Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

DISCA-278:

APLICACIÓN ANDROID PARA LA GESTIÓN INTEGRAL DE LA BÚSQUEDA Y LA LOCALIZACIÓN DE MASCOTAS PERDIDAS O ABANDONADAS

Proyecto Final de Carrera

Ingeniería Técnica en Informática de Gestión

Autor: Víctor Manuel Chisvert Amat

Director: José Luis Poza Luján

30/09/2014

Resumen

A causa de la problemática actual de los animales abandonados y perdidos, y estudiando las distintas herramientas disponibles en la actualidad, se puede afirmar, que estas herramientas aunque son capaces de recopilar ingente cantidad de información, no es una información de calidad, ya que no está organizada, ni persiste en él tiempo. La aplicación desarrollada para dispositivos móviles con el sistema operativo Android, intenta de una forma fácil, crear un software capaz de aunar distintos métodos de entrada de datos y centralizarlos en un servidor.

Palabras clave: android, mascotas perdidas, mascotas abandonadas, animales perdidos, animales abandonados, dispositivos móviles, fragments, viewpager.

Índice

Tabla de contenidos

1. Introducción

1.1 Entorno

Miles de mascotas pérdidas o abandonadas están actualmente en nuestras calles, muchas asociaciones y particulares intentan subsanar esta situación apoyándose en las herramientas disponibles actualmente, redes sociales mayoritariamente, estas herramientas no estas diseñadas para este propósito, en consecuencia la mayoría de información recopilada acaba diluyéndose, a pesar de que existe cantidades ingentes de información respecto a animales, solo una pequeña parte llega a los destinatarios adecuados, así que el impacto de estas herramientas en la disminución de mascotas en nuestras calles no es proporcional con los recursos que se destinan.

1.2 Objetivos

El objetivo de esta aplicación es la de facilitar la recopilación de información para la búsqueda y captura de animales perdidos y abandonados, y así reducir la ardua tarea de asociaciones y particulares a la hora de ubicar y buscar animales. En consecuencia reducir el número de mascotas en esta situación.

Esto se consigue apoyándose en un software capaz de aunar distintos tipos de captura de información y centralizarla de una forma sencilla, haciendo uso de nuevas tecnologías de fácil manejo y muy extendidas, que aportan una serie de interfaces de entrada muy efectivas para nuestro propósito.

1.3 Descripción del documento

Este documento se divide en ocho apartados y un anexo.

El apartado 1 con el título **Introducción**, se hace una breve descripción de la problemática que trata de resolver esta aplicación, seguido de los objetivos a los que se quiere llegar, por último se describe este documento.

El apartado 2 con el título **Estado del arte**, se hace un análisis de la problemática de los animales perdidos y abandonados, a través de un estudio.
A continuación se expone la evolución de la tecnología móvil hasta la actualidad, centrándonos en los Smartphone-Android ya que son el tipo de dispositivo a utilizar, así como una breve descripción de las herramientas y componentes de que disponen estos para la captura y divulgación de información.
Por último se hace una recopilación de herramientas que se utilizan actualmente para la búsqueda y ubicación de animales, analizando las ventajas e inconvenientes que tienen.

En el apartado 3 titulado **Entorno de desarrollo**, se describirá la tecnología y herramientas utilizadas para dar forma a la aplicación detallando cada una de ellas.

El apartado 4 con el título **Especificación de requisitos**, a través del estándar IEEE 830 se describirá las restricciones, dependencias y funcionalidades que el sistema debe realizar.

El apartado 5 con el título **Diseño** se describirá la arquitectura del sistema mostrando la estructura de la base de datos, el comportamiento del sistema o la relación y propiedades de las clases del sistema a través de estándar UML, con diagramas de casos de uso y con diagramas de clases, explicando las entidades más importantes.

En el apartado 6 con el título **Implementación** se explicarán las partes más importantes del desarrollo de la aplicación detallando en algunos casos las dificultades encontradas y como se han resuelto, y mostrando el código más significativo.

En el apartado 7 con el título **Conclusiones**, se hará un repaso del trabajo realizado, las aportaciones que desarrollan y las posibles mejoras que se pueden aportar.

En el apartado 8 con el título **Referencias,** se presentaran las fuentes tanto de Internet como bibliográficas donde se ha apoyado gran parte de la documentación.

En el último apartado con el título **Anexo** se explicará las funcionalidades de la aplicación móvil mediante un manual de usuario, que contendrá capturas de pantalla y pasos a seguir, para poder sacar el máximo rendimiento a la aplicación.

2. Estado del arte

2.1. Introducción

Un país, una civilización se puede juzgar por la forma que trata a sus animales (Mahatma Gandhi)

2.2. Problemática de los animales abandonados y perdidos

Según el "Estudio de la Fundación Affinity sobre el abandono de animales de compañía" elaborado en 2011 con datos provenientes de ayuntamientos y protectoras del 2010.

La mayoria de animales recogidos son perrros un 77.6% mientras que los gatos representan un 22.4%

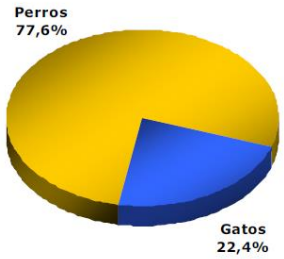


Ilustración 1: Distribución de animales recogidos

Durante el año 2010 se recogieron un total de 109.074 perros, una media de 2.3 perros por cada 1000 habitantes, que supone un descenso del 5.9% respecto al 2009, con esto se mantiene por segundo año consecutivo un descenso del número de perros recogidos en España.

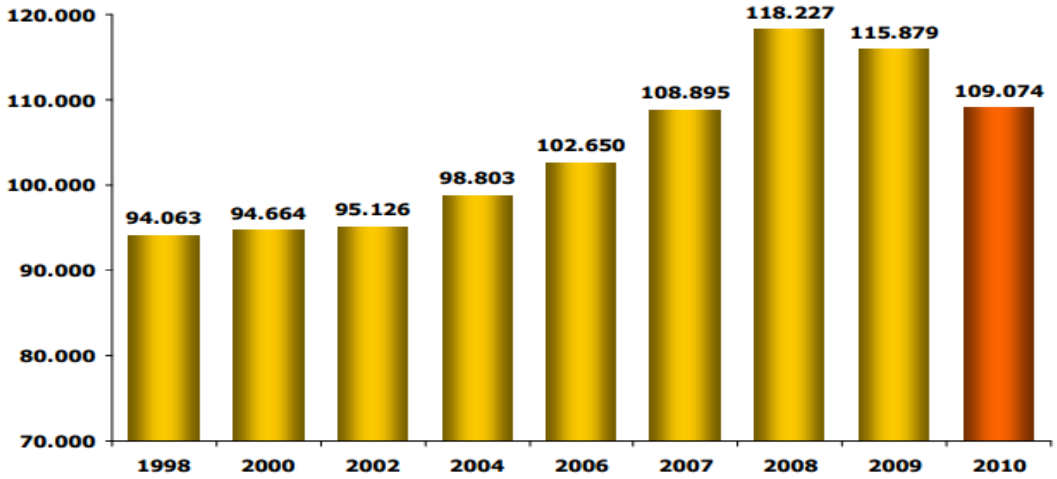


Ilustración 2: Evolución del número de perros recogidos (1998-2010)

Cabe destacar que el periodo con mayor tasa de abandono de perros es el segundo cuatrimestre del año con un 34.7%

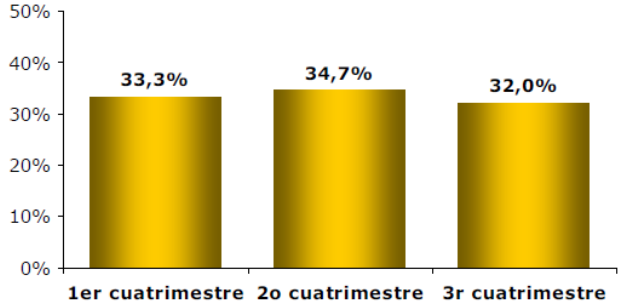
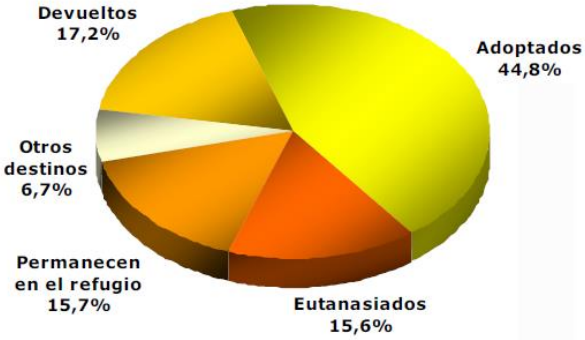


Ilustración 3: Perros recogidos por cuatrimestre

En cuanto al destino de los perros recogidos, el 44.8% de perros recogidos fueron adoptados, el 15.7% permanecieron en el refugio, el 15.6% fueron eutanasiados, el 6.7% tuvieron otros destinos y el 17.2% fueron devueltos a sus amos.
Se puede apreciar un aumento de las adopciones, mientras que los eutanasiados han disminuido.



	2008	2009
Adoptados	29,4%	39,2%
Permanecen en el refugio	30,7%	25,5%
Eutanasiados	14,7%	15,5%
Devueltos	14,0%	16,8%
Otros destinos	11,1%	3,0%

Ilustración 4: Destino de los perros abandonados

Otros datos que cabe destacar son:

- El 31.2% de los perros recogidos llevaban microchip.
- El 56.2% de los perros recogidos son machos.
- El 57.6% de los perros recogidos son adultos, el 26.3% son cachorros y el 16.1% son sénior.
- El 81.6% son de raza mestiza
- El 43.6% de los perros recogidos son de raza mediana, el 29.3% de raza grande y el 27.1% de raza pequeña.
- El 66.5% de los perros recogidos disponía de buena salud, el 20.4% presentaba alguna enfermedad, mientras el 13.1% sufría alguna herida por maltrato.
- La estancia media en la entidad de recogida para los perros adultos y sénior es de 6.7 meses, para los cachorros de 2.3 meses, mientras que para los perros de raza pura es de 4.7 meses.

Gatos:
Durante el 2010 se recogieron un total de 35.938 gatos en España, una media de 7.7 gatos por cada 10.000 habitantes, se observa un incremento del 0.5% respecto al 2009.

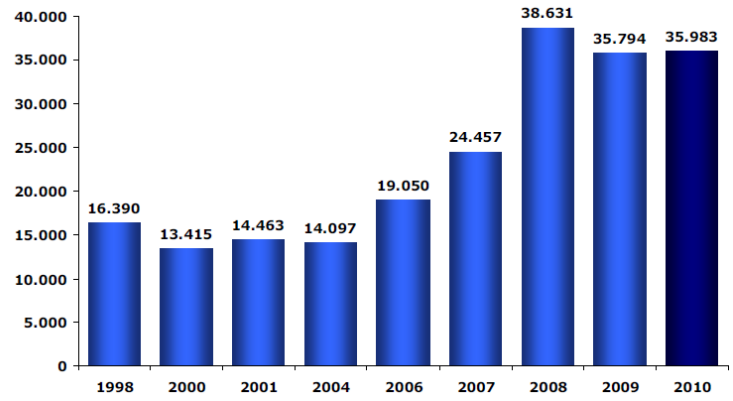


Ilustración 5: Evolución del número de gatos recogidos (1998-2010)

El segundo cuatrimestre del año es el de mayor tasa de abandono de gatos un 41.3%

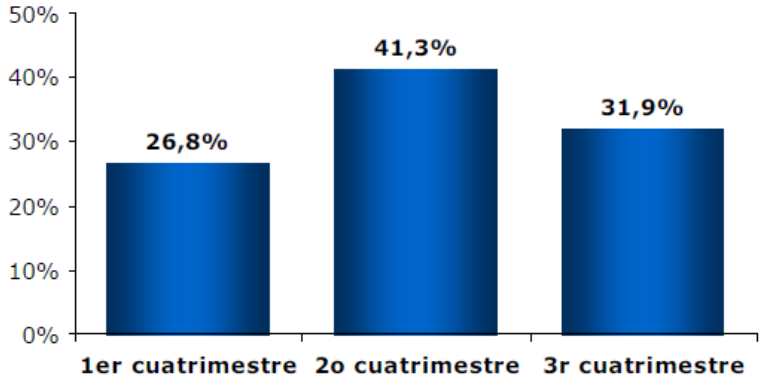
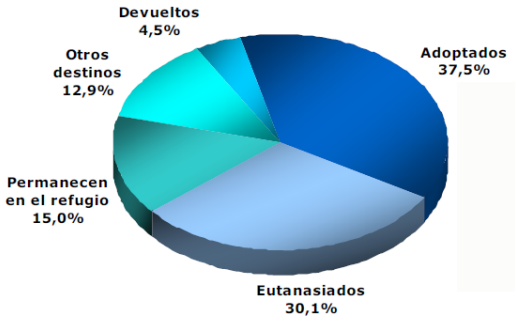


Ilustración 6: Gatos recogidos por cuatrimestre

En cuanto al destino de los gatos recogidos la mayoría un 37.5% fueron adoptados, el 30.1% fueron eutanasiados, el 15% permanecieron en el refugio, el 12.9% tuvieron otros destinos, y el 4.5% fueron devueltos a sus dueños.
Cabe destacar que ha aumentado la adopción respecto a otros años, también ha aumentado significativamente los gatos eutanasiados.



	2008	2009
Adoptados	19,3%	27,6%
Permanecen en el refugio	30,6%	38,2%
Eutanasiados	22,4%	18,6%
Devueltos	4,9%	4,7%
Otros destinos	22,8%	11,0%

Ilustración 7: Destino de gatos abandonados

Otros datos a destacar son:

- Solo el 4.6% de los gatos llevan microchip.
- El 50.8% de los gatos recogidos son machos.
- El 55.8% de los gatos recogidos son adultos el 34.1% son cachorros y el 10.1% sénior.
- El 89.1% de los gatos recogidos son de raza mestiza.

- El 59.7% de los gatos recogidos disponía de buena salud el 25.3% presentaba alguna enfermedad mientras el 15% sufría alguna herida.
- La estancia media en el centro de recogida para los gatos adultos o sénior es de 7.8 meses, para los cachorros es de 6.5 meses mientras que para los gatos de raza pura es de 2.7 meses.

- Un 60% de los animales que estuvieron en alguna sociedad protectora fueron encontrados en la calle, un 20% de los animales llevados a la entidad se realizó por personas ajenas al animal, mientras que el restante 20% se realizó pro el dueño del animal. Se debe destacar que el 11.2% de los animales llevados a la entidad corresponde con regalos de navidad.

