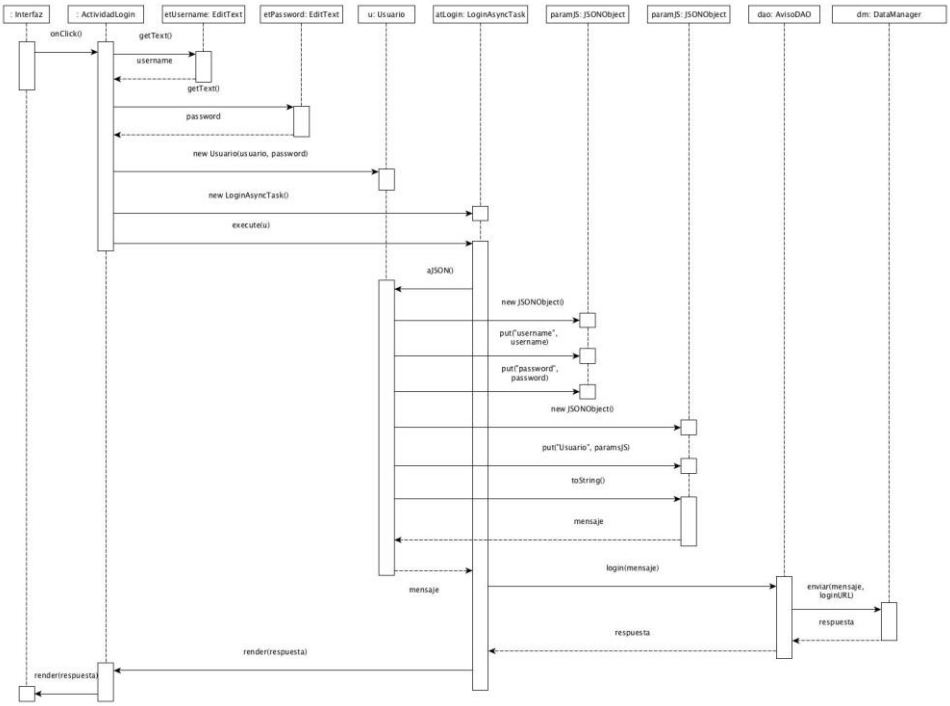

Figura 18 – Diagrama de secuencia: Inicio de sesión

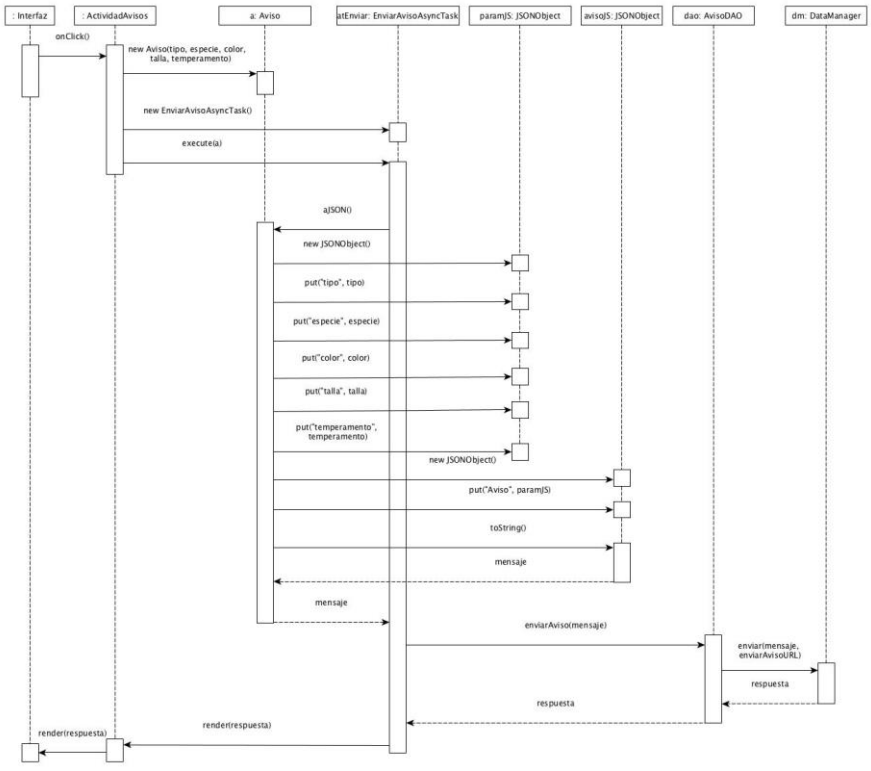

Figura 19 – Diagrama de secuencia: Enviar aviso

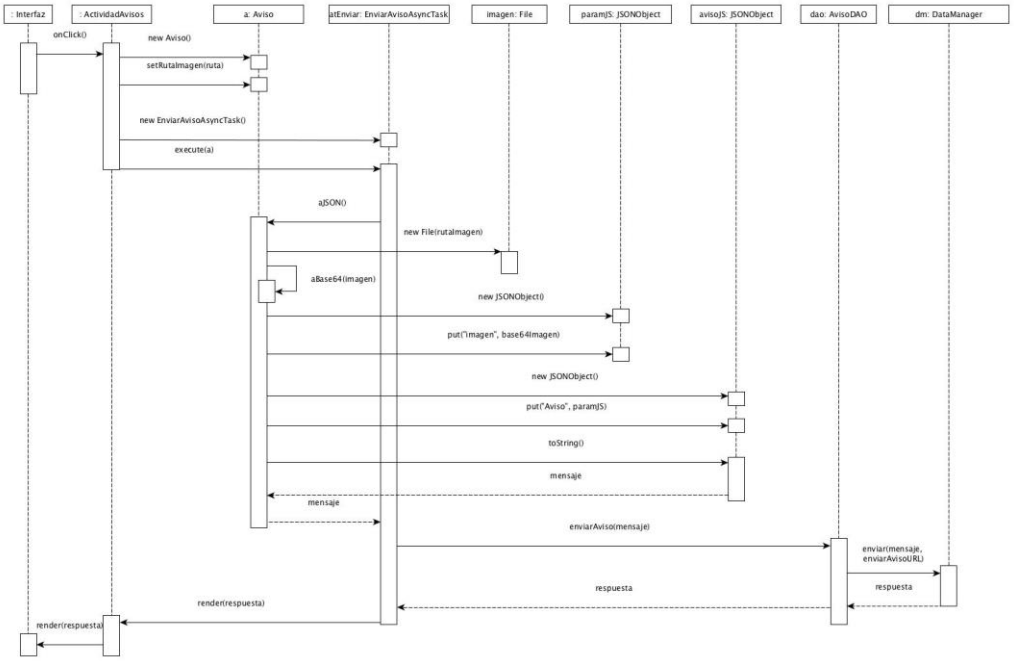

Figura 20 – Diagrama de secuencia: Enviar imagen

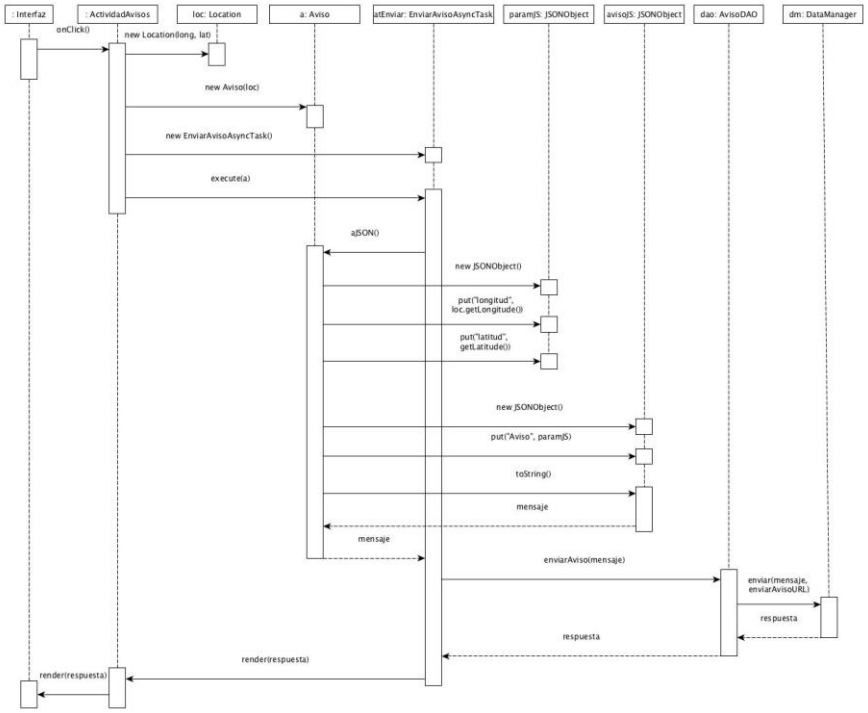

Figura 21 – Diagrama de secuencia: Enviar localización del dispositivo

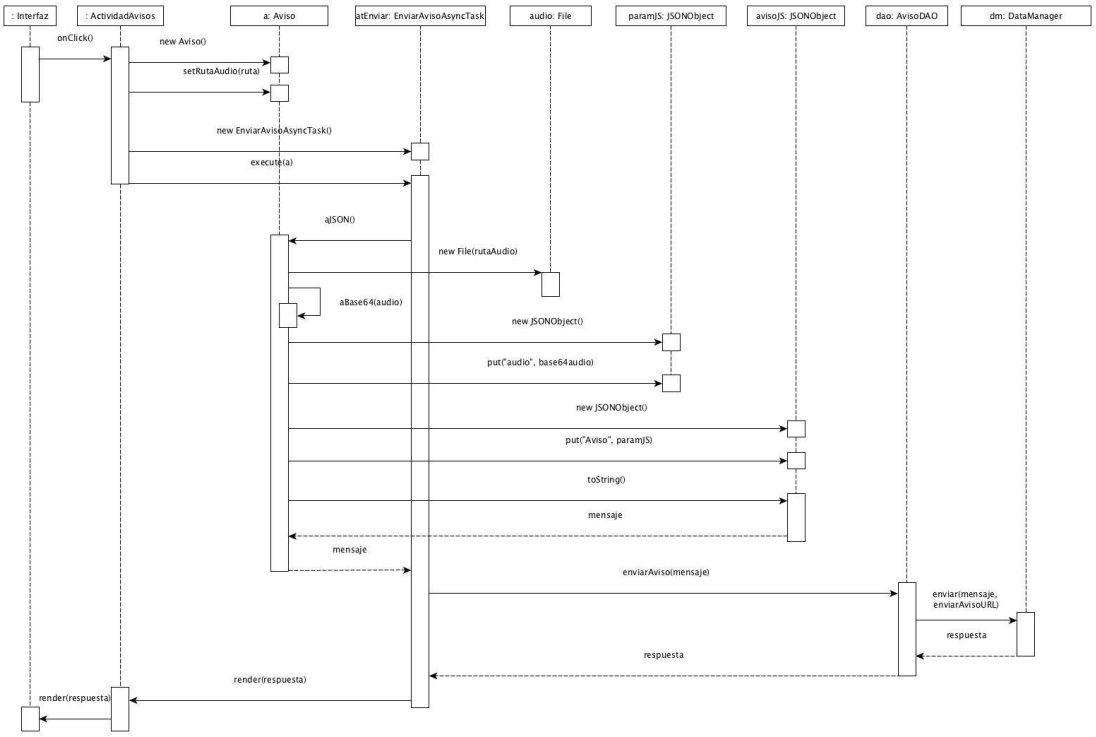

Figura 22 – Diagrama de secuencia: Enviar grabación de audio

Por último, se presentan el diagrama de secuencia para el grabador de audio (Figura 23) y la gestión de la configuración (Figura 24).

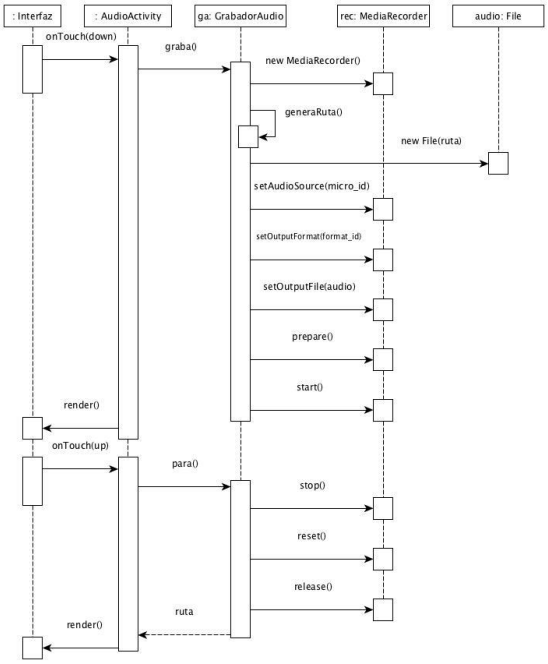

Figura 23 – Diagrama de secuencia: Grabar audio

El diagrama de secuencia para la gestión de configuración es un diagrama genérico que representa lo que ocurre al modificar un parámetro de configuración. Esta genericidad aparece en **paramView** que se refiere al componente visual específico para modificar ese parámetro concreto, **onViewEvent()** es el evento que se crea al modificar el valor, **setParam()** y **putType("conf_param", valor)** donde "type" se refiere al tipo de datos del parámetro y "param" el nombre del parámetro. Este diagrama se muestra en la figura 24.

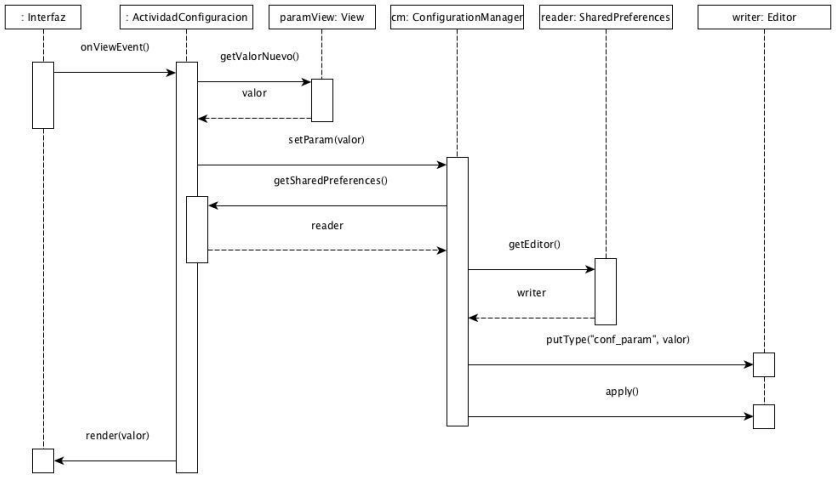

Figura 24 – Diagrama de secuencia: Cambiar parámetro configurable

Además conviene recordar que los parámetros configurables no quedaban especificados en el capítulo anterior. Por ello conviene definirlos antes de seguir adelante. En la tabla 21 se muestran los parámetros elegidos, su tipo, sus posibles valores y el motivo por los que se han escogido.

Parámetro	Tipo	Posibles valores	Motivo
Reducir tamaño de imágenes enviadas	Booleano	Si/No (True/False)	Pueden existir usuarios que no quieran hacer uso excesivo de su plan de datos por lo que se permitirá reducir las imágenes antes de ser enviadas
URL del servidor	URL	Infinitos	Este parámetro se añade para facilitar la tarea del desarrollo y testeo de la aplicación. No se mostrará al usuario final