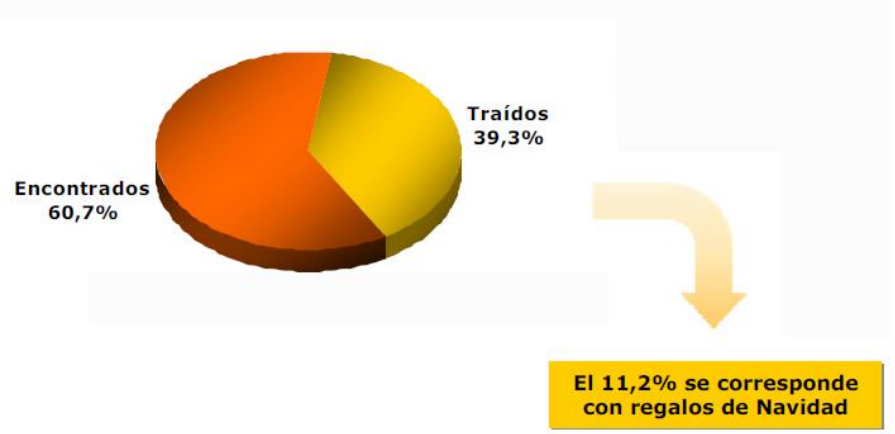


Ilustración 8: Llegada de los animales a la entidad

Con respecto a los motivos del abandono del animal, el primer motivo con un 14% son las camadas inesperadas, seguido con un 13.7% los cambios de domicilio, también hay que destacar como uno de los principales motivos son los factores económicos, que es el que ha sufrido un mayor incremento, pasando del 8.7% en 2009 al 13.2% en 2010

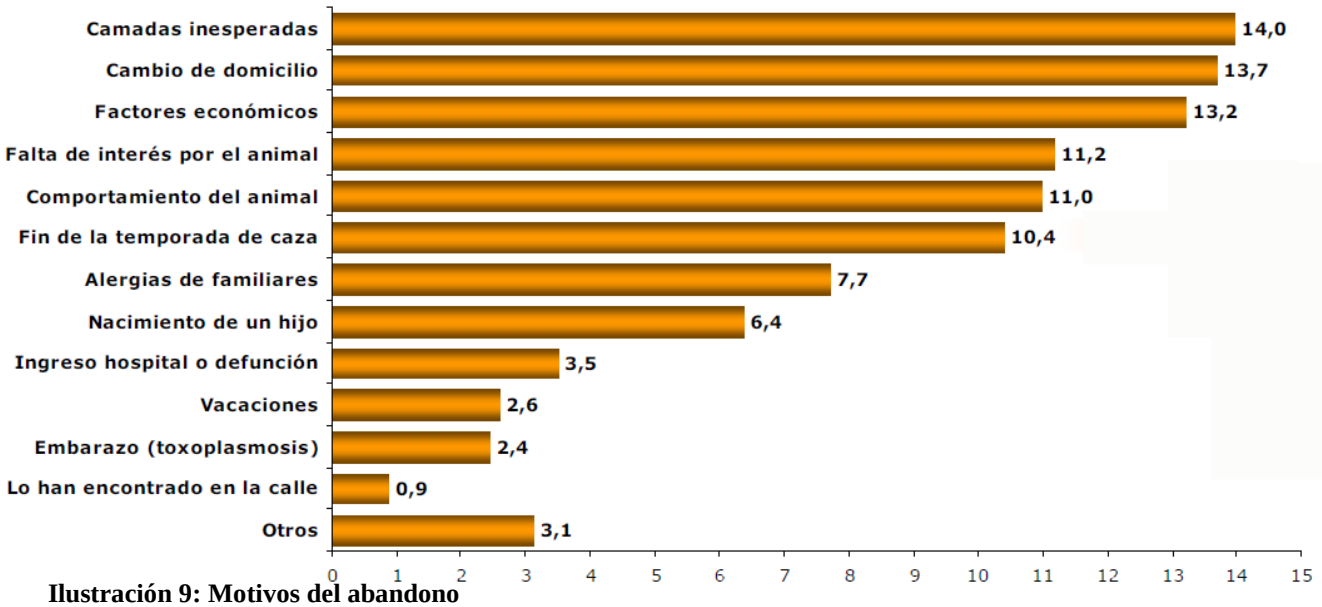


Ilustración 9: Motivos del abandono

2.3. Tecnología Móvil

La telefonía móvil ha cambiado la sociedad actual, los terminales ofrecen unos servicios similares a los ordenadores personales, pero con la posibilidad de llevarlos siempre encima, esto permite un gran abanico de aplicaciones más cercanas a los usuarios.

Origen:

La historia del teléfono móvil se inicia sobre los años 40 cuando la compañía Motorola lanza su primer modelo el Handie Talkie, para la comunicación a distancia entre las tropas militares, la trasmisión se basaba en ondas de radio de unos 600KHz, posteriormente en la década de los 50 a 60 se extendió su uso con la comercialización de los llamadas Walkie-Talkie, utilizados por la policía, bomberos ambulancias, y otros servicios de emergencias.

Ilustración 10: Handie Talkie

En 1973 se desarrolló el teléfono móvil analógico Motorola DynaTac 800x, este teléfono media unos 33 cm de largo, pesaba unos 800 gramos, era capaz de almacenar unos 30 números de teléfono, y tenía 1 hora de autonomía en uso, y 8 horas en standby.

Ilustración 11: Motorola DynaTac 800x

Primera generación (1G):

En 1981 Ericsson lanza el móvil analógico NMT450, utilizaba frecuencias alrededor de 450MHz con modulación de frecuencia FM, las conexiones eran un tanto precarias, solo transferían voz y estaba plagado de imprecisiones. En 1986 Ericsson lanzo un modelo que soportaba frecuencias de hasta 900MHz, posibilitando el servicio a un mayor número de usuarios, así como mayor portabilidad de los terminales.

Ilustración 12: Ericsson NMT450

Segunda generación (2G):

Para mejorar el servicio se comenzó a digitalizar las comunicaciones y flexibilizar los recursos, las operadoras pasaron a transmitir de manera simultánea varias conversaciones, incrementando el número de usuarios recurrentes, se mejoró la calidad de las comunicaciones, nació el servicio de SMS, y se simplifico la construcción de terminales. EN la década de los 90 se desarrollaron varios estándares para la comunicación móvil, como el GSM, el D-AMPS o el PDC. Cabe destacar el terminal Simon Personal Comunicator de IBM y BellSouth este terminal disponía de funciones de teléfono, calculadora, libreta de direcciones, correo y fax, también se debe mencionar el Motorola StarTAC un terminal con un diseño ligero.

Generación de transición (2.5G):

En esta generación se mejora la velocidad de transferencia de datos con las tecnologías de GPRS y EDGE, además se añaden nuevos servicios como el EMS, que permite la inclusión de melodías e imágenes en los SMS, y el MMS, que permite la transferencia de imágenes, sonidos, videos y texto.
Con respecto a los terminales, se redujo su tamaño Nokia 6110 (13cm y 137 gramos) y Nokia 8210 (10cm y 79 gramos).

Ilustración 13: Distintos terminales 2º Generación

Tercera Generación (3G):

Nace la necesidad de más servicios como la conexión a internet, la videoconferencia y la descarga de archivos, en este momento la tecnología ya permite un sistema nuevo para aumentar la capacidad de transmisión de datos mediante el UMTS con capacidades desde 144Kbit/s hasta 7.2Mbit/s.
En cuanto a terminales la gran revolución la provoco Apple con su iPhone, este dispositivo tenía una pantalla táctil que permitía dejar atrás el teclado, podía voltear la pantalla para ver el contenido horizontal y verticalmente, disponía de una cámara fotográfica, sincronización con otros servicios (iTunes), una entrada de auriculares y conectividad WIFI, este terminal marcó el camino para los dispositivos actuales, que permiten convergencia de voz y datos a gran velocidad, reproducir música, realizar videoconferencias, disponer de infinidad de aplicaciones y poder disfrutar de una movilidad total.

Ilustración 14: Distintos terminales 3º Generación

Sistemas Operativos Móviles

Android:

En el año 2005 Google adquiere Android Inc, una compañía recién creada, orientada a la producción de aplicaciones móviles, ese mismo año Google empieza a trabajar en la creación de una máquina virtual java optimizada para móviles.
En 2007 se crea el consorcio Handset Alliance con el objetivo de desarrollar estándares abiertos para móviles, esta plataforma se marca como objetivo el diseño y difusión de la plataforma Android, publicando una parte importante de su propiedad intelectual como código abierto bajo licencia Apache v2.0. En noviembre de 2007 se lanza la primera versión Android SDK, al siguiente año aparece el primer móvil con androide ñ T-Mobile G1, en octubre libera el código fuente del sistema, y se abre el Android Market, en abril de 2009 se lanza la versión 1.5 del SDK que incorpora la nueva característica del teclado en pantalla.
En 2011 se lanza la versión SDK 3.0, optimizada para el uso en tabletas.

Las características que lo hacen especial, son:

- Una plataforma de desarrollo libre basada en Linux y código abierto, se puede usar y customizar sin pagar royalities.
- Las aplicaciones finales, son desarrolladas en Java lo que nos permite una gran portabilidad gracias a la máquina virtual.
- Arquitectura basada en componentes inspirados de internet.
- Filosofía de dispositivo siempre conectado a internet.

- Gran cantidad de servicios incorporados: GPS, Posicionamiento por torres de telefonía móvil, bases de datos SQL, reconocimiento y síntesis de voz, navegador, mapas, etc.
- Un alto nivel de seguridad ya que los programas se encuentran aislados gracias al concepto de ejecución dentro de una caja que incorpora la máquina virtual, cada aplicación tiene unos permisos diferentes.
- Optimización para baja memoria y potencia.
- Alta calidad de gráficos y sonido OpenGL H.264 (AVC), MP3, ACC, etc.

El sistema Android corre sobre un núcleo Linux que se encarga de la seguridad, el manejo de memoria, el multiproceso, la pila de protocolos y el soporte de drivers para los dispositivos, esta capa actúa como una capa de abstracción entre el hardware y el resto del sistema.
El runtime de android es la máquina virtual Dalviik, que está basada en una máquina virtual Java optimizada para poca memoria y un procesador limitado, esta máquina ejecuta ficheros Dalvik ejecutables (.dex) un formato optimizado para ahorrar memoria, está basada en registros, y cada aplicación corre sobre su propio proceso en Linux con su propia instancia de la máquina virtual.
El runtime de Android también se incluye el "core libraries" con la mayoría de las librerías de Java disponibles, pudiendo utilizar una ingente cantidad de librerías y códigos ya desarrollados.

El sistema Android también incluye una gran variedad de librerías nativas en C/C++ como System C library, una derivación de la librería BSD de C estándar (libc), el Media Framework que soporta códecs de reproducción y grabación en multitud de formatos, Surface Manager, que maneja el acceso al subsistema de representación gráfica 2D y 3D, WebKit soporta el navegador, SGL, motor de gráficos 2D, OpenGL para la utilización de la aceleración hardware 3D, FreeType, fuentes en bitmap y renderizado vectorial, SQLite, potente motor de bases de datos relacionales, SSL, que proporciona servicios de encriptación Secure Socket Layer.

Ilustración 15: Andoid 2.2 (Froyo)

iOS:

En enero de 2007 Apple reveló la existencia del IPhone OS, un sistema operativo escrito en C/C++ y Objetive-C derivado de Mac OS X que está basado en Darwin BSD y por lo tanto un sistema operativo Unix.
iOS tiene cuatro capas de abstracción, una capa es el núcleo del sistema que contiene las características a bajo nivel como los ficheros del sistema, el manejo de memoria, la seguridad y los drivers del dispositivo, una capa de servicios principales fundamentales para el uso por parte de las aplicaciones, la capa de Medios que provee de gráficos y multimedia a la capa superior y la capa Cocoa Touch que contiene todas las clases para el desarrollo del interfaz de usuario, así como el acceso y manejo de objetos y servicios del sistema.

Algunas características de este sistema, son:

- Solo es soportado por el hardware especifico de Apple.
- Es software con licencia propietario.
- Su interfaz gráfica está diseñada para el touch screen, con capacidad para gestos multitouch.
- Su interfaz está constituida básicamente de sliders, interruptores y botones con una respuesta inmediata y fluida.
- Incluye múltiples aplicaciones para gestionar emails, fotos, cámara, mensajes, clima, notas
- Soporta multitarea.
- No tiene soporte para flash y Java.

Ilustración 16: iOS en iPhone y iPad

Blackberry OS:

Su desarrollo se remonta a 1999, este sistema en sus inicios permitía el acceso al correo electrónico, navegación web y sincronización con programas como Microsoft Exchange o Lotus Notes.
Su núcleo una máquina virtual java, y está escrito en Java y C++, tienen una licencia de propietario.
Este sistema está orientado a uso profesional, con acceso y sincronización a correo, calendario, tareas, notas y contactos.
Entre sus características más destacables es que el sistema permite multitarea y tiene diferentes métodos de entrada desarrollados y adaptados por RIM, como el trackball, y el teclado QWERTY.

Ilustración 17: Terminal Blackberry con Blackberry OS

Windows Phone:

Sucesor del Windows mobile que estaba enfocado al entorno empresarial, fue presentado en 2010, está basado en el núcleo del Windows Embedded CE 6.0, este sistema tiene licencia Microsoft (EULA).
Cuenta con la interfaz Metro (adoptado por Windows 8) , que ofrece accesos directos personalizados a la información útil del usuario, utiliza una interfaz multitáctil.
Cuenta con el navegador Internet Explorer, permite la búsqueda por voz, escaneo, traducción de texto o búsqueda de libros por códigos de barras mediante el buscador Bing que esta latamente integrado en el SO.
Permite desarrollar aplicaciones mediante el framework SilverLight que incluye el .NET Compack Framework que es una adaptación del .NET Framework, o mediante el Microsoft XNA Framework que incluye un conjunto de bibliotecas para el desarrollo de juegos.

Ilustración 18: Terminales con Windows Phone 8

Firefox OS:

Anunciado en febrero de 2013, escrito en HTML, Javascript y CSS, dispone de un núcleo Linux, es de código abierto.
Este sistema posee un kernel Linux, y una capa de abstracción de hardware, el entorno de ejecución es el Gecko, el mismo que utiliza el navegador Firefox, consiste en una serie de pilas de gráficos, un motor de dibujado y una máquina virtual de Javascript, soporta los estándares HTML 5, CSS y Javascript, la interfaz está escrita en estos lenguajes.
Sistema operativo recién nacido.

Ilustración 19: Firefox OS

Sensores:

Acelerómetro:

Mide la aceleración y las fuerzas inducidas por la gravedad, detecta la aceleración a la que se somete una masa, al cambiar de velocidad esta, así como la inclinación del teléfono respecto a la tierra.
Se construye sobre 3 ejes X, Y, Z.
Su uso más común es para saber si el dispositivo esta en horizontal, vertical o inclinado, y asignar funcionalidades al hacer movimientos específicos en el teléfono.

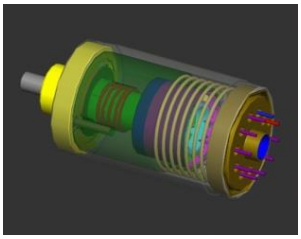


Ilustración 20: Acelerómetro

Sensor de campo magnético:

Detecta la fuerza y dirección de un campo magnético utilizando materiales magneto-sensitivos, su resistencia eléctrica cambio según el ángulo de incidencia con el campo.
Se construye sobre 3 ejes.
Se usa como brújula, detector de metales y campos eléctricos.

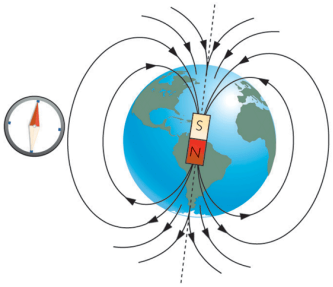


Ilustración 21: Campo magnético terrestre

Sensor de orientación:

Con la combinación del acelerómetro (vector al centro de gravedad de la tierra) y el sensor de campo magnético (vector hacia el polo norte), podemos saber a dónde apunta el teléfono en sus tres ejes.
Su principal funcionalidad es la realidad aumentada.

Ilustración 22: Ejes del terminal

El Giroscopio:

Detecta la rotación a la que es sometido el dispositivo a través del principio de conservación de la cantidad de movimiento de un cuerpo.
Sus utilidades prácticas son mejorar la prestación del sensor de orientación, y medir los giros para el navegado de un automóvil.

Ilustración 23: Giroscopio terminal móvil

Sensor de luz:

Detecta la luz ambiental a través de fotorresistencias, su utilidad principal es ajustar automáticamente el brillo de la pantalla.

Ilustración 24: Sensor de luz de un terminal móvil

Sensor de proximidad:

Detecta si hay un objeto a menos de 5 cm utilizando luz infrarroja que mide el sensor de luz, se utiliza para apagar la pantalla cuando hablamos por teléfono

Ilustración 25: Sensor de proximidad de terminal móvil

Sensor de presión:

Mide la presión atmosférica a través de piezoresistencias, su principal utilidad es de barómetro y altímetro.

Sensor de temperatura:

Detecta la temperatura interna a través de un transistor integrado, su principal utilidad es prevenir averías por sobrecalentamiento.

Localización:

GPS:

Necesita la señal de 3 satélites diferentes para la localización 2D (latitud y longitud), y la de 4 satélites para la localización 3D (altitud), cuando más señales recibe de diferentes satélites se incrementa la precisión, la precisión típica es de entre 20m-50m, la máxima precisión disponible son 10m.
La localización por GPS tarda desde 30 segundos a 12 minutos en actualizarse, no sirve en interiores, y en la ciudad los edificios interfieren con sus señales, además requiere un gran consumo de batería.

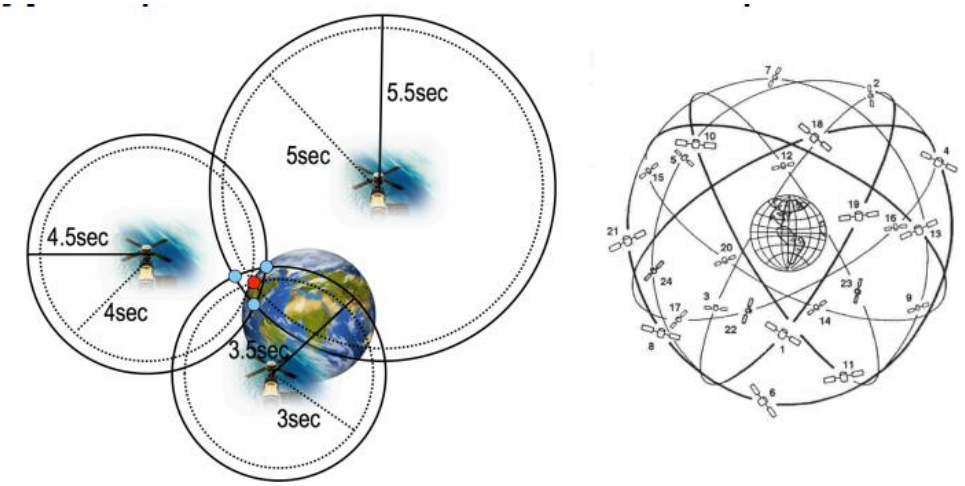


Ilustración 26: Sistema de posicionamiento GPS

Estaciones telefónicas fijas:

Para la localización mediante torres telefónicas, se obtiene el tiempo que tardan distintas señales desde diferentes estaciones fijas y así obteniendo la posición del terminal, la precisión es de unos 50m. Los obstáculos (montañas, edificios) reflejan las ondas, ocasionan una ampliación del tiempo de propagación que influye negativamente en los resultados del cálculo de la distancia.

WIFI:

Cuando un teléfono enciende la antena WIFI, con los permisos necesarios, este obtiene todas las MAC Addres de los routers visibles, este número es único y nunca se modifica, cuando obtiene los datos obtiene la localización por GPS y envía las MAC Address y la posición obtenida mediante GPS a los servidores de Google Geolocation, Skyhook, Microsoft Bing Geocode, etc.
Cuando un terminal necesita la obtener su localización, obtiene las redes disponibles mediante la antena WIFI, manda los datos a los servidores anteriormente mencionados, y estos le devuelven su localización.

2.4. Aplicaciones Similares (Redes Sociales):

Facebook:

Dog Missing / Pet Lost & Found Alert Group
(https://www.facebook.com/groups/198089680645/?fref=ts)

Grupo abierto con 8860 miembros, búsqueda de perros y gatos perdidos, abarca todo el territorio de EEUU, seguimiento por repetición.

Perros y Gatos perdidos
(https://www.facebook.com/pages/Perros-y-Gatos-Perdidos/158922580855399)

Página, mayoritariamente búsqueda y alguna adopción de perros y gatos perdidos, ubicado en México, sin seguimiento.

Busco a mi perro Simon
(https://www.facebook.com/pages/Perros-y-Gatos-Perdidos/158922580855399)

Grupo abierto con 136 miembros, exclusivo para la búsqueda de un animal extraviado en Argentina, con mayor seguimiento de la mascota ya que es exclusivo para encontrarla, publicaciones de otros temas.

Animal Search UK
(https://www.facebook.com/AnimalSearchUK?fref=ts)

Perfil para la búsqueda principalmente de gatos y algún perro, abarca todo el territorio de Gran Bretaña, escaso seguimiento de las mascotas.

Mascotas perdidas en España
(https://www.facebook.com/groups/163544690494176/)

Grupo abierto con 35 miembros, búsqueda y adopción de perros y gatos, es un grupo con una localización dispersa ya que engloba a todo el territorio nacional, no se hace ningún seguimiento de los animales.

Mascotas de toda España
(https://www.facebook.com/groups/134286866609815/?fref=ts)

Grupo abierto con 940 miembros, búsqueda y adopción de perros y gatos, como el anterior es un grupo que abarca todo el territorio nacional, y no es posible un seguimiento de los animales.

Mascotas difusión toda España
(https://www.facebook.com/groups/186432178149594/?fref=ts)

Grupo cerrado con 1208 miembros, adopción de mascotas y ayuda a protectoras y asociaciones. Abarca todo el territorio nacional, y no es posible un seguimiento de los animales.

SOS Pastores Alemanes
(https://www.facebook.com/groups/126792704062059/)

Grupo abierto con 1484 miembros, búsqueda y adopción de principalmente Pastores Alemanes, abarca todo el territorio nacional, sin seguimiento.

Adopta Salva una vida – Valencia
(https://www.facebook.com/groups/440588839294514/)

Grupo abierto con 972 miembros, búsqueda y adopción de perros y gatos, abarca principalmente Valencia aunque no hay control del origen de las publicaciones, sin seguimiento.
Este grupo dispone de una página web (http://adoptasalvaunavida.com/), donde hay una opción de búsqueda y seguimiento de animales, abarca el territorio de Valencia.

Lacua Protectora de animales de Alzira
(https://www.facebook.com/lacuaprotectoradeanimales.dealzira?fref=ts)

Perfil con 2.256 amigos perteneciente a una protectora, búsqueda y adopción de gatos y perros, seguimiento por repetición, y abarca animales de la zona de Alzira.

Ribercan – Proteger y Defender
(https://www.facebook.com/ribercan?fref=ts)

Perfil de una protectora ubicada en la Ribera Alta (Valencia), las publicaciones son principalmente de la zona, aunque sin control, búsqueda y principalmente adopción y apoyo a la protectora, seguimiento por repetición.

Google +

Gold coast Lost and found Pets
(https://plus.google.com/u/0/communities/116712528925230303560)

Comunidad con 9 miembros, ubicada en Asutralia, muy pocas publicaciones para valorar.

Pets Help Community
(https://plus.google.com/u/0/communities/116929281277306930494)

Comunidad con 42 Miembros, ubicada en Italia, búsqueda y adopción de animales, pocas publicaciones para valorar.

Twitter:

@SOS_ANIMALES
(https://twitter.com/SOS_Animales)
1729 seguidores, búsqueda y adopción de perros y gato, se entremezclan muchos temas como consejos para el cuidado de estos, ubicado en chile, imposible el seguimiento.

@AnimalesAbandon
(https://twitter.com/AnimalesAbandon)
63 seguidores, búsqueda y adopción de perros y gato, se entremezclan muchos temas como consejos para el cuidado de estos, sin ubicación, imposible el seguimiento.

@HelpingLostPets

(https://twitter.com/HelpingLostPets)
1702 Seguidores, búsqueda de gatos y perros perdidos, publicaciones bastante uniformes en cuanto al tema, con redirección a la descripción y foto de la mascota, ámbito geográfico, EEUU y Canadá.

@LostPetFinder
(https://twitter.com/lostpetfinder)
569 Seguidores, búsqueda de gatos y perros perdidos, ámbito geográfico estado de Tejas, uniforme en cuanto a publicaciones.

@PetHelpers
(https://twitter.com/PetHelpers)
1516 Seguidores, adopción de perros y gatos, publicaciones bastante uniformes, ámbito geográfico estado de california

@AdoptaMascotas2
(https://twitter.com/AdoptaMascotas2)
1405 seguidores, adopción de perros y gatos más apoyo a la asociación, publicaciones bastante uniformes, ámbito geográfico Valencia.

2.5. Análisis de las aplicaciones

Dispersión: Territorio que abarca.
Seguimiento: Seguimiento de cada mascota, se busca, se ha encontrado.
Relevancia: Número de usuarios involucrados.
Seguridad: Acceso a los contenidos.

Aplicación	Red Social	Tipo	Dispersión	Seguimiento	Relevancia	Seguridad
Dog Missing / Pet Lost & Found Alert Group	Facebook	Grupo	ALTA	BAJO	ALTA	BAJA
Perros y Gatos perdidos	Facebook	Pagina	ALTA	BAJO	-	BAJA
Busco a mi perro Simon	Facebook	Grupo	BAJA	ALTO	BAJA	BAJA
Animal Search UK	Facebook	Perfil	ALTA	BAJO	-	BAJA
Mascotas perdidas en España	Facebook	Grupo	ALTA	BAJO	BAJA	BAJA
Mascotas de toda España	Facebook	Grupo	ALTA	BAJO	MEDIA	BAJA
Mascotas difusión toda España	Facebook	Grupo	ALTA	BAJO	ALTA	ALTA
SOS Pastores Alemanes	Facebook	Grupo	ALTA	BAJO	ALTA	BAJA
Adopta Salva una vida – Valencia	Facebook	Grupo	MEDIA	ALTO	MEDIA	BAJA
Lacua Protectora de animales de Alzira	Facebook	Perfil	BAJA	BAJO	MEDIA	BAJA
Ribercan – Proteger y Defender	Facebook	Perfil	BAJA	MEDIO	BAJA	BAJA
Gold coast Lost and found Pets	Goggle +	Comunidad	ALTA	-	-	BAJA
Pets Help Community	Goggle +	Comunidad	ALTA	-	-	BAJA
@SOS_ANIMALES	Twitter	-	ALTA	BAJO	ALTA	BAJA
@AnimalesAbandon	Twitter	-	-	BAJO	BAJA	BAJA
@HelpingLostPets	Twitter	-	ALTA	BAJO	MEDIA	BAJA
@LostPetFinder	Twitter	-	MEDIA	BAJO	ALTA	BAJA
@PetHelpers	Twitter	-	MEDIA	BAJO	ALTA	BAJA
@AdoptaMascotas2	Twitter	-	MEDIA	BAJO	ALTA	BAJA
@MASCOTASDECALLE	Twitter	-	ALTA	BAJO	ALTA	BAJA

Ilustración 27: Comparativa aplicaciones

2.6. Conclusiones

En esta sección hemos podido apreciar la problemática de los animales abandonados en nuestro territorio a través del estudio de la fundación Affinity. Y observar que con la cantidad de diferentes herramientas que se utilizan para aplacar este problema, como son páginas web o mayoritariamente redes sociales, no son eficaces para alcanzar nuestro propósito, también podemos observar la gran cantidad de información que se genera de animales abandonados y perdidos, viendo que esta pasa a ser obsoleta rápidamente, por falta de herramientas adecuadas.

También se hace un repaso por las tecnologías móviles disponibles en estos momentos y los distintos sensores que estos poseen, que pueden ser utilizados para alcanzar el objetivo de encontrar y ubicar los animales perdidos y abandonados.

En el siguiente capítulo se explicara que herramientas y tecnologías se utilizarán para llevar a cabo la aplicación.

3. Entorno de desarrollo

3.1. Introducción

En este apartado veremos que tecnologías y herramientas que aprovecharemos para llevar a cabo esta aplicación, haciendo una breve introducción a las más destacadas.

3.2 Herramientas

Android-SDK
Es un paquete de software que incorpora todas las herramientas necesarias para desarrollar aplicaciones Android, este incluye librerías, documentación, conversor de código, depurador, emulador y ejemplos de código.

ADT-Bundle
Es un paquete ofrecido por google de forma gratuita, este IDE basado en el IDE Eclipse ofrece un editor de texto, asistentes y recursos que integran las herramientas ofrecidas por el Android-SDK.

Genymotion
Es un emulador de dispositivos Android basado en el uso de máquinas virtuales x86 optimizadas para correr en Virtualbox, es capaz de crear distintos tipos de dispositivos con diferentes características, para poder testear la aplicación en diferentes terminales.

Servidor HTTP Apache
Servidor de código abierto capaz de ejecutarse en multitud de sistemas operativos que implementa el protocolo HTTP/1.1, es un servidor altamente configurable y a través de módulos se puede extender su funcionalidad.
Se usa principalmente para él envió de páginas web estáticas y dinámicas y para poner a disposición servicios como la compartición de servicios.

MySQL
Gestor de bases de datos relacional con licencia GNU GLP muy extendido gracias al fácil acceso por parte de distintos lenguajes de programación a través de interfaces propias como a través del driver ODBC, es un gestor de bases de datos muy utilizado por aplicaciones web como Wordpress.
Cabe destacar el amplio subconjunto del lenguaje SQL, gran disponibilidad en diferentes sistemas, selección del motor dependiendo de las necesidades requeridas, transacciones y claves ajenas, conectividad segura, replicación y búsqueda e indexación de campos de texto.

Otras características importantes son el uso de multihilos, tablas hash en memoria temporal, soporte para operadores, funciones y funciones agregadas, sistema de contraseñas y privilegios, conexión a través de sockets TCP/IP.

3.3 Tecnologías

Android – Arquitectura

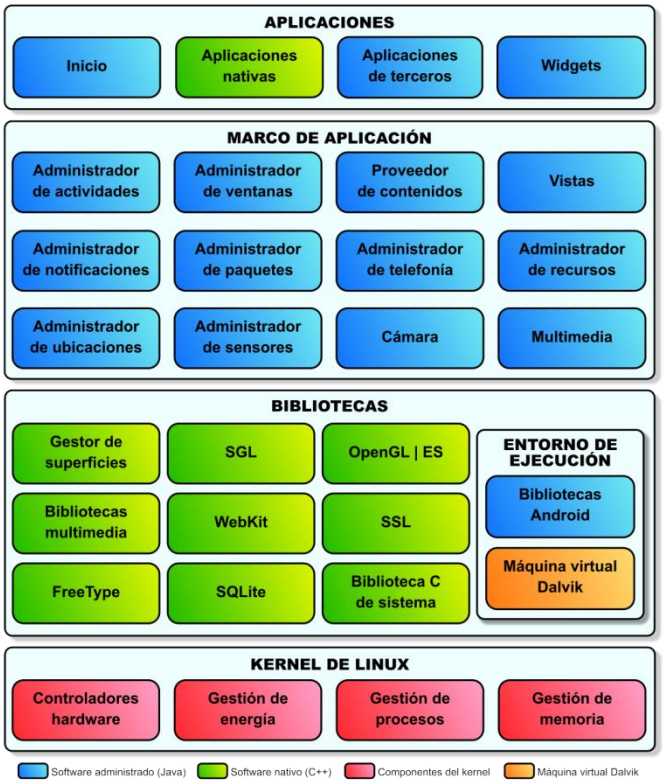


Ilustración 28: Arquitectura de Android

El núcleo de Android, está formado por un sistema operativo Linux 2.6, este se encarga de proporcionar servicios como el manejo de memoria, la seguridad, la pila de protocolos, el multiproceso y el soporte de los drivers de los distintos dispositivos.

El Runtime de Android o máquina virtual Dalvik, está basada en una máquina virtual Java adaptada para funcionar unos recursos limitados. Ejecuta ficheros con extensión .dex de formato optimizado para el ahorro de memoria y basado en registros, que permite que cada instancia en la máquina virtual corra en su propio proceso linux.

Las librerías nativas, son un conjunto de librerías en C/C++ compiladas en código nativo del procesador.
Entre estas librerías podemos encontrar:

- Media Framework que es la encargada de soportar los codecs de reproducción y grabación de los formatos más habituales de video audio e imágenes.
- Surface Manager, encargada de la representación gráfica 2D y 3D.
- WebKit/Chromium encargada de la gestión del navegador y las vistas Webview.

- SQLite un motor de base de datos relacionales.

El Entorno de aplicación, esta capa simplifica la reutilización de componentes, y proporciona a las aplicaciones una interfaz para poder obtener o generar recursos.

- Views conjunto de vistas.
- Resources Manager encargada de gestionar los recursos que no están en código.
- Activity Manager maneja el ciclo de vida de las aplicaciones.
- Notification Manager permite a las aplicaciones mostrar alertas.
- Content Providers que permite la intercomunicación entre aplicaciones.

Las Aplicaciones, que es el conjunto de aplicaciones instaladas en el dispositivo Android, estas corren sobre la máquina virtual Dalvik.

Cliente Servidor- Arquitectura

Modelo de aplicación distribuida en que las tareas se reparten entre los proveedores de recursos o servicios servidores, y los que acceden a ellos clientes.
Este modelo permite centralizar la gestión de la información y separar la capacidad de proceso para esta tarea, pudiendo asignar la parte más costosa al servidor o servidores o a los clientes.

En esta arquitectura un servidor pone a disposición unos servicios o recursos para ser accedidos a través de varios clientes.
El servidor está a las espera de solicitudes de clientes, un cliente inicia una solicitud al servidor, este procesa la solicitud mientras el cliente espera, cuando ya ha terminado el procesamiento envía la respuesta, finalmente el cliente recibe el recurso o servicio solicitado

PHP

```
<html>
 <head>
  <title>Prueba de PHP</title>
 </head>
 <body>
 <?php echo '<p>Hola Mundo</p>'; ?>
 </body>
</html>
```

Es un lenguaje de programación que se suele usar en el lado del servidor, este lenguaje es interpretado por un servidor con el modulo PHP disponible, y genera documentos HTML dinámicamente.
Este lenguaje está orientado al desarrollo de aplicaciones web dinámicas, es capaz de acceder a la mayoría de motores de bases de datos, tiene gran cantidad de módulos que expanden sus funcionalidades, es libre y posee una buena documentación.

JSON

```
{
  "arguments" : { "number" : 10 },
  "url" : "http://localhost:8080/restty-tester/collection",
  "method" : "POST",
  "header" : {
    "Content-Type" : "application/json"
  },
  "body" : [
    {
      "id" : 0,
      "name" : "name 0",
      "description" : "description 0"
    },
    {
      "id" : 1,
      "name" : "name 1",
      "description" : "description 1"
    }
  ],
  "output" : "json"
}
```

Es una notación compacta y eficiente para el intercambio de recursos.
Por su simplicidad se ha convertido en una alternativa a la notación XML, se ha extendido gracias al fácil manejo desde Javascript en envíos AJAX, y al soporte por parte de los principales lenguajes.