Color de la interfaz	Color	Naranja, rojo, azul y violeta	Esto permitirá al usuario elegir el color de la interfaz permitiéndole así que se sienta más cómodo con esta.

Tabla 21 – Parámetros configurables en la aplicación

4.3.3. Persistencia

Por último para la capa de persistencia se ha hecho una propuesta de interfaz de servicios Web REST. Dado que esta interfaz ha de estar aceptada por el equipo a cargo del desarrollo del servidor, este diseño es el más proclive a sufrir cambios en el futuro. La interfaz propuesta en este trabajo se presenta en la tabla 22. Se ha utilizado JSON para los mensajes al estar más familiarizado con este lenguaje.

Servicio Web	Descripción	Entrada (JSON)	Salida (JSON)

Conexión	El servidor web recuperará el registro asociado al usuario y devolverá *true* si la contraseña es igual. Si no lo es o si no se encuentra el usuario se devuelve *false*.	{"Usuario":
		{
		"username": *string*,
		"password": *string*
		}
		}
			{"resultado": *boolean*}

Enviar Aviso	El servidor guardará el aviso siendo todos los campos opcionales. Si no hay ningún error	{"Aviso":
		{
		"tipo": *integer*,
		"especie": *integer*,
		"talla": *integer*,
		"color": *integer*,
		"temperamento": *integer*,
		"latitud": *double*,
		"longitud": *double*,
		"audio": *base64-blob*,
		"imagen": *base64-blob*,
		"usuario": *Usuario*
		}
		}
			{"resultado": *boolean*}

Tabla 22 – Definición de los servicios web REST y de la estructura de los mensajes JSON

4.4. Conclusiones

En este capítulo se ha mostrado el trabajo realizado en cuanto al diseño de la aplicación se refiere. Este diseño, primero introducido de forma global, después desglosado según el enfoque clásico de capas de presentación, negocio y persistencia establecen una base y una guía sobre la cual implementar la aplicación de manera consistente.

5. Implementación y Pruebas

5.1. Introducción

Una vez presentado el diseño para la aplicación Android objetivo de este proyecto, el siguiente paso es implementar la aplicación a partir de este diseño. En el presente capítulo se tratarán todas las cuestiones relacionadas con la implementación realizada.

En el segundo apartado se mostrará cómo se ha implementado la app Android. Se tratará en primer lugar el código Java que da cuerpo a la aplicación para pasar después a introducir lo que en este trabajo se ha llamado recursos estáticos de la aplicación que no es más que diferentes elementos predefinidos generalmente en ficheros XML y a los que la aplicación tiene acceso.

En adición se ha implementado un servidor en PHP con la interfaz de servicios Web propuesta en el capítulo de diseño y que hará de soporte para la aplicación Android. Este servidor se tratará en el tercer apartado de este capítulo.

Por último en el cuarto capítulo la aplicación será puesta a prueba con motivo de comprobar que la aplicación funciona correctamente y además se adecua a los requisitos establecidos para la aplicación.

5.2. Aplicación Android

La implementación de la *app* se ha estructurado en diferentes paquetes Java de manera que el código está mejor organizado y es más fácil de encontrar en todo momento el archivo que sea necesario. Como todo proyecto Android, este está separado en dos directorios principales: **src**, el cual contiene el código fuente en java de la aplicación y **res**, con ficheros XML que definen las interfaces de la *app*, estructura de los menús, valores enteros y textuales estáticos, así como estilos utilizados en las interfaces.

5.2.1. Código

El directorio **src** es el que alberga la estructura en paquetes del proyecto. Todos y cada uno de estos paquetes cuelgan de es.upv.inf.animales, por lo que cuando hablemos de uno de estos paquetes, por simplicidad no se dirá, por ejemplo “es.upv.inf.animales.usuarios”, si no “el paquete **usuarios**” directamente. En la tabla 23 se muestran los diferentes paquetes definidos y una breve descripción de su contenido y en la figura 25 se puede ver todo el contenido de **src**.

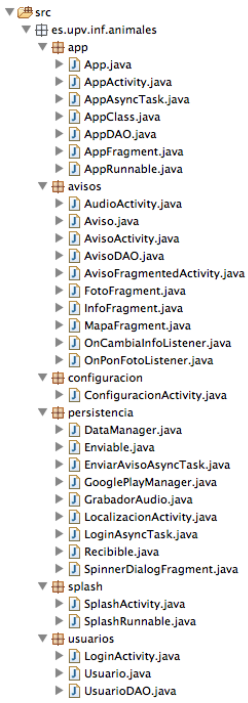

Figura 25 – Carpeta src

Paquete	Contenido
app	Este paquete contiene clases utilizadas en toda la *app* en general.
persistencia	Contiene clases que implementan la persistencia de la aplicación, incluyendo la comunicación con el servidor y con Google Play Services para la localización. También se incluye el diálogo de espera que se muestra durante los envíos y otras clases auxiliares.
avisos	En este paquete se incluyen las clases para la gestión de los avisos. Incluye su clase de la lógica de negocio, su clase DAO y la implementación de las interfaces gráficas.
configuracion	Aquí están las clases para la configuración de usuario de la aplicación.
splash	Implementación de la pantalla de arranque de la *app* conocida en inglés como pantalla *splash*.(Véase **[26]**)
usuarios	Contiene las clases que implementan la funcionalidad relativa a los usuarios. Incluye su clase de la lógica de negocio, su clase DAO y la implementación de las interfaces gráficas.

Tabla 23 – Relación de paquetes en los que se organiza la aplicación

A diferencia de en el capítulo de diseño, que se dividía la aplicación en tres capas, la de presentación, la de negocio y la de persistencia, en la implementación se puede apreciar que se ha hecho otra división diferente. A pesar de esto, esta organización no es arbitraria. Cada uno de estos paquetes se encarga de un aspecto de la aplicación que es independiente de las demás.

Por ejemplo la gestión de usuarios, en el paquete **usuarios** habrá que implementar los detalles de los mensajes específicos a la gestión de usuarios pero no habrá que volver a implementar el envío mediante Post de HTTP. Para esto último, se delegará en el paquete de **persistencia**, que contendrá los detalles de este protocolo y que son comunes en todos los envíos.

App

En el paquete **app** se encuentra la clase **App**, que sobrescribe el comportamiento por defecto de la aplicación. Para ello, se extiende de la clase de Android **Application** y se indica en el manifiesto de la aplicación que esta clase se encargará de ello.

En este caso, la clase App tiene dos funciones principales. La primera es proporcionar un objeto de la clase **Resources** a todas las demás clases de la aplicación que no sean capaces de conseguirlo por sí mismas. Este objeto está disponible para todas las clases que hereden de **Context** mediante el método **getResources()** y permite el acceso a los recursos estáticos de la aplicación albergados en ficheros XML en la carpeta **res**.

Su segunda función es la que en el capítulo de diseño se otorgaba al **ConfigurationManager**. Dado que esta clase representa la propia aplicación y su ciclo de vida se extiende a lo largo de toda la ejecución, parece conveniente que sea esta clase la que se encargue de gestionar de manera centralizada los parámetros de configuración.

Además de la clase **App**, se han definido en este paquete diferentes clases abstractas que heredan de las diferentes clases Android utilizadas. Estas clases que empiezan todas con la palabra App, implementan comportamientos que serán comunes en sus clases descendientes además de tener un objeto **Resources** de la aplicación o propio. Estas clases se describen en la tabla 24.

Persistencia

El paquete **persistencia** es uno de los más importantes de la aplicación. Incluye la implementación de tres componentes antes mencionados en el capítulo de diseño: **DataManager**, **GooglePlayManager** y **GrabadorAudio**. Estos componentes tratan con diversos orígenes de datos y son los encargados de obtener información de ellos o introducir nueva en el caso del **DataManager**. Además todos estos componentes son capaces de recuperarse de los posibles errores que puedan surgir en tiempo de ejecución.

Clase	Descripción
AppActivity	Las clases que hereden de esta implementarán la interfaz de usuario. Entre otras cosas, se encargan de aplicar al fondo de la interfaz el color seleccionado en la configuración por el usuario.
AppFragment	Esta clase define los fragmentos (clase **Fragment** de Android) de la aplicación. Un fragmento se trata de una subpantalla y siempre pertenece a una **Activity**, que puede cambiar durante el ciclo de vida del fragmento. En esta clase genérica se mantiene una referencia siempre actualizada de la actividad que alberga el fragmento.
AppAsyncTask	Clase genérica para el envío de mensajes. Cuando empieza su ejecución muestra un diálogo de espera, y lo oculta al terminar. Su ejecución es asíncrona, así que la **Activity** que la llamó puede no existir al terminar. Este problema se arregla en esta clase.
AppRunnable	Añade ciertos comportamientos a la clase **Runnable** de Java.
AppClass	Esta clase hereda de **Object** directamente. Permite el acceso a los recursos a cualquier clase nueva que herede de **Object**.
AppDAO	Esta clase genérica hereda de **AppClass**. Todos los objetos DAO heredarán de esta clase, por lo tanto, se encarga de inicializar si hiciese falta el **DataManager** del paquete persistencia, que será utilizado por cualquier clase DAO. Todas las clases DAO implementan el patrón *singleton*, para que solo haya una instancia de ellos en todo momento.

Tabla 24 – Clases genéricas del paquete app

El primer componente, albergado en la clase **DataManager**, implementa mediante el método **enviaJSON** el envío y recepción de mensajes mediante el método Post del protocolo HTTP. A este método se le pasa el mensaje JSON a enviar, y la ruta relativa del servicio web a utilizar. En caso de correcto funcionamiento devuelve la respuesta, en caso contrario, devuelve *null* y el error será presentado de forma agradable al usuario en capas superiores. Esta clase al igual que las clases DAO utiliza el patrón *singleton* (Véase **[27]**) para que haya una única instancia de la clase en todo momento.

Además se definen dos interfaces para traducir los objetos de negocio a JSON y viceversa. Estas interfaces son **Enviable** con el método **aJSON** y **Recibible** con el método **desdeJSON**. Si un objeto de negocio puede ser enviado, implementará **Enviable**, y si puede ser recibido, implementará **Recibible**.

Así mismo en el paquete también se encuentran las implementaciones de los envíos asíncronos en sí (clases que heredan de **AppAsyncTask**) que se encargan de coordinar las clases de negocio, las clases DAO y el **DataManager** de la manera adecuada según el envío a realizar y al terminar, hacer los cambios pertinentes sobre la interfaz mediante la actividad Android activa en ese momento.

Por último, la pantalla de espera que se muestra al principio de todas las **AppAsyncTask** y se oculta al final de estas, está en este paquete también. Se trata de un fragmento Android cuya interfaz está formada únicamente por una circunferencia abierta giratoria sobre un fondo transparente, que permite ver por detrás la pantalla que estuviese activa justo antes. Esta pantalla de espera se puede ver en la figura 26.