3.4 Conclusiones

En este apartado se han descrito las herramientas y tecnologías más importantes que se utilizaran en la aplicación.

Usaremos el entorno de desarrollo ADT-Bundle para desarrollar la aplicación Android, apoyándonos en las herramientas del SDK. Para la parte del servidor se utilizara el servidor HTTP Apache para la recolección y centralización de la información apoyándose con el gestor de base de datos MySQL para guardar y estructurar los datos.

Por otra parte usaremos las tecnología Android para realizar el cliente que se encargara de recopilar los datos mediante la acción del usuario y sensores disponibles, este se encargara de trasmitir mediante notación JSON al servidor los datos recopilados y enviar los archivos obtenidos, el servidor mediante PHP procesara la información y los archivos recibidos, y los clasificará y guardará.

4. Especificación de requisitos

4.1 Introducción

Ahora vamos a pasar a explicar las necesidades que va a tener la aplicación planteada, para describir estas necesidades emplearemos el estándar IEEE 830.

4.1.1 Propósito

En este apartado se detallarán que requisitos y funcionalidades debe poder satisfacer esta aplicación para lograr todo o parte de su propósito final, que es el de agrupar una serie de funcionalidades para mejorar la localización y búsqueda de mascotas.

Este documento servirá de referencia para la mejora o ampliación del software, así como para entender el funcionamiento para desarrollar e implementar el servicio centralizado.

4.1.2 Ámbito del sistema

La aplicación **LostPet** permitirá, para cualquier usuario, poder hacer una fotografía o descripción de un animal indicando su localización de forma sencilla, ordenada y en tiempo real, centralizando los datos, y así disponer de forma unificada datos de animales perdidos o abandonados, que simplificarán la gestión que se puede hacer de estos.

Se seguirá el estándar IEEE 830 de referencia actualmente.

4.1.3 Definiciones, acrónimos y abreviaturas

AJAX: Javascript Asíncrono y XML, Se mantienen una comunicación asíncrona con el servidor para modificar una página web de forma dinámica.

Animal abandonado: Animal del que se desconoce, o no tiene un propietario.

Animal perdido: Animal que si se conoce su propietario, pero este no conoce su ubicación.

API: Interfaz de Programación de Aplicaciones, grupo de rutinas que provee un sistema operativo, una aplicación o una biblioteca que definen como hacer uso desde un programa de sus servicios.

Aplicación: Programa informático creado para llevar a cabo una tarea en un dispositivo.

Aplicación Móvil: Programa informático diseñado para ser ejecutada en un teléfono inteligente, tableta o cualquier dispositivo móvil.

BBDD: Base de datos, es una colección de información organizada para que un sistema pueda seleccionar fragmentos de datos.

Cliente: Programa que necesita la conexión a otro programa, llamado servidor, para completar todas sus funcionalidades.

HTTP: Protocolo de Transferencia de Hipertexto, facilita la semántica y la sintaxis en la comunicación web entre clientes y servidores.

IDE: Entorno Integrado de Desarrollo, conjunto de programas que usan una interfaz única para todos ellos.

Metadatos: Datos sobre datos, información no relevantes para el usuario pero si para el sistema.

SDK: Kit de Desarrollo de Software, conjunto de herramientas y programas de desarrollo que permite crear software sobre una plataforma o sistema en concreto.

Servidor: Programa que pone servicios a disposición de otros programas, normalmente clientes.

SQL: Lenguaje de Consultas Estructurado, permite la especificación de distintas clases de operaciones hacia el gestor de la base de datos.

TCP/IP: Protocolo de Control de Transmisión/Protocolo de Internet, sistema de protocolos que hacen posible la comunicación entre dispositivos.

Ventana Modal: Ventana que no permite interactuar sobre la interfaz de usuario que no pertenezca a la propia ventana.

Vista: Parte visible de una interfaz de usuario.

XML: Lenguaje de Marcas Extensible, es un lenguaje que permite la organización y etiquetado de documentos.

4.1.4 Visión general del documento

Este apartado consta de tres partes, una en la que se proporcionará una visión general, una segunda que explicara el contexto de la aplicación, y una tercera en que se detallarán las funcionalidades que se esperan de esta.

4.2 Descripción General

Esta aplicación deberá permitir mediante la integración de diversas utilidades enviar a un servidor central imágenes y metadatos, que permitirá en un futuro, mediante nuevos desarrollos, la identificación mediante reconocimiento de imágenes, comparación y cruce de datos el seguimiento y localización de animales.

4.2.1 Perspectiva del producto

Esta aplicación intenta integrar y unificar diversos desarrollos independientes entre sí pero que tienen el mismo propósito.

La aplicación en su parte cliente estará desarrollada en Android haciendo uso de las bibliotecas ya existentes y de código abierto, y se encargara de capturar y enviar los datos a un servidor.

En su parte servidor se encargara de recibir y guardar los datos enviados, esto se realiza mediante un servidor Web Apache que recibirá mediante cadenas JSON los metadatos y las imágenes, esto se procesará mediante el lenguaje de programación del lado de servidor PHP, estos datos se guardarán en una base de datos MySQL.

La parte servidor será una parte de una solución mayor, que después de recibir los datos se encargara de identificarlos y clasificarlos como se estime más oportuno.

4.2.2 Funciones del producto

La aplicación de una manera cómoda y sencilla realizará las siguientes funciones:

- Captura de imágenes de un animal.
- Selección de imágenes ya tomadas.
- Descripción del animal (Formulario, Reconocimiento voz y descripción virtual).
- Indicar geo localización del animal.
- Identificación del usuario que hace uso de la aplicación.
- Verificación de los datos a enviar.
- Envío de metadatos e imágenes al instante o posteriormente.
- Captura de datos en un servidor centralizado.

4.2.3 Características de los usuarios

Los usuarios de la aplicación no necesitaran ningún nivel de estudios ni experiencia para poder utilizar la aplicación, solo se requerirá conocer mínimamente el funcionamiento de cualquier Smartphone, Tablet o dispositivo.
Tanto la funcionalidad como la interfaz están concebidas para un uso fácil e intuitivo.
La aplicación solo será utilizada por un único usuario, teniendo la posibilidad de identificarse.

4.2.4 Restricciones

Para poder utilizar la aplicación en la parte cliente se requerirá poseer un dispositivo con el sistema operativo Android 3.0 o superior y la posibilidad de tener en algún momento acceso a internet, ya sea 3G o WIFI.
Algunas funcionalidades pueden no estar disponibles dependiendo del hardware del dispositivo, si no tiene cámara no se podrán capturar imágenes, si no tiene GPS no se podrá obtener la localización automáticamente, y si no se dispone de internet 3G, no se podrá realizar el envío de datos en tiempo real.

En cuanto a la parte servidor será necesario disponer de un servidor web Apache con PHP y MySQL instalados, una conexión a internet y un dominio o IP fija para poder localizarlo en la red.

4.2.5 Suposiciones y dependencias

En la parte del cliente el sistema puede variar en la versión del sistema operativo soportado si se requiere una versión superior para poder implementar alguna funcionalidad o mejora critica.

En la parte servidor la aplicación puede modificase completamente prescindiendo de todos los requisitos de software especificados si los futuros requerimientos lo requieren.
La forma y formato en que los datos serán enviados no debe modificarse sustancialmente.

4.3 Requisitos específicos

Esta sección contiene los requisitos que deberá cumplir el sistema con el suficiente nivel de detalle para permitir planificar, implementar, diseñar las pruebas y validar el cumplimiento de todas las funcionalidades aquí descritas.

4.3.1 Interfaces externas

La aplicación, parte cliente, está diseñada para funcionar en cualquier dispositivo con sistema operativo Android 3.0 o superior independientemente del tamaño de su pantalla.

El sistema, parte cliente, se comunica con los distintos componentes y servicios mediante librerías internas implementadas y soportadas por el sistema operativo.

Algunas de las funcionalidades están sujetas a los permisos que el usuario aceptará al instalar la aplicación cliente, como el acceso a internet, acceso al sistema de ficheros, localización, acceso al dispositivo de captura de imágenes, etc.

Para servicio de geo localización se requiere la inclusión de una librería externa de código abierto y totalmente soportada por el sistema operativo "google-play_services_lib" así como la activación y obtención de una clave de acceso al API de GoogleMaps, este servicio está restringido a unas 15.000 peticiones diarias.
Este servicio podrá ver afectada su precisión dependiendo de los componentes hardware de que dispone el dispositivo, y de la disponibilidad de acceso a internet en el momento de usarlo.

Para el correcto envío de datos y recepción de estos tanto el servidor como el cliente debe disponer de una conexión a internet, a su vez el servidor debe disponer de una IP accesible desde el dispositivo cliente ya sea una IP estática o un dominio.

El servidor deberá tener la capacidad de poder recibir mediante conexiones HTTP y gestionar mediante un lenguaje por parte del servidor, PHP en este caso, el guardado de imágenes, así como la interpretación correcta de cadenas con formato JSON, a su vez deberá disponer de una base de datos, MySQL en este caso, en que se guardará la información.

4.3.2 Funciones

Cliente

Pantalla inicial

Descripción:
Vista inicial donde se podrá ver en la zona central distintas funcionalidades de la aplicación. Menú de la aplicación y botón siguiente.

Entrada:
Se generará un identificar ID.

Salida:
Las distintas vistas.

Restricciones:
(Ninguna)

Descripción:
Se pueden visualizar las distintas funcionalidades, cámara, formulario, sonido y descripción virtual, mediante el desplazamiento horizontal, se podrá acceder al menú así como a un botón para pasar a la siguiente funcionalidad.

Visualización de las imágenes tomadas

Descripción:
El usuario podrá ver miniaturas de las imágenes tomadas mediante la cámara o seleccionadas de la galería y un icono para añadir más imágenes.

Entrada:
Archivos de las miniaturas de las imágenes tomadas o seleccionadas.

Salida:
Visualización de las miniaturas y un icono para agregar más.

Restricciones:
Se pueden tomar o seleccionar un máximo de 12 imágenes.

Funcionamiento:
Se cargaran las imágenes disponibles desde el directorio temporal con el id generado al iniciar la aplicación, están se visualizarán en forma de tabla, y tendrá disponible un icono para añadir más, mediante la pulsación larga sobre el icono este cambiara de aspecto según este abrirá la aplicación de captura de imágenes o la de selección desde la galería.

Captura de imagen a través de la cámara

Descripción:

El usuario podrá tomar una captura del animal encontrado.

Entrada:

Imagen capturada.

Salida:

El sistema guardara la imagen tomada así como una miniatura de esta, y estará disponible en la interfaz para visualizarla.

Restricciones:

Para realizar esta función el dispositivo tiene que disponer de una cámara.

Funcionamiento:

En la vista inicial aparece un icono en forma de cámara que al pulsarlo se abrirá la aplicación de captura de imágenes, se realizara la fotografía y se aceptara, la aplicación guardara la imagen tomada en un directorio temporal creado para este uso, también generará una miniatura de la misma imagen, los archivos tendrán nombres únicos generados mediante un identificador y la fecha de sistema, en la vista inicial se podrá ver la miniatura de la imagen tomada, así como el icono de la cámara para tomar otra imagen.

Selección de imágenes a través de la galería

Descripción:

El usuario podrá seleccionar una imagen de la galería.

Entrada:

Imagen seleccionada.

Salida:

El sistema, si tiene permiso, guardara la imagen seleccionada así como una miniatura de esta, y estará disponible en la interfaz para visualizarla, si no tiene permiso se podrá visualizar una advertencia de que no se puede obtener la imagen.

Restricciones:

Se debe tener permisos para seleccionar las imágenes de ciertas partes de la galería.

Funcionamiento:

En la vista inicial aparece un icono en forma de carga que al pulsarlo se abrirá la aplicación de galería de imágenes, se seleccionara una imagen y si se tienen los permisos suficientes, la aplicación copiara la imagen tomada en un directorio temporal creado para este uso, también generará una miniatura de la misma imagen, los archivos tendrán nombres únicos generados mediante un

identificador y la fecha de sistema, en la vista inicial se podrá ver la miniatura de la imagen tomada, así como el icono de la carga para seleccionar otra imagen.

Visualización de las imágenes

Descripción:

El usuario podrá visualizar en formato grande las imágenes tomadas o seleccionadas.

Entrada:

Imagen actual.

Salida:

Vista visualización de imágenes.

Restricciones:

Debe haber al menos una imagen tomada o seleccionada.

Funcionamiento:

Al hacer una pulsación normal sobre una imagen se abrirá una vista en que se visualizara esta en formato grande, si hay más de una se podrá visualizar a través del desplazamiento horizontal.

Borrado de imagen

Descripción:

El usuario podrá eliminar una imagen tomada o capturada mediante una pulsación larga.

Entrada:

Una advertencia de que se va a proceder a borrar la imagen para confirmar o cancelar.

Salida:

Tanto si se acepta o se cancela vuelve a la vista de visualización de imágenes.

Restricciones:

Para realizar esta función debe haber al menos una imagen capturada o seleccionada.

Funcionamiento:

Mediante una pulsación larga encima de una imagen sale una alerta indicando si se está seguro de borrar la imagen, si se acepta la imagen como su miniatura se eliminan del directorio temporal.

Formulario

Descripción:

El usuario podrá indicar detalles del animal a través de un formulario.

Entrada:

Varios campos que se deben rellenar, tipo de animal, sexo, tamaño, características.

Salida:

Vista Inicial.

Restricciones:

(Ninguna)

Funcionamiento:

Se podrá visualizar un formulario que se deberá rellenar, cada campo puede ser texto libre, o elegir una opción de las disponibles.

Campos:

Tipo (perro, gato, otro).
Sexo (macho, hembra).
Tamaño (grande, mediano, pequeño)
Color (texto libre)
Raza (Distintas opciones)
Características (Texto libre)

Reconocimiento voz

Descripción:

El usuario podrá indicar detalles del animal a través del reconocimiento de voz.

Entrada:

Un icono que al pulsar sobre él se procederá al reconocimiento de voz.

Salida:

Texto reconocido y vista inicial.

Restricciones:

El dispositivo debe de tener micrófono y acceso a internet.

Funcionamiento:

La vista dispone de un icono donde al pulsar se procederá al reconocimiento de voz, a finalizar el texto reconocido se mostrará.

Menú

Descripción:
El usuario dispondrá de un menú para realizar diversas acciones.

Entrada:
Ítems del menú.

Salida:
Funcionalidad seleccionada o vuelta atrás.

Restricciones:
(Ninguna)

Funcionamiento:
El usuario dispondrá de un menú que dependiendo de la opción seleccionada se abrirá la vista correspondiente.

Preferencias

Descripción:
El usuario podrá añadir o modificar una serie de preferencias.

Entrada:
Las preferencias añadidas anteriormente.

Salida:
Vista principal.

Restricciones:
(Ninguna)

Funcionamiento:
Se abrirá una vista donde se podrán añadir o modificar las preferencias, estas preferencias son:

Envío de datos solo con WIFI: Solo se podrá realizar el envío de datos cuando este activado el WIFI.
IP de servidor (Modo debug): donde se indicara la IP donde se aloja el servidor.
Puerto (Modo debug): donde se indicará el puerto de escucha del servidor.

Botón siguiente

Descripción:

El usuario podrá indicar la finalización de la captura de datos pulsando este botón.

Entrada:

Todas las vistas individuales.

Salida:

Si hay datos rellenados vista Mapa, si no hay o no son suficientes los datos indicados un mensaje de advertencia.

Restricciones:

Hará falta un mínimo de datos indicados, una imagen, o una descripción mediante reconocimiento de voz, o tipo, sexo y tamaño del animal.

Funcionamiento:

Al pulsar sobre el botón siguiente se validará que un mínimo de datos han sido introducidos, se procederá a mover las imágenes i miniaturas de un directorio temporal a uno de imágenes validadas, y se guardarán en una base de daos interna tanto las rutas de las imágenes como los datos indicados, se mostrará la vista mapa.

Mapa

Descripción:

El usuario podrá indicar a través de un mapa la posición actual del animal.

Entrada:

Un mapa de GoolgeMaps indicando tu posición actual, y el identificador ID de los datos.

Salida:

Vista de envíos.

Restricciones:

Acceso a internet y dispositivos de localización activados.

Funcionamiento:

Se mostrara un mapa indicando, si se puede obtener, tu posición actual, si se hace una pulsación larga sobre cualquier parte del mapa se añadirá una marca que indicará la posición del animal al capturar los datos, esta posición se guardara en la base de datos apoyándose en un identificador previamente recibido. El mapa soporta desplazamiento vertical horizontal y zoom. Si se indica la posición esta se guarda en la base de datos.

Envíos pendientes

Descripción:
El usuario podrá visualizar todos los envíos pendientes de enviar al servidor.

Entrada:
Un listado con todos los envíos no finalizados.