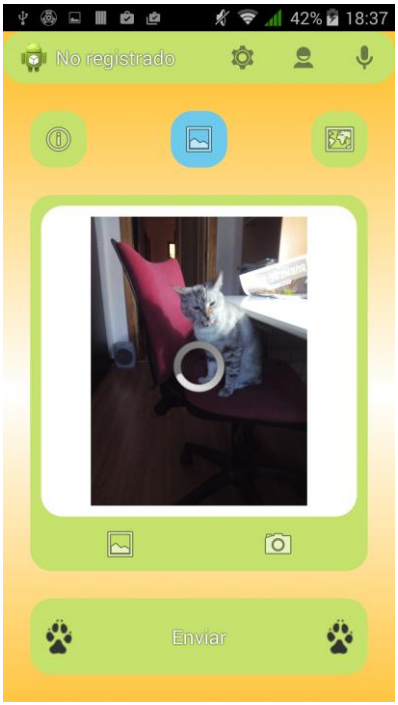

Figura 26 – Pantalla de espera

Es importante mencionar que dado que el protocolo HTTP codifica sus mensajes mediante caracteres, los archivos binarios (las imágenes y archivos de audio) no se pueden enviar tal cual. Por lo tanto estos archivos son codificados mediante el algoritmo Base64 que convierte los bytes de los archivos a caracteres que se pueden enviar. Este algoritmo codifica cada grupo de tres bytes a cuatro bytes de manera que el archivo enviado será un tercio más grande que el archivo original.

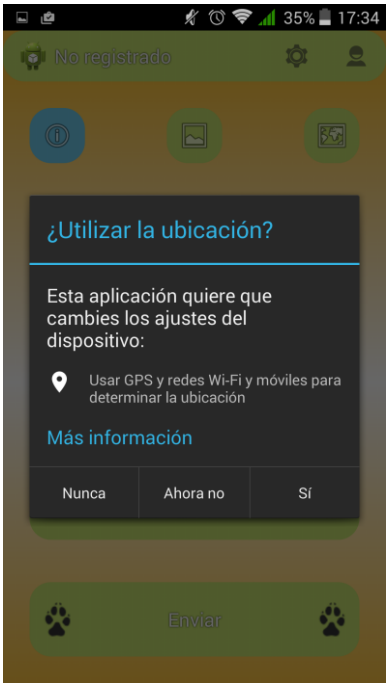

Figura 27 – Diálogo de configuración de la localización

El componente **GooglePlayManager**, en la clase con el mismo nombre, se encarga de la obtención de la localización del dispositivo. Para ello cuenta con una segunda clase, **LocalizacionActivity**, una nueva clase abstracta que hereda de **AppActivity** y cuya función es la de utilizar correctamente el **GooglePlayManager** a lo largo del ciclo de vida de la actividad.

Cualquier clase que herede de **LocalizacionActivity**, dispondrá siempre que sea posible de la localización, sin tener que implementar absolutamente nada. Esta clase también se encarga de mantener la localización si el sistema operativo destruye y vuelve a crear la actividad, algo que ocurre por ejemplo cuando el usuario cambia la orientación del dispositivo.

Estas dos últimas clases, gestionan una gran cantidad de posibles errores que se pueden dar durante este proceso como puede ser que el usuario no permita la localización, que no disponga de Google Play Services o no tenga conexión a internet. Si el usuario no permite la localización se le mostrará un diálogo que le ofrecerá la opción de activarla o no (figura 27). En caso de que el error no sea recuperable, se detiene todo el proceso, y se le indica al usuario que no se puede adquirir su localización.

Por último, la clase **GrabadorAudio** se encarga de interactuar con el hardware del dispositivo para permitir al usuario la grabación de notas de voz para su posterior envío al servidor. Esta clase expone métodos que permiten a una actividad Android crear el archivo de audio en formato MP4 a partir de lo que el micrófono sea capaz de capturar.

Dado que los archivos enviados por la aplicación se codifican en Base64, y este algoritmo codifica cada grupo de tres bytes en cuatro bytes, el tamaño máximo de la nota de voz se calcula a partir del espacio máximo en la base de datos para el archivo (N bytes) según la formula (N/4) * 3, y cuando este tamaño es alcanzado, la grabación se para automáticamente y el audio es guardado para el envío.

Avisos

El paquete avisos contiene todo lo necesario para que un usuario de la aplicación pueda enviar un aviso de un animal perdido o encontrado al servidor. Para este fin se mostraban en el capítulo de diseño cuatro *mock-ups* en la figura 16 Por cuestiones visuales se ha decido organizar de forma diferente estas cuatro pantallas y no seguir al pie de la letra los prototipos diseñados.

Se ha creado una única pantalla (en el fichero **AvisoFragmentedActivity**) que alberga tres secciones o fragmentos accesibles mediante tres botones en la parte superior: **InfoFragment**, con los rasgos del animal, **FotoFragment** para añadir la foto del animal y **MapaFragment** con el mapa de Google Maps para mostrar la localización del dispositivo.

En esta única pantalla existe además un botón para realizar el envío con huellas de animal y una barra de herramientas común para todas las secciones. Esta barra permite regresar a la pantalla de inicio de sesión, acceder a las preferencias de la *app* y acceder a la cuarta pantalla diseñada para introducir información, el de la grabación de audio.

Esta última en un principio iba a ser una sección más y así tener todo accesible en una misma pantalla, pero fue la última en implementarse y por cuestiones de espacio se implementó en una actividad aparte (**AudioActivity**). En cuanto al botón para enviar, al ser pulsado se encarga de reunir toda la información introducida en las tres secciones, y el posible archivo de audio, y proceder al envío.

Por último, cabe destacar además que al modificar los rasgos del animal la interfaz va cambiando para plasmar los cambios realizados. De esta forma, si se cambia la especie, las huellas de la interfaz pasan a ser de la especie seleccionada, si se cambia el color del animal, los controles toman este color, y si se modifica el tamaño, las huellas que aparecen en el botón enviar, adoptan el tamaño seleccionado.

Configuración

En este paquete se encuentra la actividad **ConfiguracionActivity** que es la encargada de dar al usuario la posibilidad de cambiar la configuración de la *app*. Como se ha mencionado antes al hablar del paquete **app**, los valores de las preferencias son gestionados de manera centralizada por la clase App. Los métodos modificadores de esta clase serán llamados desde **ConfiguracionActivity**, para que los nuevos valores se encuentre disponibles en todo momento desde cualquier parte del código, además de hacer al momento los cambios que sean necesarios para adaptarse a los nuevos valores introducidos.

Si en esta pantalla el usuario elige reducir el tamaño de las imágenes, estas serán reducidas a proporción para que quepan en un cuadrado de 500 por 500 de manera que el mayor entre la longitud y la altura, tomará el valor de 500, y el otro se calculará por una regla de tres para que este a proporción. Si en caso contrario se elige no reducir, la imagen será enviada tal y como se incluyó en la aplicación.

Pantalla de Inicio

La pantalla de inicio que se encuentra dentro del paquete **splash**, muestra una interfaz muy simple (en este momento, solo muestra el icono de la aplicación) durante dos segundos y medio al arrancar la aplicación. Se ha hecho de manera que el usuario con cualquier acción sobre el dispositivo, puede ocultarla y pasar a la pantalla de autenticación.

Usuarios

En este paquete se encuentra la pantalla de inicio de sesión, **LoginActivity**, que permite autenticarse ante el servidor o saltarse la autenticación. También alberga las clases DAO y de negocio para los usuarios (**UsuarioDAO** y **Usuario**).

5.2.2. Recursos estáticos e interfaces

Los recursos estáticos que se guardan en la carpeta **res** del proyecto, como su nombre indica son elementos que no cambian. Si no se vuelve a compilar la aplicación, estos mantendrán su valor. Estos están definidos en ficheros XML de Android en su mayoría, aunque también algunos de ellos son imágenes con formato PNG. Concretamente los que se encuentra dentro de los ficheros XML pueden ser números, cadenas de caracteres, definiciones de estilos, de interfaces, y de partes de interfaces. En las figuras 28 y 29 se muestra gran parte de la estructura de **res**.

La gran ventaja de guardar en ficheros separados toda esta información es la adaptabilidad de la *app*: por ejemplo las cadenas de caracteres se almacenan todas en el archivo **values/strings.xml**, para los valores en inglés, o **values-es/strings.xml** para los valores en español, etc. El entorno Android se encargará durante la ejecución de cargar el fichero strings.xml adecuado según el lenguaje del dispositivo. De la misma forma se pueden definir distintas interfaces o estilos para las distintas orientaciones de la pantalla o para las distintas versiones de Android.