Salida:
Salida de la aplicación.

Restricciones:
(Ninguna)

Funcionamiento:
Se mostrará un listado con todos los envíos no finalizados con las opciones de enviar, o borrar

Enviar pendiente

Descripción:
Se enviara los datos e imágenes almacenados en la aplicación al servidor.

Entrada:
Datos e imágenes no enviados.

Salida:
Envío correcto o no realizado.

Restricciones:
Acceso a internet.

Funcionamiento:
Se pulsara sobre el botón Enviar del envío pendiente, se comprobara que hay acceso a internet y se enviarán los datos e imágenes, si el servidor las acepta, las imágenes son eliminadas de la carpeta de imágenes confirmadas con sus miniaturas, y los registros de la base de datos son también removidos.
El ítem es eliminado.
Si no se puede enviar aparece un mensaje de error en el envío.
Si el envío es parcial, falta alguna imagen, se guarda en la base de datos local el id del registro en el servidor para localizar este registro en posteriores intentos.

Borrar pendiente

Descripción:
El usuario podrá eliminar un envío pendiente.

Entrada:
Datos e imágenes no enviados.

Salida:
Mensaje de confirmación de borrado y vista envíos pendientes.

Restricciones:
(Ninguna)

Funcionamiento:
Se pulsara sobre el icono borrar del envío pendiente, si se confirma el borrado, las imágenes son eliminadas de la carpeta de imágenes confirmadas con sus miniaturas, y los registros de la base de datos son también removidos.
El ítem es eliminado.

Servidor

Recoger datos

Descripción:
El servidor recogerá los datos enviados por el cliente y los guardará en la base de datos.

Entrada:
Datos e imágenes enviados.

Salida:
Mensaje de confirmación de recibido, y persistencia de datos.

Restricciones:
Conexión a internet.

Funcionamiento:
El servidor dispondrá un servicio al que el cliente enviará los datos, este al recibirlos los guardará en la base de datos o directorio si es una imagen, y enviará como respuesta la confirmación del éxito de la operación.

5. Diseño

5.1 Introducción

Esta sección describirá la arquitectura del sistema para cumplir los requisitos que se detallan el apartado anterior. Se dará una breve explicación de las características más importantes.

5.2 Capa de datos

Esta capa es la encargada de almacenar los datos del sistema y de los usuarios. A través de la capa de negocio recibe o proporciona los datos.

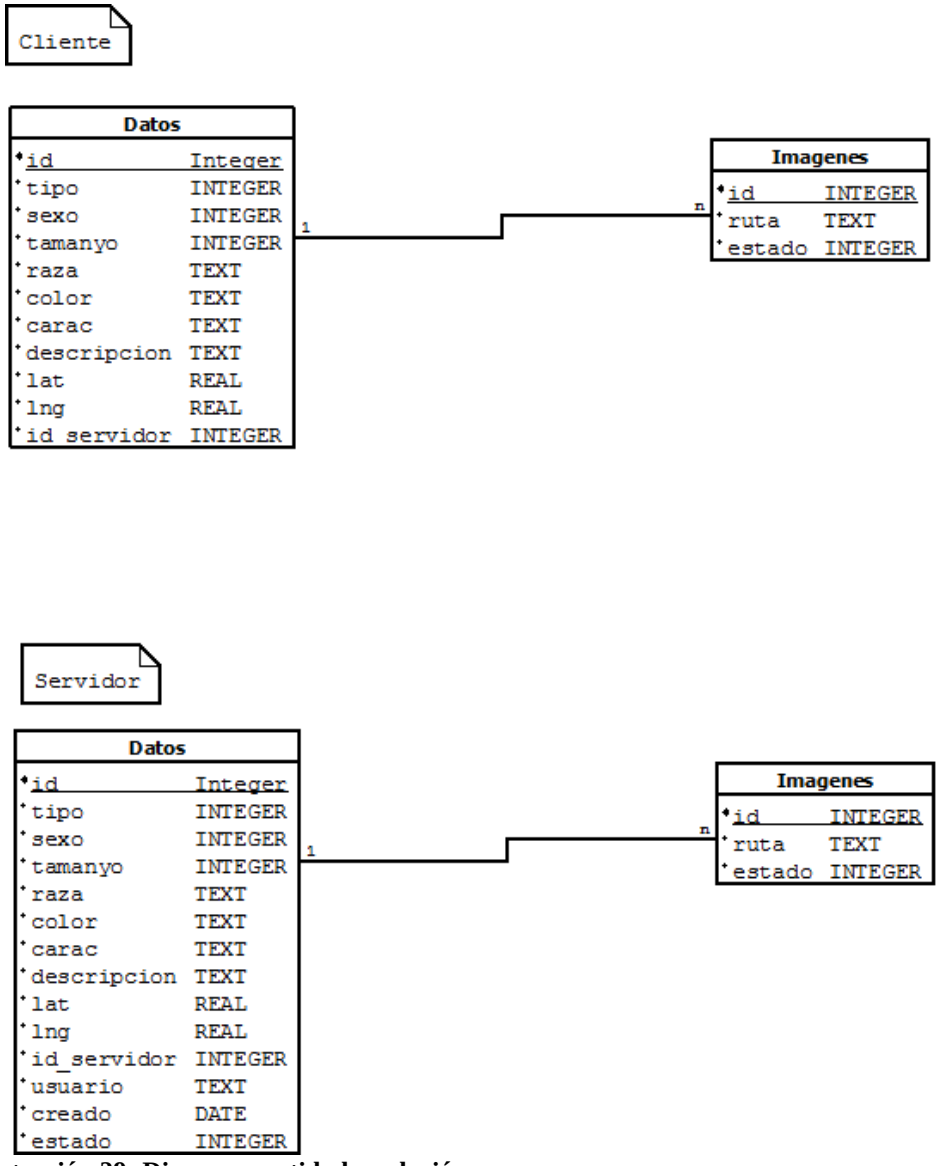


Ilustración 29: Diagrama entidad – relación

La capa de datos estará dividida en dos bases de datos diferentes.

Cliente

El cliente deberá contener una base de datos para guardar envíos pendientes.

Tabla Datos
Contiene los datos que se obtienen en la aplicación

id: identificador id como clave principal y autonumérico.
id_servidor: Entero que guarda un id proporcionado por el servidor, para conocer si los datos ya han sido enviados. En el caso que el cliente haya mandado los datos y no las imágenes este campo dice si se han de volver a enviar los datos.

Tabla Imágenes
Contiene las rutas de las imágenes

id: identificador id como clave principal y autonumérico.
estado: Entero que guarda si una imagen ha sido enviada.
id_dato: Clave foránea con la tabla Datos

La tabla Datos se relaciona con la tabla Imágenes con una relación de uno a muchos.

Servidor

El servidor contendrá los datos de varios clientes.

Tabla Datos
Contiene los datos obtenidos por los clientes.

id: identificador id como clave principal y autonumérico.
id_servidor: Entero que guarda un id autonumérico, para conocer si los datos ya han sido recibidos.
usuario: Texto para conocer que usuario ha hecho el envío.
estado: Entero para conocer si los datos son activos o pasivos (desechados).

Tabla Imágenes
id: identificador id como clave principal y autonumérico.
id_dato: Clave foránea con la tabla Datos

La tabla Datos se relaciona con la tabla Imágenes con una relación de uno a muchos.

5.3 Capa de Negocio

Es la encargada de recibir peticiones por el usuario o el sistema, procesar la información y devolver la respuesta. Esta capa se comunica con la de presentación de donde recibe o envía la información y la de datos donde almacena y obtiene los datos necesarios.

5.3.1 UML

El lenguaje de modelado UML es el estándar para modelar sistemas, este específica, describe y documenta un sistema de forma visual.

Existen varios tipos de diagramas para describir diferentes aspectos del sistema.

5.3.2 Diagrama de Casos de Uso

El diagrama de casos de uso describe los pasos que tiene un proceso, una secuencia que inicia un actor y los distintos pasos que da el sistema para proporcionar una respuesta deseada.

El **actor** a la entidad externa sea humana u otro sistema, que demanda una actividad o proceso de este.

El **caso de uso** es cada actividad o función que realiza el sistema.

Las **relaciones** son la comunicación entre actividades o entre actividades y actores.

Existen cuatro tipos de relación:

- Relación de **comunicación** es la relación que hay entre una actividad y un actor.
- Relación de **inclusión** cuando una actividad depende de otra.
- Relación de **extensión** cuando una actividad extiende el comportamiento de otra.
- **Generalización** es cuando una actividad hereda las características de otra.

Caso de uso de la aplicación cliente

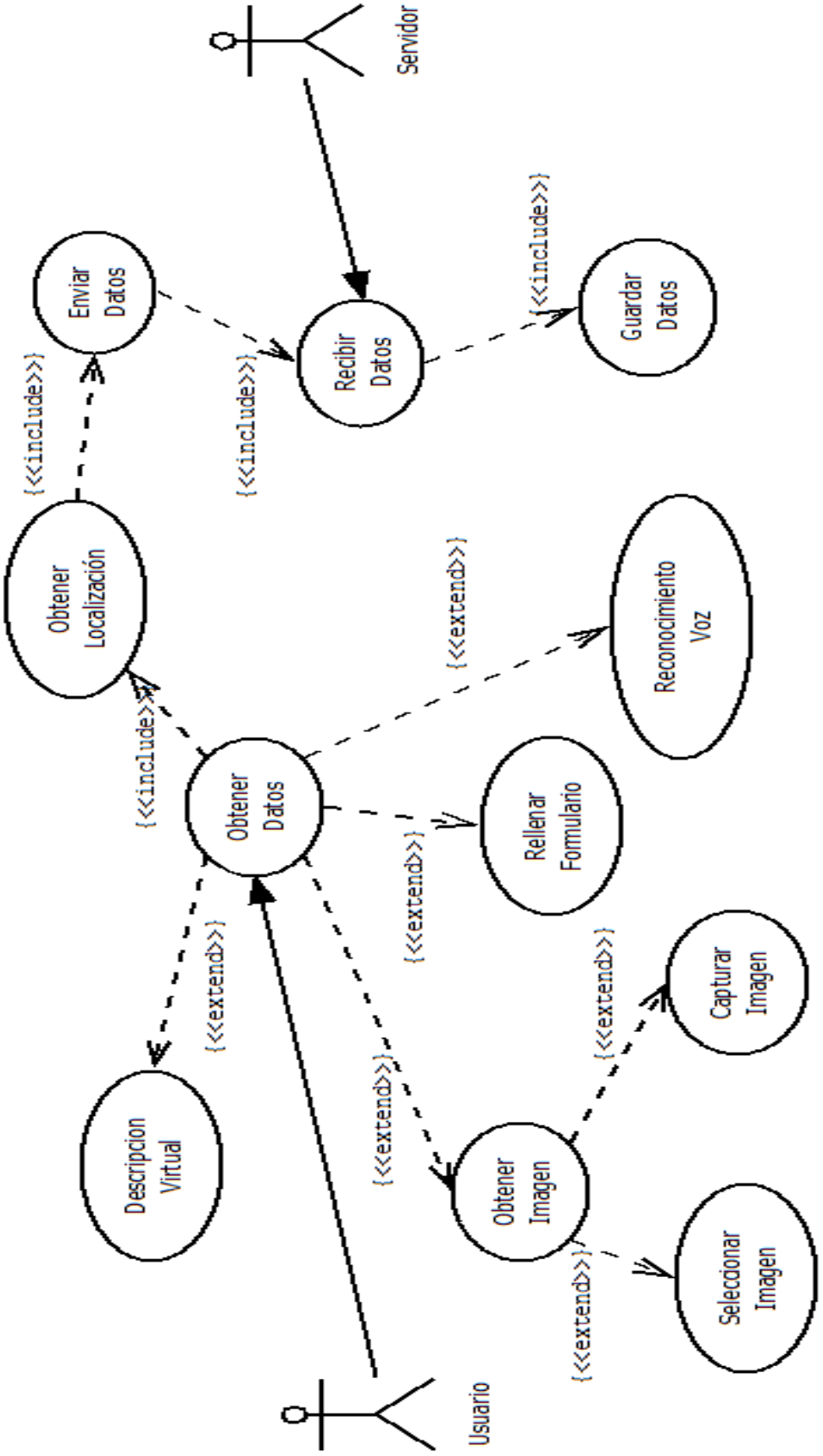


Ilustración 30: Casos de uso

El actor usuario inicia la actividad de obtener datos, estos se pueden obtener a partir de 3 vías, rellenar un formulario, reconocimiento de voz, descripción virtual, también tenemos la posibilidad de iniciar la actividad de obtener imagen, en este caso podemos elegir o hacer una captura con la cámara, o seleccionar una imagen desde la galería. Una vez terminada lo obtención de datos por parte del usuario se inicia la actividad de obtener localización, esta se encarga de recuperar las coordenadas de la ubicación, finalmente a través de la actividad enviar datos se envía los datos al actor servidor.

El actor servidor se encarga de obtener los datos y las imágenes y procesarlos, una vez guardar la información enviada por el sistema en respuesta de la actividad iniciada por el actor usuario.

5.3.3 Diagrama de clases

El diagrama de clases describe la arquitectura del sistema mostrando sus clases, las relaciones entre estas, y las propiedades y funciones de cada clase.

Es una forma fácil y clara de mostrar que clases atributos y procedimientos tendrá un sistema en su implementación, ya que en un lenguaje orientado a objetos es fácil pasar de un diagrama de clases a clases en código.

A continuación se mostrarán los diagramas de clase del sistema.

UML Inicio 1/3 Vista contenedor principal

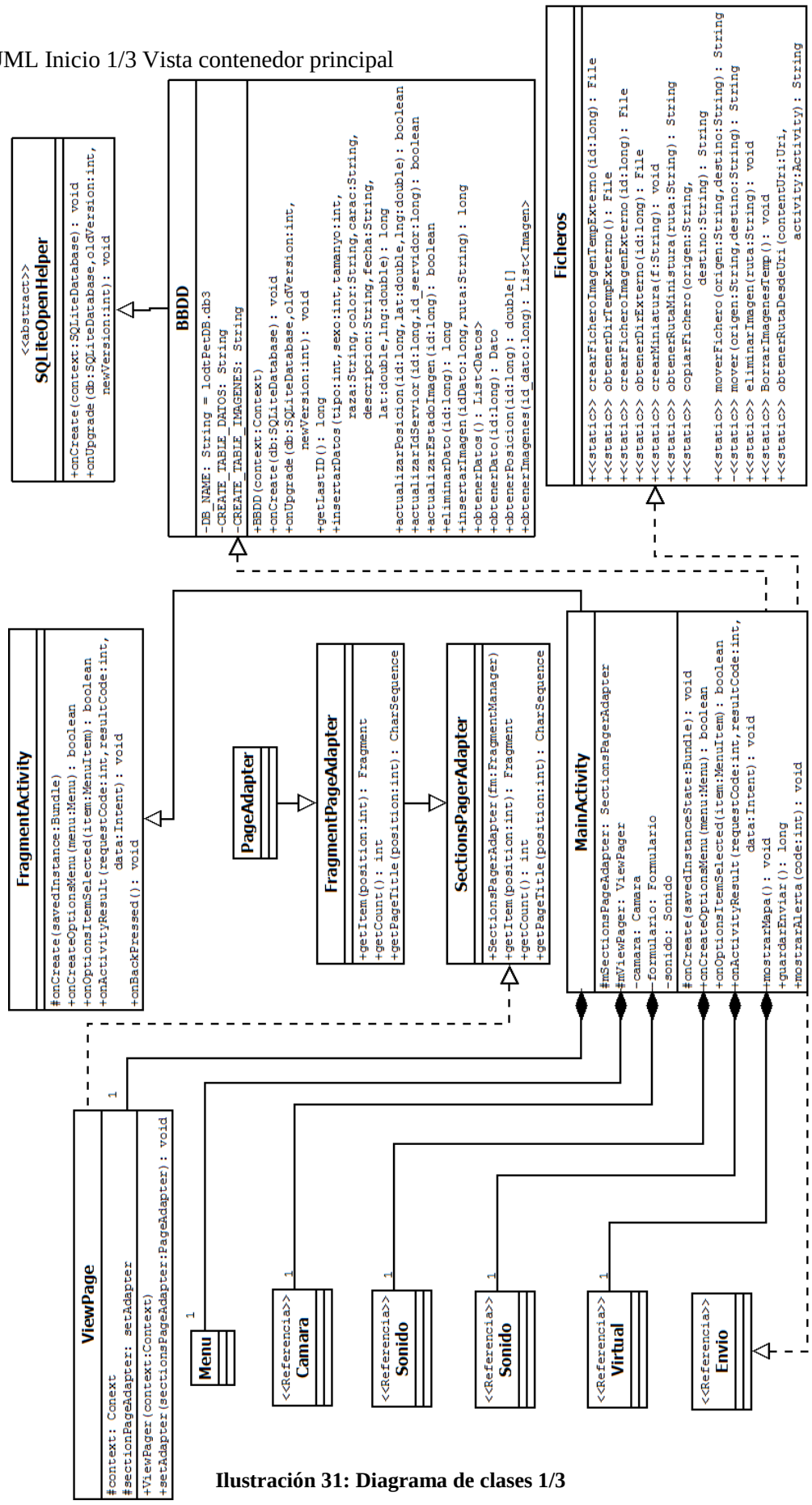


Ilustración 31: Diagrama de clases 1/3

MainActivity

La primera clase creada en la aplicación es MainActivity que hereda de FragmentActivity, clase que es capaz de contener Fragments.

MainActivity se compone de un objeto ViewPager que se encarga de la simulación del deslizado de diferentes vistas. ViewPager necesita un adaptador para posicionar y conocer los atributos de cada página, este adaptador es el SectionPageAdapter, este hereda de FragmentPageAdapter, que a su vez hereda de PageAdapter que es la clase que gestiona las vistas como páginas.

MainActivity también se encarga de crear el menú y los Fragments, y añadirlos al SectionPageAdapter, así como de gestionar la recogida, validez y persistencia de los datos, a través de la clase BBDD y la clase estática Ficheros. Se encarga de crear los dialogos modales, con el objeto Dialog, para mostrar advertencias.

Por ultimo una vez creados y validados los datos necesarios se encarga de lanzar la clase Envió pasándole un identificador obtenido desde la BBDD al guardar los datos.

Ficheros

Esta clase a través de métodos estáticos se encarga de gestionar las imágenes, haciendo las miniaturas, copiando las imágenes de la galería, guardando las imágenes tomadas de la cámara, o borrando las imágenes ya enviadas. También proporciona las rutas de acceso a las ubicaciones de las imágenes.

Todas estas funciones las hace mediante la clase File, FileInputStream y FileOutputStream, una serie de clases incluidas en Java para la gestión de archivos.

BBDD

Esta clase se encarga de la persistencia de la aplicación, crear y actualizar la base de datos.
Hereda de la clase abstracta SQLiteOpenHelper, esta clase se ocupa de la creación y actualización del esquema de la base de datos, proporciona una implementación fácil para el ContentProvider que interactúa con el gestor de base de datos SQLite.

Esta clase obtiene un objeto de la clase SQLiteOpenHelper, llamado SQLiteDatabase que proporciona varios métodos para lanzar consultas y operaciones con sentencias SQL en formato cadena, y crear las sentencias mediante paso de parámetros.

UML Inicio 2/3 Vista principal sección Formulario, Sonido y Preferencias

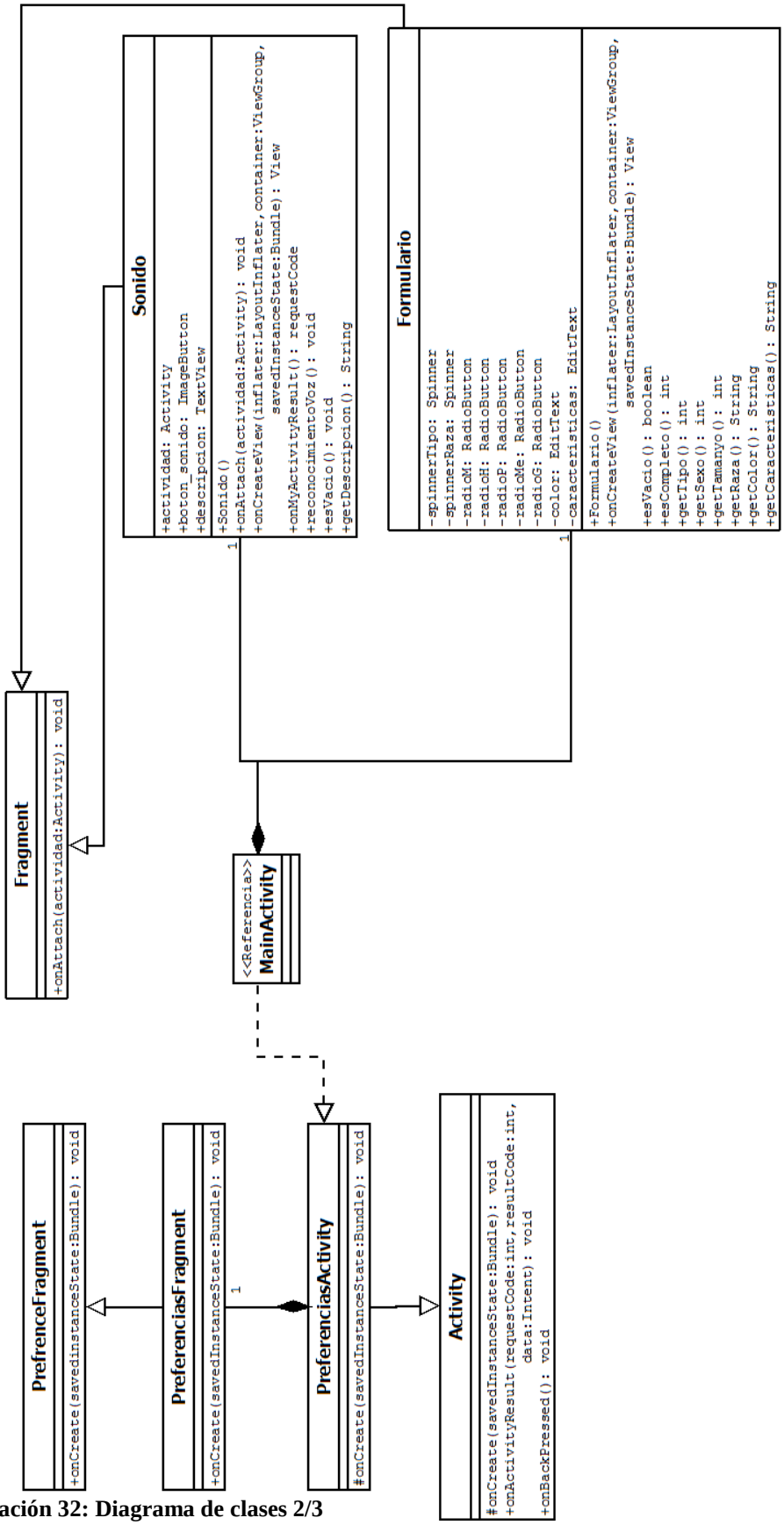


Ilustración 32: Diagrama de clases 2/3

Sonido

Clase que hereda de Fragment, para incluir su propia vista en la aplicación de forma dinámica, esta clase contiene varios objetos de tipo (Forms Widget) para mostrar los datos en estos.

Esta clase utiliza un ContentProvider para lanzar el servicio de reconocimiento de voz y obtener el texto resultado. También se encarga de validar y proporcionar los datos obtenidos.

Esta clase es utilizada desde MainActivity, que posee un objeto de Sonido.

Formulario

Clase que hereda de Fragment, para representar su interfaz dinámicamente, y utilizada desde MainActivity.

Contiene varios objetos Forms Widgets, como TextView, contenedor de texto no editable, EditText, contenedor de texto editable, Spinner, desplegable con varias opciones, este se alimenta de un recurso normalmente llamado array.xml, y RadioButton, para elegir una opción entre unas dadas.

Tiene funciones para validar sus propios datos y devolverlos.

PreferenciasActivity

Clase que hereda de Activity, ya que en realidad es otra actividad de la aplicación, crea y contiene un objeto PrefereciasFragment que hereda de PreferenceFragment. Se encarga de crear una interfaz con opciones prestablecidas en el recurso llamado preferencias.xml.
Esta persiste sus resultados internamente, gestionado por el sistema Android.

UML Inicio 3/3 Vista Cámara

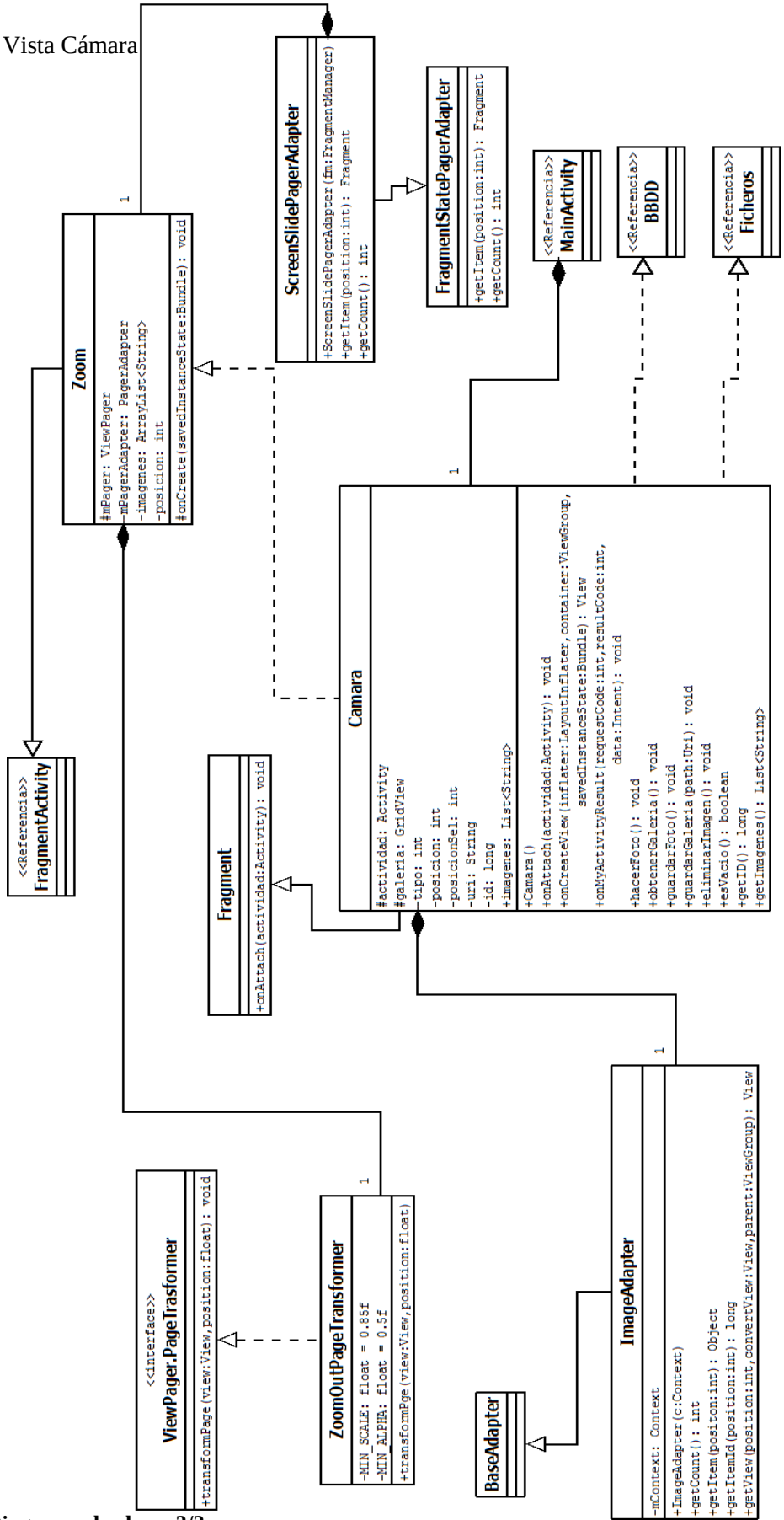


Ilustración 33: Diagrama de clases 3/3

Cámara

Clase que hereda de Fragment, que será creada y utilizada por MainActivity de forma dinámica. Para las capturas de imágenes puede utilizar o bien el ContentProvider de la cámara del dispositivo, que permite tomar la imagen utilizando la aplicación de cámara disponible en el dispositivo y recogerla, o bien tomando la imagen desde el ContetProvider de galería de imágenes.
Para mostrar las imágenes se utiliza un objeto GridView, una vista compuesta por varios componentes. Este objeto para conocer que va mostrar y cómo hacerlo requiere de un objeto BaseAdapter personalizado, así que utilizamos la clase ImageAdapter que hereda de BaseAdapter para proporcionarle la funcionalidad al GridView.

Esta clase posee funciones tanto para validar como para proporcionar las rutas de las imágenes, también se apoya en la clase BBDD para obtener un identificador libre, y con la clase estática Ficheros, que le proporciona los nombres y rutas de las imágenes obtenidas.

Zoom

Clase llamada desde la clase Camara, hereda de FragmentActivity, ya que es una actividad independiente.

Contiene un ViewPager que al igual que MainActivity permite crear la animación de deslizar las diferentes vistas de forma horizontal. Este ViewPager necesita de un PageAdapter en este caso ScreenSlidePageAdapter que hereda de FragmentStatePageAdapter, se encarga de hacer las transiciones. También utilizamos un objeto ZoomOutPageTransformer que hereda de PageTrasformer, esta clase se encarga de personalizar la animación al cambiar de vistas.

Por último la clase Zoom contiene un objeto ArrayList genérico que contendrá la lista de las rutas de las imágenes para acceder a ellas.

UML Mapa

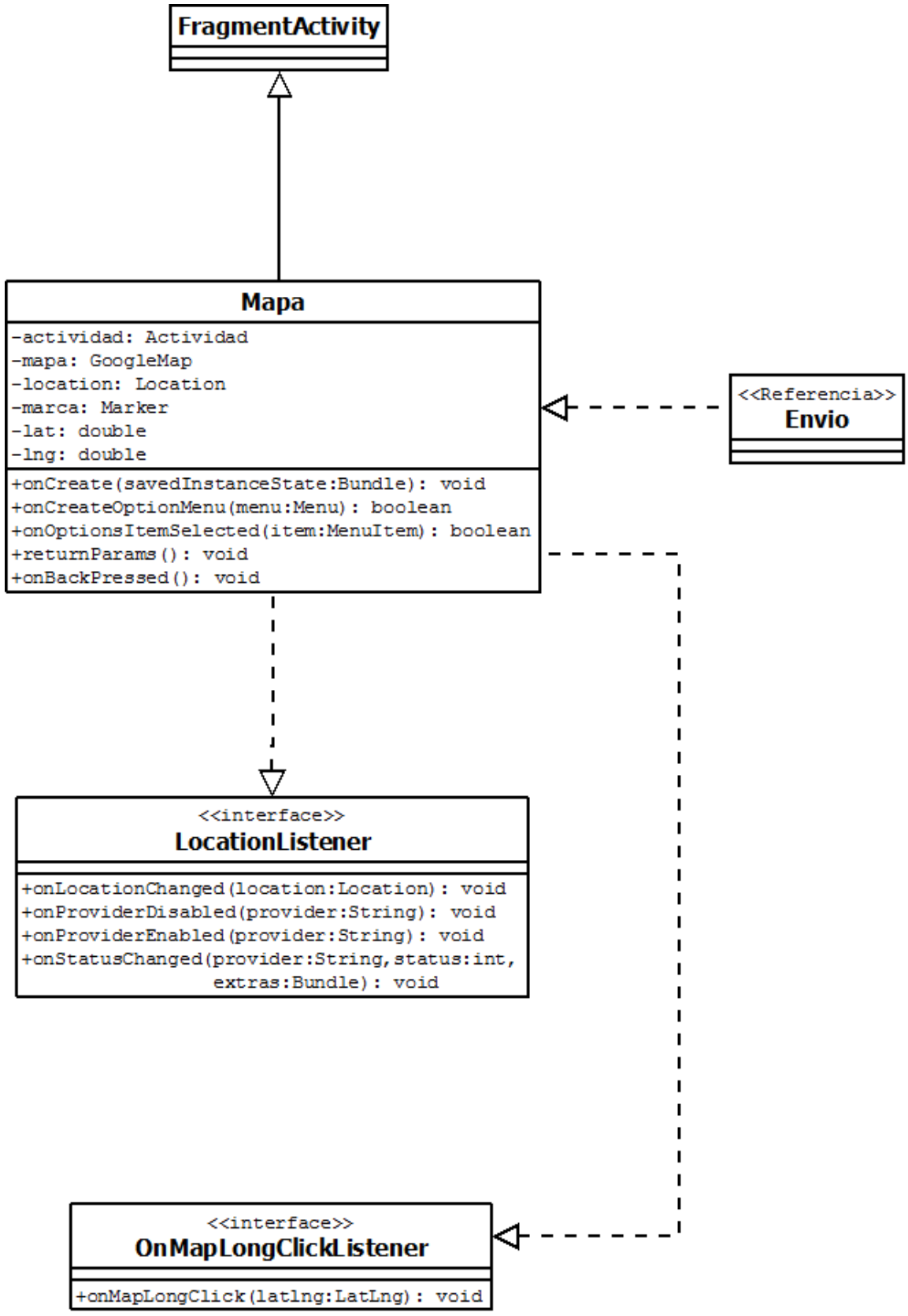


Ilustración 34: Diagrama de clases Mapa

Mapa

Clase utilizada desde la clase Envio al crearse, esta clase es la encargada de obtener las coordenadas y mostrar el mapa, esta hereda de FragmentActivity ya que es una actividad independiente.
Implementa la interfaz LocationListener, que es la encargada de obtener la localización regularmente y proporcionarla.
También implementa la interfaz OnMapLogClickListener, que se encarga de capturar las pulsaciones largas sobre el mapa.