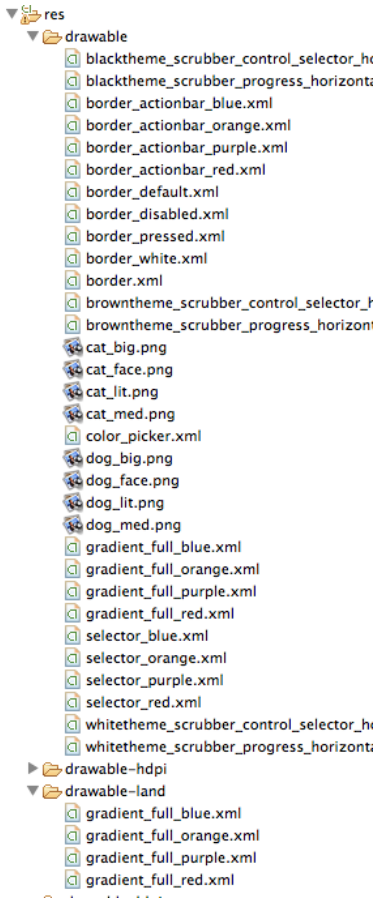

Figura 28 – Carpeta res (1/2)

En este proyecto, una gran cantidad de los recursos se encuentran externalizados de esta manera, y se han aprovechado en gran medida estas posibilidades. A pesar de ello, la aplicación solo contiene los valores en castellano. Seguidamente se hablará de las distintas interfaces definidas durante el desarrollo de esta aplicación.

Interfaces

De entre todos los recursos estáticos antes mencionados posiblemente las interfaces de usuario se traten de los más importantes, extensos, y los que más tiempo han necesitado para su creación. Desde el principio del desarrollo se ha seguido una misma guía: se han tomado los elementos y controles Android existentes, y se han personalizado dándoles formas redondeadas y colores claros, de manera que la interfaz no es demasiado agresiva para el usuario, pero en cierto modo se mantiene nativa a Android.

Los colores utilizados en la aplicación han sido definidos en el fichero **colors.xml**. Una gran variedad de colores y tonalidades se han incluido en este fichero, para poder cambiar de uno a otro fácilmente a la hora de encontrar la mejor opción. De entre todos, dos colores principales se han utilizado: el naranja y el verde.

El naranja, primero por sí mismo, más tarde en combinación con el blanco, formando un gradiente vertical naranja-blanco-naranja (horizontal para la orientación horizontal del dispositivo), es utilizado de fondo en toda la aplicación. Más tarde, este mismo gradiente se ha creado en otros tres distintos colores (con blanco) para permitir al usuario elegir el que prefiera.

El verde, ha sido utilizado como contenedor para los elementos manipulables de la interfaz, resaltándolos así del fondo naranja. Estos contenedores verdes son rectángulos con las esquinas redondeadas, que se adaptan al tamaño de los elementos que albergan y diferentes versiones de ellos están definidos en ficheros XML cuyo nombre empieza por **border**.

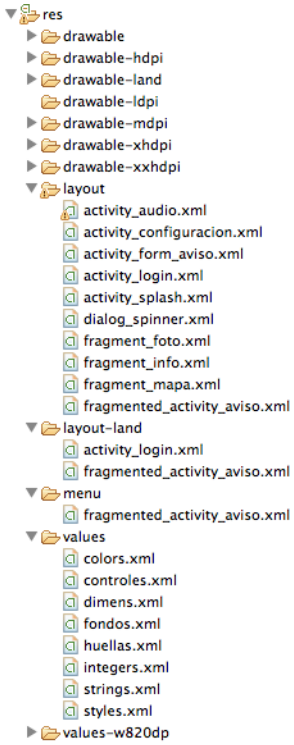

Figura 29 – Carpeta res (2/2)

El principal llamado **border**, toma distintos colores dependiendo de su estado. Normalmente es verde claro, cuando se pulsa es verde más oscuro, y si está deshabilitado (o seleccionado) en azul. En algunas situaciones se ha utilizado una variante blanca definida en **border_white**, por ejemplo, en los campos de introducción de texto. Por último, se ha definido una última versión siempre verde para la barra de opciones (**ActionBar**) definida en **border_actionbar**.

Para los colores del texto, inicialmente se utilizó el negro, en un segundo momento el blanco, pero la versión definitiva es en blanco con un fino borde negro para que resalte sobre el fondo verde. Esto está definido en el fichero **styles.xml**. Seguidamente se tratará una a una las distintas interfaces y sus distintas versiones durante el proceso de desarrollo.

Interfaz de arranque de la aplicación

La pantalla de arranque que se muestra cada vez que la aplicación arranca dispone de una interfaz que únicamente tiene el icono de la aplicación sobre el fondo que se esté usando. Aunque no contiene ningún elemento manipulable, mediante código se ha hecho que al tocar cualquier punto, esta pantalla se desvanezca. Su definición se encuentra en el fichero **activity_splash.xml**. Esta interfaz se muestra en la figura 30.

Interfaz de inicio de sesión

Para la interfaz de la pantalla de inicio de sesión se han definido dos partes, la primera con los campos de texto para que el usuario introduzca su nombre de usuario y contraseña, la segunda con dos botones, uno para autenticarse en el servidor y otro para saltarse esta pantalla sin autenticarse.

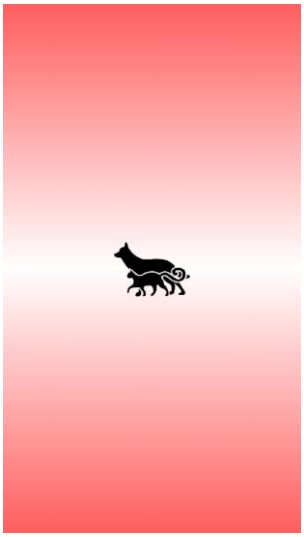

Figura 30 – Interfaz de arranque

Estos elementos se han definido por un lado en el fichero **layout/activity_login.xml** con la primera parte encima de la otra para cuando la orientación de la pantalla es vertical (Figura 31). Por otro lado, se ha redefinido en el fichero **layout-land/activity_login.xml** con la primera parte a la izquierda, y la segunda a la derecha para aprovechar mejor el espacio cuando el dispositivo está en orientación horizontal (Figura 32).

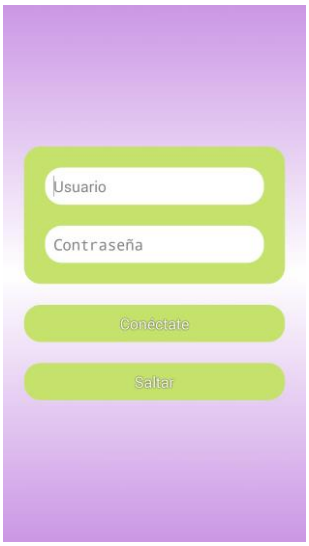

Figura 31 – Interfaz de inicio de sesión (Orientación vertical)

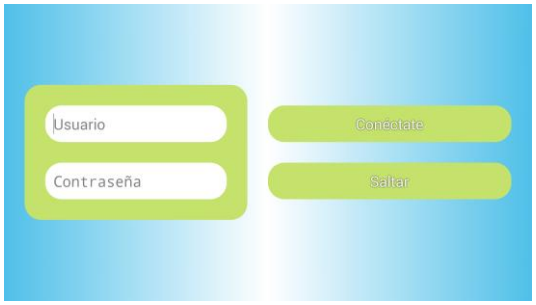

Figura 32 – Interfaz de inicio de sesión (Orientación Horizontal)

Interfaz de configuración

La interfaz de configuración reúne los componentes necesarios para que el usuario pueda configurar a su gusto la aplicación. En primer lugar se encuentran cuatro rectángulos cada uno de un color para elegir el fondo de la aplicación. Estos rectángulos se tratan de contenedores de imágenes (**ImageView**) cuya imagen es un elemento **border** del color correspondiente. Aunque no se aprecie visualmente, el color seleccionado se encuentra deshabilitado para que no se ejecute el código que cambia el color por el color actual. Esto no es importante porque el usuario puede ver el color actual en el fondo.

En segundo lugar se encuentra un componente para introducir texto cuyo fin es introducir la URL del servidor. Este componente no se mostrará al usuario final, pero durante el transcurso de este trabajo, es conveniente su aparición por la flexibilidad que da durante todo el proceso.

Por último aparece un *checkbox* (un componente visual que permite indicar un valor verdadero o falso), permite decidir si las imágenes serán reducidas para los envíos o no, para aquellos usuarios que no quieran usar en exceso su plan de datos. Un texto explicativo a la izquierda del *checkbox* indica al usuario la función de este. Esta interfaz se muestra en la Figura 33.

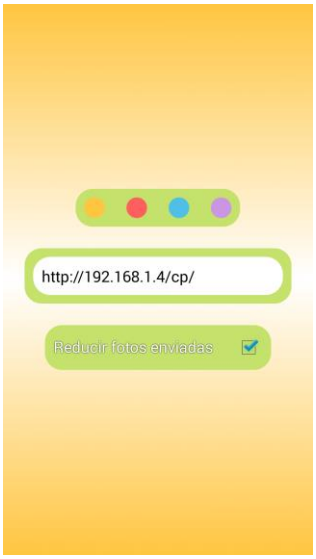

Figura 33 – Interfaz de configuración

Interfaz de envío de avisos

En esta interfaz cada sección utiliza su propio diseño como se aprecia en la figura 34. La primera, a la izquierda, utiliza *seek bars* para definir al animal, y diferentes elementos visuales para mostrar los cambios al usuario. Al ser una sección con mucha información no toda cabe en la pantalla por lo que se han habilitado barras de desplazamiento. La segunda, en el centro, es un cuadro para mostrar la foto seleccionada (en blanco si no hay ninguna) que se expande lo máximo posible, con dos botones debajo para elegir una imagen de la galería o tomar una con la cámara. La tercera, a la derecha, es un mapa de Google Maps que de la misma forma, también se expande lo máximo posible.

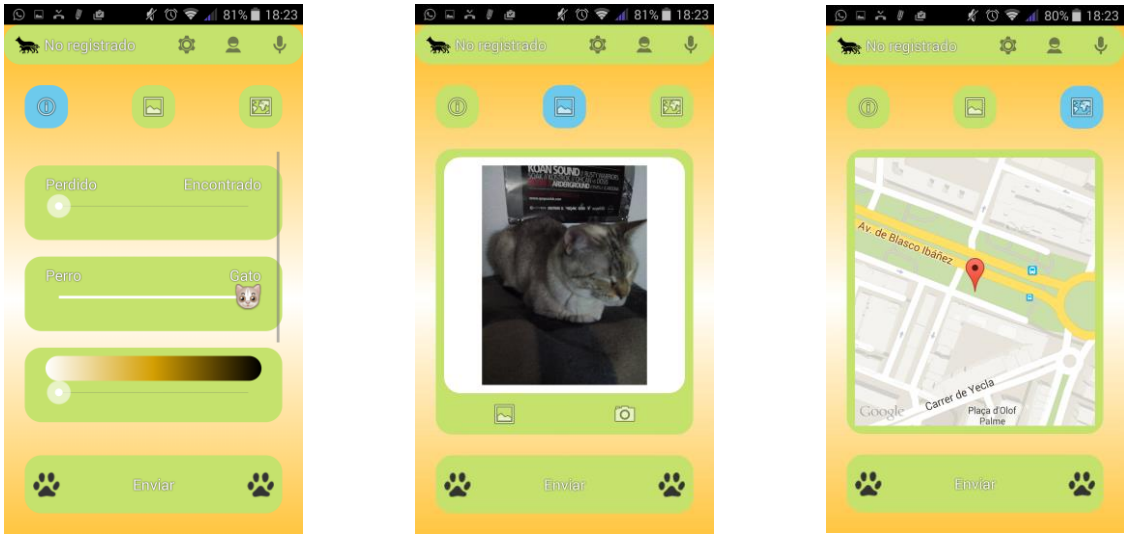

Figura 34 – Interfaces de envío de aviso (Orientación vertical)

Además se definió una versión alternativa para cuando el dispositivo se encuentra en posición horizontal. Esta tiene los tres botones para elegir sección a la izquierda, el de enviar a la derecha, con una imagen de enviar en vez de texto. En medio se muestra el fragmento elegido. Cabe destacar que los botones de las secciones así como el de enviar para la versión horizontal, toman las imágenes del sistema operativo, así que dependiendo de la versión de Android, la imagen puede cambiar. Esta versión alternativa se puede ver en la figura 35.

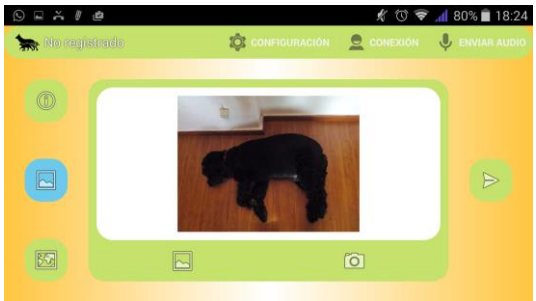

Figura 35 – Interfaz de envío de aviso (Orientación Horizontal)

En cuanto a la barra de herramientas, utiliza una forma **border**, y muestra el icono de la aplicación, el nombre de usuario actual o "No registrado" y las opciones disponibles. Las opciones, definidas aparte en el fichero **menu/fragmented_activity_aviso.xml,** son gestionadas por el sistema operativo, y pueden aparecer con icono y texto, solo con icono, u ocultas en un menú de más opciones, en función del espacio disponible.

Interfaz de grabación de audio

Esta interfaz está compuesta únicamente por un botón con un micrófono dibujado y un sencillo texto explicativo. Este texto inicialmente indica que para empezar a grabar hay que pulsar el botón mientras que durante la grabación, este texto cambiará a "Grabando…". Al tener pocos componentes, solo se ha definido una única interfaz (figura 36) para ambas orientaciones del dispositivo.

Figura 36 – Interfaz de grabación de audio

5.3. Sistema soporte de la aplicación

En el capítulo de Especificación de requisitos se destacaba que una de las suposiciones que se hacían para este proyecto era la de disponer de la implementación de una interfaz de servicios web REST albergada en un servidor. La función a grandes rasgos de este servidor es la de actuar como persistencia de la aplicación Android, y además procesar y cotejar la información recibida con la existente para encontrar avisos similares.

Durante el transcurso de este TFG esta suposición no se ha cumplido, así que para poder seguir con el trabajo, además de la aplicación Android se ha implementado un servidor web con la interfaz web propuesta en el capítulo de diseño, y la base de datos correspondiente. Es importante hacer notar que el servidor web no es el objetivo de este trabajo, de manera que no se ha invertido un tiempo excesivo en él, y no incluye ni procesamiento, ni cotejamiento ni cuestiones de seguridad, sino que actúa como sistema de soporte para que la aplicación Android pueda funcionar.

Tanto los servicios web como la base de datos se encuentran albergados en un servidor web MAMP (servidor para el sistema **M**ac OS X, y que dispone de los componentes **A**pache, **M**ySQL y **P**HP/**P**erl/**P**ython, véase **[28]**), y la aplicación Android accede a él mediante la IP del ordenador, que no es fija, y depende de la red que compartan los dos dispositivos.

5.3.1. Servicios Web Implementados

Para implementar los servicios web se ha utilizado CakePHP, un Framework para el desarrollo rápido de aplicaciones web en PHP (Véase **[28]**, **[29]**, **[30]**, **[31]**, **[32]**, **[38]**, **[39]** y **[40]**) que permite la creación de aplicaciones web en general con apenas unas cuantas líneas de código. Se han implementado los dos servicios web propuestos en el capítulo de diseño. En la tabla 25 se muestra la ruta relativa para acceder a cada servicio web, con ejemplos de archivo JSON para la petición y para la respuesta.

El primero, permite que la aplicación autentique al usuario. Cuando la aplicación envía el nombre de usuario y contraseña introducidos, el servidor recupera de la base de datos la información del usuario en cuestión. Si la contraseña almacenada no fuese la misma que la que la *app* ha enviado, o si no se encontrase al usuario en la base de datos, se responde con un mensaje JSON negativo indicando que hay algún problema (*false*). En caso contrario se envía un mensaje afirmativo (*true*). Además se actualizará el campo **modificado** de la base de datos con la fecha y hora actual del servidor (esta es la fecha de última conexión).

El segundo añade la posibilidad de enviar avisos y almacenarlo en el servidor. Cuando un aviso es enviado, la información se extrae del mensaje y cada dato se almacena en su correspondiente campo de la base de datos. Hay una excepción en esto: el nombre de usuario del usuario que envía el aviso es sustituido por su **id** en la base de datos de usuarios. En el caso de que esto no se pueda hacer, ningún **id** es insertado, pero la tabla de avisos tiene un valor por defecto para el **id** del usuario correspondiente al usuario anónimo o no registrado. Además el campo **creado** se actualiza con la fecha y hora actual del servidor. Si todo sale bien se envía una respuesta afirmativa, en caso contrario se responde negativamente.

Servicio Web	Ruta	Entrada	Salida

Conexión	/usuarios/conexion	{"Usuario":
		{
		"username": alsanqui,
		"password": animales
		}
		}
			{"resultado": *"true"*}

Enviar Aviso	/animales/insertar	{"Aviso":
		{
		"tipo": 0,
		"especie": 1,
		"talla": 2,
		"color": 0,
		"temperamento": 1,
		"latitud": 39.4739020,
		"longitud": -0.3492620,
		"usuario":
		{"Usuario":
		{
		"username": alsanqui
		}
		}
		}
		}
			{"resultado": *"false"*}

Tabla 25 – Servicios Web REST implementados y ejemplos de mensajes JSON

5.3.2. Base de datos

Para poder almacenar toda la información recibida en el servidor, el sistema de gestión de bases de datos es un componente imprescindible. En este caso para gestionar las bases de datos se ha utilizado MySQL (véase **[33]** y **[41]**), un sistema de código abierto bastante extendido en el mundo de las aplicaciones web y que viene incluido en el paquete MAMP mencionado al principio de este apartado.