Tiene el objeto GoogleMap que se encarga de obtener y mostrar el mapa de Google, es necesario tener la librería google-play-servie_lib asociada a la aplicación, esta librería proporciona las clases necesarias para conectar, obtener y poner a disposición de la aplicación todas las funcionalidades de GoogleMap.

También posee un objeto Location, que es el que guardará las coordenadas obtenidas. Esta clase permite mediante unciones devolver la longitud y latitud.

UML Envío

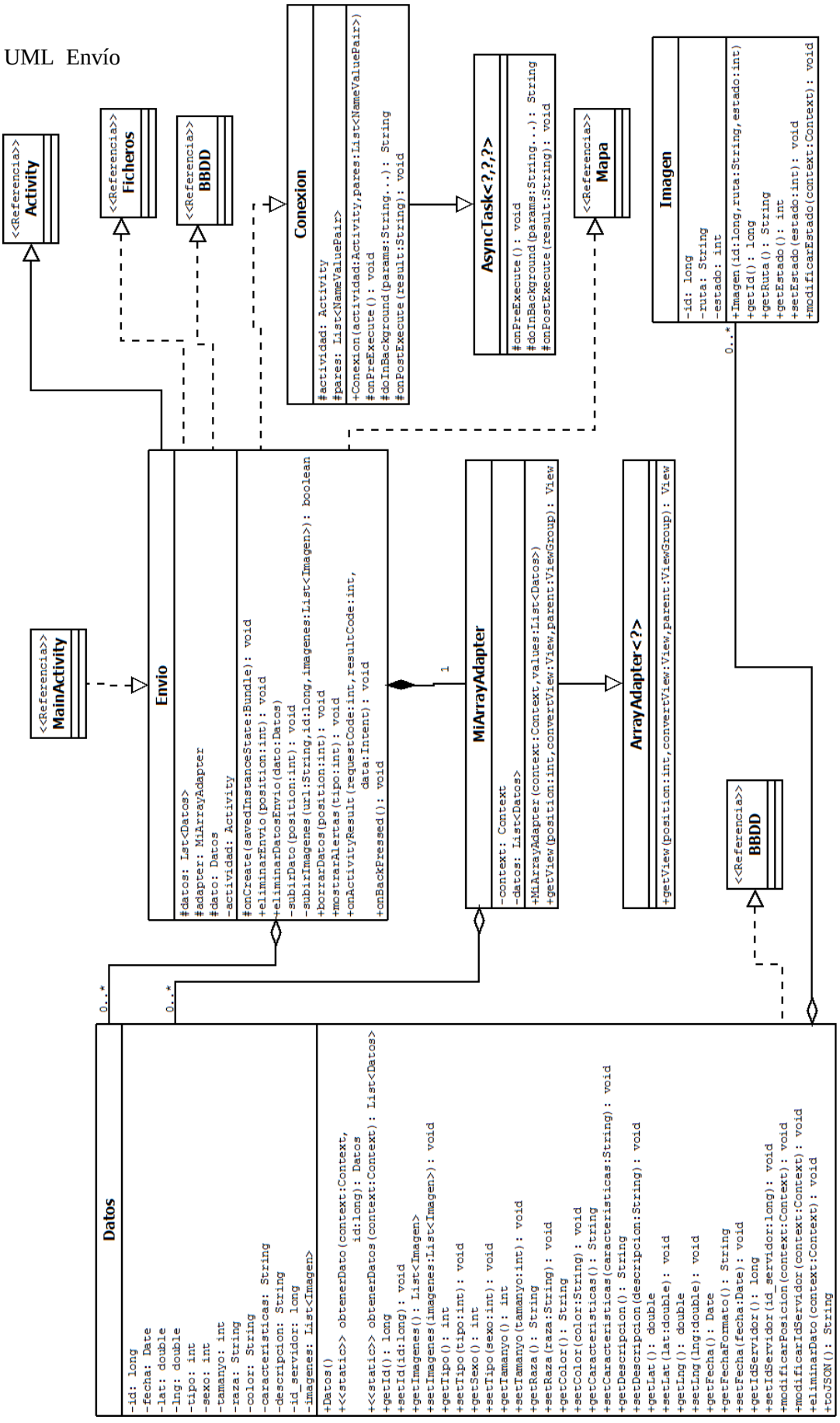


Ilustración 35: Diagrama de clases Envío

Envío

Clase que es lanzada desde MainActivity y hereda de Activity, se encarga de gestionar los envíos pendientes, así como de lanzar nuevos envíos.

Tiene un objeto ListView que crea y gestiona el listado de imágenes, para utilizar esta funcionalidad, necesita un objeto ArrayAdapter, por eso se implemente la clase MiArrayAdapter que hereda de ArrayAdapter, para crear y manejar cada posición del listado.

Utiliza las clases Mapa, BBDD, Ficheros y Datos para obtener los datos guardados para el envío, él envió lo realiza mediante la clase Conexión.

Se encarga de lanzar Mapa si es necesario, crea un objeto MiArrayAdapter que muestra los envíos pendientes, gestiona las acciones de enviar o borrar los datos, y gestionar las acciones a tomar según las respuestas del servidor.

Datos

Datos es una clase que contiene los atributos capturados por la aplicación, con un listado de objetos de tipo Imagen, esta clase tienen sus respectivos getters y setters, también tiene una función que transforma los datos de la clase en notación JSON.

Imagen

Esta clase contiene los atributos de las imágenes capturadas por la aplicación, con sus respectivos getters y setters.

Conexión

Esta clase se encarga de realizar los envíos al servidor, hereda de AsyncTask, la clase AsyncTask permite crear un hilo que se puede comunicar con la interfaz de usuario, así se puede realizar él envió en segundo plano.

Esta clase utiliza la librería org.apache.http.* para la comunicación HTTP con el servidor.

El método preExecute se lanza antes de ejecutar el hilo, el método doInBackground es el método que se realiza en segundo plano, por último tenemos el método onPostExecute que se lanza al terminar doInBackground.

5.4 Conclusiones

A través de los diferentes diagramas hemos podido ver el diseño del sistema, que estructura tiene la base de datos a través del diagrama de Entidad-Relación, como se comportara el sistema a través del diagrama de Casos de uso, o cómo interactúan las clases entre sí con el diagrama de Clases.

6. Implementación

6.1 Introducción

En este apartado se describirá el proceso de implementación de la aplicación, haciendo hincapié y mostrando el código más relevante.

6.2 Desarrollo

Cliente

AndroidManifest.xml

Archivo de configuración donde podemos aplicar la configuración básica de nuestra aplicación.

En este archivo indicamos el nombre de la aplicación, el tema visual que tendrá por defecto, la actividad inicial, que permios se requerirán, las claves para acceder a la API de GoogleMaps y las actividades que se podrán lanzar.

```
15      <permission
16          android:name="org.example.lostpet6.permission.MAPS_RECEIVE"
17          android:protectionLevel="signature"/>
18
19      <uses-permission android:name="org.example.lostpet6.permission.MAPS_RECEIVE"/>
20      <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
21      <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
22      <uses-permission android:name="android.permission.CAMERA" />
23      <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
24      <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
25      <uses-permission android:name="android.permission.INTERNET" />
26      <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
27      <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
28
29      <application
30          android:allowBackup="true"
31          android:icon="@drawable/ic_launcher"
32          android:label="@string/app_name">
```

Ilustración 36: Archivo AdroidManifest.xml permisos

Los permisos requeridos son:

- MAPS_RECIVE: Permite acceder a la API de GoogleMaps.
- READ_GSERVICES: Permite acceder al sensor GPS.
- WRITE_EXTERNAL_STORAGE: Permite acceder a la SDCard del sistema.
- CAMERA: Permite acceder a la cámara del sistema.

- ACCES_FINE_LOCATION: Permite obtener la ubicación de una forma precisa, GPS.
- ACCES_COARSE_LOCATION: Permite acceder a la ubicación utilizando posicionamiento por redes móviles, esta tiene menor precisión.
- INTERNET: Permite acceder a internet a través de los datos móviles.
- ACCESS_NETWORK_STATE: Permite acceder a el estado de conexión a internet de datos móviles, para conocer si está conectado.
- ACCESS_WIFI_STATE: Permite acceder al estado de la conexión WIFI, WIFI disponible, conectado.

En este archivo también contiene la clave necesaria para tener acceso a la API de Google Maps.

```
android:label="@string/app_name">
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="AIzaSyAtma4n09tACOAFrUi[redacted]"/>
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
```

Ilustración 37: Archivo AdroidManifest.xml GoogleMaps api key

MainActivity.java

Clase inicial de la aplicación encargada de construir la interfaz de usuario inicial, validar los datos para él envió, iniciar el mapa, e iniciar la interfaz de preferencias.

Cabe destacar que esta clase hereda de FragmentsActivity, esta clase es un contendor de Fragments, es necesario tener importado el paquete de soporte v4 para poder utilizarla.

Los Fragments son unidades de interfaz de usuario que se pueden intercambiar de forma dinámica por otros fragments según tipo de dispositivo, un ejemplo de uso de fragments es la aplicación de Gmail, esta posee dos fragments la barra latera con los tipos de correo, y otra con los correos en sí, en un dispositivo ovil, solo si pulsas sobre la parte izquierda de la pantalla aparece el fragment de barra lateral, en un dispositivo con un pantalla más grande, aparecerán los dos fragments a la vez.

En esta aplicación los fragments son las interfaces de usuario de entrada de datos, y se intercambian al deslizar el dedo por la pantalla. Esto se hace gracias a la clase ViewPager, que es la que implementa esta funcionalidad. A esta clase se le pasan el total de páginas que tendrá y los respectivos fragments que mostrara en cada página.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentManager());
    mViewPager = (ViewPager) findViewById(R.id.pager);
    mViewPager.setAdapter(mSectionsPagerAdapter);
    //Evita que se destruyan los fragments
    mViewPager.setOffscreenPageLimit(4);

    //Fragmentos
    camara=new Camara();
    formulario=new Formulario();
    sonido=new Sonido();
```

Ilustración 38: Creación del ViewPager

Una dificultad de utilizar este contenedor de Fragments, es que al intercambiar cualquier fragment este se elimina, esto es un problema si por ejemplo rellenamos el formulario y luego deseamos hacer una foto, en este caso al intercambiar el fragment Formulario por el de Cámara los datos del formulario se pierden, para evitar que se destruya se utiliza la siguiente instrucción.

```
//Evita que se destruyan los fragments
mViewPager.setOffscreenPageLimit(4);
```

Ilustración 39: Código MainActivity.java

Otro problema que encontramos al utilizar Fragments y ViewPager es que cuando en un fragment lanzamos algún ContentProvider en algunos casos el resultado de la actividad no se devuelve al fragment que la ha lanzado, sino que se devuelve a la vista que lo contiene, en este caso MainActivity, así que tenemos que recoger los datos devueltos y proporcionárselos a los fragments.

```
else if(resultCode!=Activity.RESULT_CANCELED){
    switch(requestCode){
        case GLOBALES.SELECCIONAR_IMAGEN:{
            camara.onMyActivityResult(requestCode, resultCode, data);
            break;
        }
        case GLOBALES.RECONOZIMINETO_DE_VOZ_REQUEST_CODE:{
            sonido.onMyActivityResult(requestCode, resultCode, data);
            break;
        }
    }
}
```

Ilustración 40: Código captura onActivityResult.

Camara.java

Clase, que extiende de Fragment, encargada de gestionar la captura de imágenes de la cámara del dispositivo o la obtención de las imágenes disponibles en la galería, así como de mostrar las imágenes ya tomadas.

Cabe destacar que esta clase hace uso de Content Providers. Los ContentProviders son herramientas de las aplicaciones Android que permiten la intercomunicación entre diferentes aplicaciones, con esta librería podemos llamar a la aplicación de cámara o de galería con una sola instrucción.

```
Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(f));

Intent intent=new Intent(Intent.ACTION_PICK,MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
```

Ilustración 41: Código lanzamiento providers.

Para mostrar las imágenes se hace uso de un contenedor GridView, este muestra las imágenes en forma de tabla. Un problema al utilizar este contenedor es que al tomar las imágenes con alta resolución este contenedor las muestra escaladas y necesita cargarlas en memoria, en este caso al tener un número considerable de imágenes el alto consumo de recursos ralentiza la aplicación y hasta puede causar un error, para solventar esto se ha decidido que al obtener una imagen se creará automáticamente una miniatura de esta, que será la que se cargue en este contendor, así se evita el alto consumo de memoria.

```
public void guardarFoto(){
    //Uri es variable de clase porque por data no lo coje
    if(uri!=null){
        Ficheros.crearMiniatura(uri);
        imagenes.add(uri);
        posicion++;
        adapter.notifyDataSetChanged();
    }
}

public void guardarGaleria(Uri path){
    try {
        String origen=Ficheros.obtenerRutaDesdeUri(path,actividad);
        if(origen!=null){
            File f = Ficheros.crearFicheroImagenTempExterno(this.id);
            Ficheros.copiarFichero(origen, f.getAbsolutePath());
            Ficheros.crearMiniatura(f.getAbsolutePath());
            imagenes.add(f.getAbsolutePath());
            posicion++;
            adapter.notifyDataSetChanged();
        }
        else{
            /*DialogoAlerta dc=new DialogoAlerta();
```

Ilustración 42: Código MainActivity.java crea miniaturas.

Formulario.java

Clase que extiende de fragments, que se encarga de crear el formulario para cumplimentar los datos del animal mediante componentes como Spinner (caja desplegable), RadioButton (elección obligatoria entre varias opciones) o EditText (campo de texto).
La clase posee las funciones necesarias para comprobar los campos rellenados.

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,Bundle savedInstanceState) {
    View rootView = inflater.inflate(R.layout.formulario,container, false);
    spinnerTipo=(Spinner) rootView.findViewById(R.id.spinner_tipo);
    spinnerRaza=(Spinner) rootView.findViewById(R.id.spinner_raza);
    radioM=(RadioButton) rootView.findViewById(R.id.radioButton_macho);
    radioH=(RadioButton) rootView.findViewById(R.id.radioButton_hembra);
    radioP=(RadioButton) rootView.findViewById(R.id.radioButton_pequenyo);
    radioMe=(RadioButton) rootView.findViewById(R.id.radioButton_mediano);
    radioG=(RadioButton) rootView.findViewById(R.id.radioButton_grande);
    color=(EditText) rootView.findViewById(R.id.editText_color);
    caracterisiticas=(EditText) rootView.findViewById(R.id.editText_caracteristicas);
    return rootView;
}
```

Ilustración 43: Código creación del formulario

Sonido.java

Clase que extiende de Fragment, que permite mediante un ContentProvider lanzar la aplicación de reconocimiento de voz y presentarlo en un campo de texto.

```
public void reconocimientoVoz(){
    Intent intent=new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    //Llamamos desde la actividad a causa del bug
    actividad.startActivityForResult(intent, GLOBALES.RECONOZIMINETO_DE_VOZ_REQUEST_CODE);
}
```

Ilustración 44: Código de lanzamiento del reconocimiento de voz

DialogFragment.java

Clase que hereda de DialogFragment, que se encarga de crear una ventana modal para mostrar cualquier mensaje o advertencia.

```
public Dialog onCreateDialog(Bundle savedInstanceState){
    if(getArguments().containsKey("message"))
        this.message=getArguments().getString("message");
    if(getArguments().containsKey("title"))
        this.title=getArguments().getString("title");
    if(getArguments().containsKey("button"))
        this.button=getArguments().getString("button");
    return new AlertDialog.Builder(getActivity())
    .setMessage(this.message)
    .setTitle(this.title)
    .setPositiveButton(this.button, new DialogInterface.OnClickListener()  {
        public void onClick(DialogInterface dialog, int id) {
            dialog.dismiss();
        }
    })
    .create();
}
```

Ilustración 45: Creación dela ventana de dialogo.