Mediante este sistema, se ha creado una base de datos para la aplicación llamada Animales con una versión simplificada del diagrama de clases mostrado en el capítulo de análisis dado que el cotejamiento no se ha implementado (Figura 37). Como se aprecia, se han definido dos tablas, **avisos** y **usuarios**, teniendo la primera una clave ajena a la segunda mediante el campo **usuario_id** que indica el usuario que ha añadido el aviso.

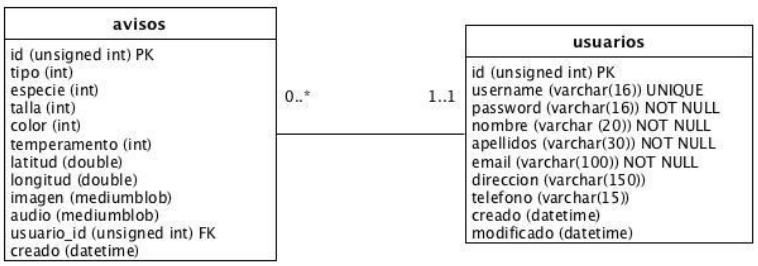

Figura 37 – Tablas de la base de datos

5.4. Pruebas

Una vez implementado todo, el siguiente paso es asegurarse de que se cumplen los requisitos funcionales definidos en el capítulo de especificación de requisitos. Para ello se han definido una serie de pruebas que tendrán uno o más requisitos funcionales asociados, y con distintos condiciones iniciales. Al realizarlas sobre la aplicación, ofrecerán unos resultados que habrá que comparar con lo que se espera que ocurra. De esta manera es posible cerciorarse que la aplicación funciona correctamente. Estas pruebas se han separado en función de la característica a la que están asociadas.

5.4.1. Gestión de usuarios

Prueba	PU01
Descripción	Entrar sin autenticación
Casos de uso asociados	RF01
Condiciones iniciales	-
Resultado obtenido	La pantalla para enviar avisos se muestra indicando “No registrado” en la barra superior.

Tabla 26 – Prueba realizada: Gestión de usuarios 1

Prueba	PU02
Descripción	Iniciar sesión
Casos de uso asociados	RF01
Condiciones iniciales	Usuario no existe
Resultado obtenido	Se muestra un mensaje de error en pantalla, y no se muestra la pantalla de envío de avisos.

Tabla 27 – Prueba realizada: Gestión de usuarios 2

Prueba	PU03
Descripción	Conexión
Casos de uso asociados	RF01
Condiciones iniciales	El usuario existe pero la contraseña es errónea
Resultado obtenido	Se muestra un mensaje de error en pantalla, y no se muestra la pantalla de envío de avisos

Tabla 28 – Prueba realizada: Gestión de usuarios 3

Prueba	PU04
Descripción	Conexión
Casos de uso asociados	RF01
Condiciones iniciales	El usuario existe y la contraseña es correcta
Resultado obtenido	La pantalla para enviar avisos se muestra indicando el nombre de usuario en la barra superior

Tabla 29 – Prueba Realizada: Gestión de usuarios 4

Prueba	PU05
Descripción	Conexión
Casos de uso asociados	RF01
Condiciones iniciales	Problema de conexión o URL errónea
Resultado obtenido	Se muestra un mensaje de error en pantalla, y no se muestra la pantalla de envío de avisos

Tabla 30 – Prueba realizada: Gestión de usuarios 5

5.4.2. Gestión de avisos

Prueba	PA01
Descripción	Enviar aviso
Casos de uso asociados	RF09
Condiciones iniciales	El usuario no está registrado
Resultado obtenido	Se indica por pantalla que el aviso se ha enviado correctamente, y una nueva fila es añadida a la tabla **avisos** con los datos del aviso e *id* del usuario con valor 1 (no registrado)

Tabla 31 – Prueba realizada: Gestión de avisos 1

Prueba	PA02
Descripción	Enviar aviso
Casos de uso asociados	RF01, RF09
Condiciones iniciales	El usuario está registrado
Resultado obtenido	Se indica por pantalla que el aviso se ha enviado correctamente, y una nueva fila es añadida a la tabla **avisos** con los datos del aviso e *id* del usuario que ha enviado el aviso

Tabla 32 – Prueba realizada: Gestión de avisos 2

Prueba	PA03
Descripción	Enviar aviso
Casos de uso asociados	RF09
Condiciones iniciales	Problema de conexión o URL errónea
Resultado obtenido	Se muestra un mensaje de error en pantalla y no se crea una nueva fila en la tabla **avisos**

Tabla 33 – Prueba realizada: Gestión de avisos 3

Prueba	PA04
Descripción	Seleccionar imagen
Casos de uso asociados	RF02
Condiciones iniciales	Imagen de la cámara
Resultado obtenido	Aparece la cámara del dispositivo, y al tomar una imagen, esta se muestra en la sección de imagen de la pantalla de enviar aviso

Tabla 34 – Prueba realizada: Gestión de avisos 4

Prueba	PA05
Descripción	Seleccionar imagen
Casos de uso asociados	RF03
Condiciones iniciales	Imagen de la galería
Resultado obtenido	Se muestra en pantalla la galería de imágenes del dispositivo y al seleccionar una, esta se muestra en la sección de imagen de la pantalla de enviar aviso

Tabla 35 – Prueba realizada: Gestión de avisos 5

Prueba	PA06
Descripción	Enviar imagen
Casos de uso asociados	RF04
Condiciones iniciales	Imagen seleccionada de galería o cámara
Resultado obtenido	Se indica que el aviso se ha enviado correctamente y una nueva fila se añade a la tabla avisos con la imagen en formato base64. Si la imagen se decodifica y se abre con un visualizador de imágenes, se ve bien

Tabla 36 – Prueba realizada: Gestión de avisos 6

Prueba	PA07
Descripción	Grabar audio
Casos de uso asociados	RF05
Condiciones iniciales	-
Resultado obtenido	Tras grabar el audio se indica en pantalla que la grabación se ha hecho correctamente.

Tabla 37 – Prueba realizada: Gestión de avisos 7

Prueba	PA08
Descripción	Enviar audio
Casos de uso asociados	RF06
Condiciones iniciales	Audio grabado
Resultado obtenido	Se indica que el aviso se ha enviado correctamente y una nueva fila se añade a la tabla avisos con la grabación en formato base64. Si la grabación se decodifica y se abre con un reproductor de audio/video, se escucha bien.

Tabla 38 – Prueba realizada: Gestión de avisos 8

5.4.3. Localización

Prueba	PL01
Descripción	Obtener localización del dispositivo
Casos de uso asociados	RF07
Condiciones iniciales	Configuración correcta
Resultado obtenido	Tras unos segundos se muestra al usuario que la localización se ha obtenido y este la puede visualizar en el mapa

Tabla 39 – Prueba realizada: Localización 1

Prueba	PL02
Descripción	Obtener localización del dispositivo
Casos de uso asociados	RF07
Condiciones iniciales	Configuración incorrecta sin posible corrección
Resultado obtenido	Se avisa al usuario que su localización no se puede obtener, y el mapa no muestra ninguna localización

Tabla 40 – Prueba realizada: Localización 2

Prueba	PL03
Descripción	Obtener localización del dispositivo
Casos de uso asociados	RF07
Condiciones iniciales	Configuración incorrecta con posible corrección
Resultado obtenido	Se le pide al usuario si quiere cambiar la configuración (si o no)

Tabla 41 – Prueba realizada: Localización 3

Prueba	PL04
Descripción	Obtener localización del dispositivo
Casos de uso asociados	RF07
Condiciones iniciales	Configuración incorrecta con posible corrección y el usuario acepta
Resultado obtenido	Tras unos segundos se muestra al usuario que la localización se ha obtenido y este la puede visualizar en el mapa

Tabla 42 – Prueba realizada: Localización 4

Prueba	PL05
Descripción	Obtener localización del dispositivo
Casos de uso asociados	RF07
Condiciones iniciales	Configuración incorrecta con posible corrección y el usuario rechaza
Resultado obtenido	Se avisa al usuario que su localización no se puede obtener, y el mapa no muestra ninguna localización

Tabla 43 – Prueba realizada: Localización 5

Prueba	PL06
Descripción	Enviar localización del dispositivo
Casos de uso asociados	RF08
Condiciones iniciales	Localización obtenida
Resultado obtenido	Se indica que el aviso se ha enviado correctamente y una nueva fila se añade a la tabla avisos con las coordenadas de la localización. Si se introducen en Google Maps se muestra la posición correcta

Tabla 44 – Prueba realizada: Localización 6

5.4.4. Configuración

Prueba	PC01
Descripción	Cambiar la configuración del dispositivo
Casos de uso asociados	RF10
Condiciones iniciales	-
Resultado obtenido	La configuración se cambia al instante y se mantiene hasta el próximo cambio o hasta que la aplicación se desinstala o se borran sus datos que los valores configurables vuelven a su valor por defecto

Tabla 45 – Prueba realizada: Configuración

5.4.5. Resultado de las pruebas

Una vez realizadas las pruebas se aprecia que el comportamiento de la aplicación es bastante aproximado a lo que se espera de ella. Se ha comprobado que la información llega correctamente al servidor, que la aplicación cumple las funciones que se esperan de ella y que indica al usuario en todo momento de los errores ocurridos y de aquello que sea relevante para el usuario. Por lo tanto se puede decir que se adapta a los requisitos especificados anteriormente.

6. Conclusiones

Para terminar, este capítulo tratará las principales dificultades encontradas durante el transcurso del trabajo y cómo estas han sido superadas en el primer apartado. En un segundo apartado se extraerán las aportaciones realizadas con este trabajo y por último en el apartado tercero se presentarán posibles ampliaciones para la aplicación.

6.1. Dificultades encontradas

Sin duda la mayor dificultad encontrada durante el transcurso de este TFG ha sido la falta de la segunda parte importante del sistema, el servidor. Sin éste la aplicación no puede funcionar ni se puede testear en condiciones además de no poder guardar la información. A pesar de disponer de un conocimiento limitado sobre tecnologías web, esto se ha sobrevenido creando lo que en el capítulo anterior se ha llamado sistema soporte de la aplicación.

Aun así esta falta de servidor en condiciones ha sido uno de los mayores cuellos de botella del trabajo y se ha optado por ser realista y establecer un contrato para la aplicación en forma de especificación de requisitos y otro para el servidor en forma de interfaz web cuyas clausulas se podían cumplir a lo largo de este trabajo. De esta forma se obtiene así un sistema cerrado, en el sentido de que la aplicación fruto de este trabajo parte de una base bien definida, y cumple todas las expectativas que se especificaron inicialmente para ella como se ha visto en el apartado de pruebas del capítulo anterior.

6.2. Aportaciones realizadas

Seguramente las dos aportaciones más destacables de este trabajo son las dos aplicaciones Android de las que se ha hablado. En primer lugar está la aplicación ideal que nace del análisis estratégico realizado en el segundo capítulo y que define una propuesta de cómo tendría que ser una aplicación destinada a la localización y aviso de animales perdidos.

La segunda aplicación aportada es la aplicación real desarrollada para este trabajo. Se trata de una aplicación funcional, que es capaz de recuperarse de errores y con un buen grado de adaptabilidad. Además ofrece al usuario una gran variedad de formas de introducir la información.

6.3. Trabajo futuro

La siguiente base a alcanzar en este trabajo sería sin duda conseguir una implementación en condiciones del servidor, robusta, con la seguridad adecuada y con una interfaz más amplia que ofrecer a la aplicación Android. Una vez esto conseguido la aplicación será capaz de crecer más en funcionalidad acercándose más así a la aplicación ideal. Una propuesta de posibles futuras funcionalidades podría ser:

- Permitir registrarse en la *app*
- Visualización y modificación de avisos anteriores
- Mejor gestión de usuarios y añadir perfil de colaborador
- Mayor grado de adaptación añadiendo por ejemplo más lenguajes

Por último habrá que encontrar soporte para la aplicación en dos sentidos. Para que tenga éxito la aplicación habrá que alcanzar un gran público que empiece a utilizarla y además la aplicación tendrá que estar respaldada por las asociaciones de animales aportando colaboradores que ayuden en la tarea.

Bibliografía

[1] Wikimedia. Android (Operating System), 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/Android_(operating_system)

[2] Simplifica Software. Animales Sin Hogar, 2012 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=com.simplifica.ash.dashboardash

[3] Alzago Soluciones Tecnológicas S.L. Alerta Animal, 2014 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=com.alzago.alertaanimal

[4] PetFinder.my and KindMeal.my Team. PetFinder.my, 2013 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=com.petfinder.andykoh

[5] Goddard Information Systems. HeLP Lost and Adoptable Pets, 2014 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=com.helpinglostpets.colin

[6] Larry Colett. Charleston Animal Society, 2012 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=com.app_chasanimal.layout

[7] Alconsoft Panama. Callejerito Help!, 2014 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=com.alchesoft.callejerohelp

[8] Bukosabino. Guau! qué perros, 2015 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=com.ionicframework.buscaperrocliente620403

[9] Javier Porto. Encuéntralos, 2014 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=com.ayudaaencontrarlos.encuentralos

[10] Magma. Find My Pet, 2014 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=myapp.magma.findmypet

[11] Creativity Apps. Pet Finder, 2015 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=co.creativityapps.petfinder

[12] petsearch. Pet Search, 2014 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=com.appbuilder.u501565p781228

[13] Michael Hasibo. IcPet – Report/Find Lost Pets, 2015 [en línea]. Google Play [fecha de consulta: 21 de febrero de 2015]. Disponible en https://play.google.com/store/apps/details?id=com.hasiboapps.icpet

[14] Wikimedia. Global Android version distribution since December 2009, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 21 de junio de 2015]. Disponible en https://en.wikipedia.org/wiki/Android_version_history#/media/File:Android_historical_version_distribution_-_vector.svg

[15] Wikimedia. Dimensionality Reduction, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/Dimensionality_reduction

[16] The Institute of Electrical and Electronics Engineers, Inc. IEEE Recommended Practice for Software Requirements Specifications, 1998 [en línea]. University of Alaska Anchorage [fecha de consulta: 10 de marzo de 2015]. Disponible en http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf

[17] Wikimedia. Hypertext Transfer Protocol, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[18] Wikimedia. JSON, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/JSON

[19] Wikimedia. Representational State Transfer, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/Representational_state_transfer

[20] Wikimedia. Web Service, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/Web_service

[21] Wikimedia. XML, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/XML

[22] World Wide Web Consortium. XML, 2015 [en línea]. World Wide Web Consortium [fecha de consulta: 22 de julio de 2015]. Disponible en http://www.w3.org/XML/

[23] Garrote-Hildebrand, D., Poza-Lujan, J. L., Posadas-Yagüe, J. L., Simó-Ten, J. E., Jiménez-García, J. L. Distributed System for Intelligent Management of Lost or Abandoned Pets, 2014 [en línea]. Universitat Politècnica de València [fecha de consulta: 19 de febrero de 2015]. Disponible en http://wks.gii.upv.es/cobami/files/IJIR-R-AIR-3.pdf

[24] Wikimedia. Business Object, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/Business_object

[25] Wikimedia. Data Access Object, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/Data_access_object

[26] Santiago Lezica. Writing Splash Screens The Right Way, 2012 [en línea]. blogActivity; [fecha de consulta: 22 de julio de 2015]. Disponible en https://blogactivity.wordpress.com/2012/02/24/writing-splash-screens-the-right-way/

[27] Wikimedia. Singleton Pattern, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/Singleton_pattern

[28] Wikimedia. MAMP, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/MAMP

[29] Wikimedia. CakePHP, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/CakePHP

[30] CakePHP. CakePHP Cookbook, 2015 [en línea]. CakePHP [fecha de consulta: 22 de julio de 2015]. Disponible en http://book.cakephp.org/3.0/_downloads/en/CakePHPCookbook.pdf

[31] Wikimedia. PHP, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/PHP

[32] Seralo. PHP Frameworks comparison, 2010 [en línea]. Social Compare, Collaborative Comparison Engine [fecha de consulta: 22 de julio de 2015]. Disponible en http://socialcompare.com/en/comparison/php-frameworks-comparison

[33] Wikimedia. MySQL, 2015 [en línea]. Wikipedia, The Free Encyclopedia [fecha de consulta: 22 de julio de 2015]. Disponible en https://en.wikipedia.org/wiki/MySQL

Enlaces adicionales

[34] Android Reference:

http://developer.android.com/reference/packages.html

[35] Stack Overflow – Android:

http://stackoverflow.com/questions/tagged/android

[36] The New Boston – Android Application Development Tutorial:

https://www.youtube.com/watch?v=SUOWNXGRc6g&list=PL33384E9848C4F55E

[37] MAMP:

https://www.mamp.info/en/

[38] CakePHP:

http://cakephp.org/

[39] The New Boston – Beginner PHP Tutorial:

https://www.youtube.com/watch?v=iCUV3iv9xOs&list=PL442FA2C127377F07

[40] w3schools.com – PHP:

http://www.w3schools.com/php/php_intro.asp

[41] MySQL:

https://www.mysql.com/