BBDD.java

Clase que se encarga de crear y gestionar la base de datos SQLite del dispositivo.
Cuando se accede a la BBDD esta clase comprueba que existe y si no es así crea las tablas necesarias, también tiene distintos métodos para realizar consultas y operaciones SQL.

```
private static final String CREATE_TABLE_DATOS="" +
        "CREATE TABLE Datos ( " +
            "id INTEGER PRIMARY KEY AUTOINCREMENT," +
            "tipo INTEGER DEFAULT 0," +
            "sexo INTEGER DEFAULT 0," +
            "tamanyo INTEGER DEFAULT 0," +
            "raza TEXT DEFAULT ''," +
            "color TEXT DEFAULT ''," +
            "carac TEXT DEFAULT ''," +
            "descripcion TEXT DEFAULT ''," +
            "fecha CHAR(19)," +
            "lat REAL DEFAULT 0," +
            "lng REAL DEFAULT 0 " +
        ")";

private static final String CREATE_TABLE_IMAGENES="" +
        "CREATE TABLE Imagenes( " +
            "id INTEGER PRIMARY KEY AUTOINCREMENT," +
            "ruta TEXT," +
            "id_dato INTEGER," +
            "FOREIGN KEY (id_dato) REFERENCES Datos(id) ON DELETE CASCADE" +
        ")";
```

Ilustración 46: Código creación tablas en BBDD.java

Ficheros.java

Clase estática encargada de gestionar los ficheros de la aplicación imágenes, esta clase crea las miniaturas de las imágenes obtenidas, y gestiona tanto las originales como las miniaturas.

```
public static void crearMiniatura(String f){
    Bitmap b=Utilidades.getBitmap(f, 100);
    FileOutputStream out;
    try {
        out = new FileOutputStream(obtenerRutaMiniatura(f));
        b.compress(Bitmap.CompressFormat.JPEG, 75, out);
    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

public static String obtenerRutaMiniatura(String ruta){
    int pos=ruta.lastIndexOf("/");
    String path=ruta.substring(0, pos);
    String name=ruta.substring(pos+1, ruta.length());
    return path+"/min_"+name;
}
```

Ilustración 47: Código gestión miniaturas

Mapa.java

Clase encargada de conectar con la API de GoogleMaps para mostrar un mapa, también se encarga de obtener la ubicación.
Esta clase se apoya con un contenedor llamado GoogleMap proporcionado por la librería google-play-services_lib, para mostrar y gestionar el mapa.

```
mapa = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap();
mapa.setMapType(GoogleMap.MAP_TYPE_NORMAL);
mapa.setMyLocationEnabled(true);
LocationManager lmanager=(LocationManager)getSystemService(LOCATION_SERVICE);
Criteria criteria=new Criteria();
String provider=lmanager.getBestProvider(criteria, true);
location=lmanager.getLastKnownLocation(provider);
```

Ilustración 48: Creación del mapa

Para dar la posibilidad al usuario de indicar la posición manualmente, la aplicación obtiene la posición automáticamente mediante la interfaz LocationListener y muestra el mapa centrado en esa ubicación, para marcar la ubicación verdadera se deja al usuario poner una marca en el mapa, esta utilidad se implementa mediante la interfaz OnMapLongClickListener.

```
@Override
public void onMapLongClick(LatLng latLng) {
    if(marca!=null){
        marca.remove();
    }
    marca=mapa.addMarker(new MarkerOptions().position(latLng));
    this.lat=latLng.latitude;
    this.lng=latLng.longitude;
}


@Override
public void onLocationChanged(Location location) {
    double latitude=location.getLatitude();
    double longitude=location.getLongitude();
    LatLng latLng=new LatLng(latitude,longitude);
    if(marca==null){
        mapa.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mapa.animateCamera(CameraUpdateFactory.zoomTo(14));
    }
}
```

Ilustración 49: Código obtener ubicación y poner marca.

Envio.java

Clase que se encarga de crear y gestionar la interfaz de usuario que muestra los envíos pendientes, tiene un contenedor ListView que gestiona el listado de envíos. Al igual que con el contenedor GridView, si las imágenes que se muestran en el listado fueran escaladas desde el original podrían ocasionar un consumo excesivo de memoria, así que se aprovecha las miniaturas ya creadas anteriormente para mostrar las imágenes.

```
if(d.getImagenes().size()>0){
    img.setScaleType(ImageView.ScaleType.CENTER_CROP);
    img.setImageURI(Uri.fromFile(new File(Ficheros.obtenerRutaMiniatura(d.getImagenes().get(0).getRuta())))
}
else{
    img.setImageResource(android.R.drawable.ic_menu_gallery);
}
```

Ilustración 50: Código obtención archivos de miniaturas

Esta clase también se responsabiliza de enviar los datos, al enviar los datos se envía una cadena en notación JSON con los datos y las imágenes, ya que puede haber algún error en el proceso de envío, esta clase se encarga de enviarlo por partes y esperar la confirmación del servidor para conocer que la recepción ha sido satisfactoria.

```
try {
    //No esta guardado en el servidor
    if(dato.getIdServidor()==0){
        String json = dato.toJSON();
        List<NameValuePair> pares = new ArrayList<NameValuePair>();
        pares.add(new BasicNameValuePair("operacion","guardar_dato"));
        pares.add(new BasicNameValuePair("usuario",usuario));
        pares.add(new BasicNameValuePair("dato",json));
        Conexion con=new Conexion(this,pares);

        String res=con.execute(url,"post").get();
        if(Utilidades.esNumero(res)){
            dato.setIdServidor(Long.parseLong(res));
            dato.modificarIdServidor(getApplicationContext());
            if(subirImagenes(url,dato.getIdServidor(),dato.getImagenes())){
                borrarDatos(position);
                mostrarAlertas(4);
            }
        }
    }
    //Si esta guardado en el servidor
    else{
        if(subirImagenes(url,dato.getIdServidor(),dato.getImagenes())){
            borrarDatos(position);
            mostrarAlertas(4);
        }
    }
} catch (InterruptedException e) {
```

Ilustración 51: Código envío datos e imágenes

```
private boolean subirImagenes(String url, long id, List<Imagen> imagenes)
    for(Imagen img : imagenes){
        if(img.getEstado()==0){
            Conexion con=new Conexion(this,null);
            url+="?operacion=guardar_img&id="+id;
            String res=con.execute(url,"file",img.getRuta()).get();
            if(res.equals("ok")){
                img.setEstado(1);
                img.modificarEstado(getApplicationContext());
                Thread.sleep(1000);
            }
            else{
                throw new InterruptedException();
            }
        }
    }
    return true;
}

private void borrarDatos(int position){
    Datos dato=datos.get(position);
    for(Imagen img : dato.getImagenes()){
        Ficheros.eliminarImagen(img.getRuta());
    }
    dato.eliminarDato(getApplicationContext());
    datos.remove(position);
    adapter.notifyDataSetChanged();
}
```

Ilustración 52: Código subir imágenes y borrar imágenes subidas.

Conexion.java

Esta clase se utiliza para gestionar el envío al servidor, la conexión con el servidor se hace mediante HTTP. Esta clase extiende de AsyncTask, que permite crear un hilo que se ejecuta en paralelo con la aplicación, que será el que se encargue de enviar los datos e imágenes.

```
if(opcion.equals("post")){
    try{
        //Poner el timeout
        HttpParams param=new BasicHttpParams();
        HttpConnectionParams.setConnectionTimeout(param,20000);

        //Conectar
        HttpClient client = new DefaultHttpClient(param);
        HttpPost post = new HttpPost(url);

        //Las variables post
        try {
            post.setEntity(new UrlEncodedFormEntity(pares));
        } catch (UnsupportedEncodingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        HttpResponse response = client.execute(post);
        HttpEntity entity = response.getEntity();
        InputStream is = entity.getContent();

        //Leer del buffer
        BufferedReader reader = new BufferedReader(new InputStreamReader(is,"UTF8"),8);
        StringBuilder sb = new StringBuilder();
        String line = null;
        while((line = reader.readLine()) != null) {
            sb.append(line);
```

Ilustración 53: Envió de datos al servidor

main.xml

Este archivo es un layout, un layout es un contenedor de vistas que controla su comportamiento y posición, tiene un formato xml en el que se indica que componentes se visualizan y cuál será su forma, en este caso este layout especifica la vista ViewPage especificando que ocupe toda la pantalla tanto de ancho como de alto entre otras propiedades.

```
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
```

Ilustración 54: Layout ViewPager

camara.xml

En esta ocasión se especifica las propiedades del GridView que contendrá la vista cámara, ocupa toda a pantalla disponible, sus columnas serán se 110 dp (pixeles por densidad), las celdas se rellenarán ocupando todo el espacio, se dejara un margen de 5dp entre celdas, las columnas se adaptarán al espacio disponible y por último, se centrara el contenido de las celdas.

```
<GridView
    android:id="@+id/galeria"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:columnWidth="110dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="5dp"
    android:horizontalSpacing="5dp"
    android:stretchMode="columnWidth"
    android:gravity="center" />
```

Ilustración 55: Layout Grid View

strings.xml

Este es un archivo de recursos en notación xml, en este se indican unos ítems y su correspondiente cadena de caracteres, la cadena será el texto que deseemos que aparezca en la interfaz.
Según la carpeta que lo contenga, el sistema Android elegirá uno u otro dependiendo del idioma que tengamos predefinido en el dispositivo. El archivo string.xml que este en la carpeta res/values/string.xml será el utilizado por defecto, si el archivo está en la carpeta res/values-es/strings.xml mostrará estas cadenas en la interfaz si el idioma del dispositivo es el español.

```
<string name="app_name">LostPet</string>
<!-- Botones Menu -->
<string name="send">Enviar</string>
<string name="settings">Configurar</string>
<string name="siguiente">Siguiente</string>
<string name="pendiente">Envios Pendientes</string>
```

Ilustración 56: Archivo de recursos

Servidor

repartidor.php

Este script utilizado en el servidor, e implementado con el lenguaje PHP, se utiliza para según el tipo de llamada POST o GET que reciba, lanzar una acción dependiendo del valor obtenido en la cadena operación.

```
if(!isset($_POST["operacion"]) && !isset($_GET["operacion"]))
    die("ERROR");
if(isset($_POST["operacion"]))
    $operacion=$_POST["operacion"];
else{
    $operacion=$_GET["operacion"];
}
if($operacion=="guardar_dato"){
    echo Logica::Guardar();
}
elseif($operacion=="guardar_img"){
    echo Logica::GuardarImagen();
```

Ilustración 57: Script repartidor

logica.php

Este script tiene diversos métodos para hacer diversas acciones como, guardar los datos que son enviados en formato JSON, o guardar las imágenes.

```
public static function Guardar(){
    $usuario=$_POST["usuario"];
    if($usuario==""){
        $usuario="Anonimo";
    }
    $json=$_POST["dato"];
    $datos=json_decode($json,true);
    $res=BD::insertarDatos($datos["tipo"],$datos["sexo"],$datos["tamanyo"],$dat
    return $res;
}

public static function GuardarImagen(){
    $id_dato=$_GET["id"];
    $ruta=self::GuardarArchivoImagen($id_dato);
    if(!empty($ruta)){
        $res=BD::InsertarImagen($id_dato,$ruta);
        if(is_numeric($res)){
            return "ok";
        }
        return $res;
    }
    return "error";
}
```

Ilustración 58: Script lógica

bbdd.php

Este script contiene métodos para hacer consultas y operaciones sobre la base de datos MySQL.

```
public static function ObtenerDatos(){
    $consulta="select * from lostpet_datos";
    $bd=new BBDD("android");
    return $bd->Consulta($consulta);
}

public static function ObtenerImagenes($id_dato){
    $consulta="select * from lostpet_imagenes where id_dato=$id_dato";
    $bd=new BBDD("android");
    return $bd->Consulta($consulta);
}
```

Ilustración 59: Script acceso a base de datos

6.3 Conclusiones

En esta sección se han descrito las partes que más importantes y que más dificultades han presentado en la implementación de la aplicación, explicando las decisiones tomadas y las ventajas de llevarlas a cabo, mostrando el código más relevante.

En el siguiente capítulo se presenta el trabajo realizado, que aporta este software y mejoras y ampliaciones se pueden desarrollar en el futuro.

7. Conclusiones

7.1 Resumen del trabajo desarrollado

En este documento hemos podido ver la problemática social de los animales perdidos y abandonados, así como apreciar la gran cantidad de información recolectada pero imposible de usar eficientemente para su propósito.

Para solventar este problema se ha realizado una aplicación en Android que pretende centralizar la información en un servidor común, desde distintos métodos de entrada.

Se ha visto que herramientas y tecnologías están disponibles para capturar la mayor cantidad de datos en tiempo real y de la forma más automatizada posible.

En cuanto a los objetivos de disponer de una interfaz que pueda aunar varios métodos de entrada de datos, de una forma fácil y clara, y centralizarlos en un servidor, se ha creado un software capaz de realizar este propósito.

Se han descrito las partes más relevantes del código, así como las decisiones tomadas ante inconvenientes que han surgido durante el desarrollo de la misma.
Así como un manual de uso de la aplicación, para sacar el máximo rendimiento de esta.

Por último el objetivo principal que es ser una herramienta de ayuda para la búsqueda y localización de animales perdidos y abandonados, no se puede probar que es totalmente útil, ya que solo se ha probado en un ámbito universitario.

7.2 Aportaciones

Las aportaciones realizadas en este documento son:

Una visión de la problemática actual de las mascotas perdidas y abandonadas, y un estudio de las herramientas disponibles actualmente, y la eficacia de estas.

Un estudio de los dispositivos y tecnologías disponibles para recolectar información en tiempo real.

Un software capaz de aunar diversos métodos de captura de datos y centralizarlos en un servidor.

7.3 Trabajo futuro

A continuación se exponene diversas mejoras y ampliaciónes para hacer esta herramienta más util en su proposito final.

En primer lugar integrar de forma funcional algunos metodos de entrada de datos como la realidad virtual, o mejorar el acceso a ciertas capturas de datos mediante Widgtets, que por ejemplo proporcionen acceso directo a la toma de una imagén.

Támbien seria interesante extender a otras plataformas este software, como iOS o Windows Phone, así se incrementaria la información recopilada.

Seria conveniente crear un cliente web para el sistema centralizado, para ofrecer los datos recopilados. Así como desarrollar algún software capaz de reconocer imágenes de animales y en consecuencia poder obtener una trazavilidad de las mascotas perdidas para poder recuperarlas.

8. Referencias

Internet:

Estudio Fundación Affinity:
http://www.federacionandaluzadegalgos.com/pdf/FUNDACION%20AFFINITY%20ESTUDIO%202010.pdf

Definición AJAX:
http://www.alegsa.com.ar/Dic/ajax.php

Definición API:
http://www.alegsa.com.ar/Dic/api.php

Definición Aplicación:
http://www.mastermagazine.info/termino/3874.php

Definición Aplicación Móvil:
http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_m%C3%B3vil

Definición BBDD:
http://www.masadelante.com/faqs/base-de-datos

Definición Cliente:
http://es.wikipedia.org/wiki/Cliente_(inform%C3%A1tica)

Definición HTTP:
http://www.definicionabc.com/tecnologia/http.php

Definición IDE:
http://www.mastermagazine.info/termino/5320.php

Definición Metadatos:
http://es.wikipedia.org/wiki/Metadato

Definición SDK:
http://www.alegsa.com.ar/Dic/sdk.php

Definición Servidor:
http://es.wikipedia.org/wiki/Servidor

Definición SQL:
http://definicion.de/sql/

TCP/IP:
http://www.masadelante.com/faqs/tcp-ip

Definición XML:
http://definicion.de/xml/

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

Ejemplos código:
http://stackoverflow.com/

UML:
http://www.uml.org/#UML2.0

Modelado de casos de usos:
http://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso

Modelado de clases:
http://users.dcc.uchile.cl/~psalinas/uml/modelo.html

Bibliografía:

El gran libro de Android / Jesús Tomas Gironés / Editorial Mocambo S.A. 2011.

Una guía para la realización y supervisión de proyectos final de carrera (PFC) en el ámbito de la web / Félix Buendía García / Editorial Universidad Politécnica de Valencia.

Anexo

Introducción

En esta sección se va a mostrar cómo funciona el sistema a nivel de usuario, explicando con imágenes y texto todas las funcionalidades que un usuario puede utilizar en la aplicación, para sacar el máximo partido al software.

Software

La aplicación LostPet es un sistema para la captura de datos a través de diferentes entradas disponibles en cualquier dispositivo móvil, este permite la captura de imágenes, la introducción de los datos más relevante de un animal a través de un formulario, la descripción vía reconocimiento del lenguaje y la descripción de la mascota apoyándonos en la realidad virtual, toda esta información más algunos metadatos capturados automáticamente o de forma manual serán enviados a un servidor. Esta aplicación pretende a través de un cliente móvil tomar la mayor cantidad de datos y centralizarlos en un servidor de forma clara y ordenada, y así evitar la pérdida de información o que se da en otros sistemas.

En este manual servirá para conocer todas las funcionalidades disponibles y como utilizarlas, para así obtener el mayor rendimiento de la aplicación.

Instalación

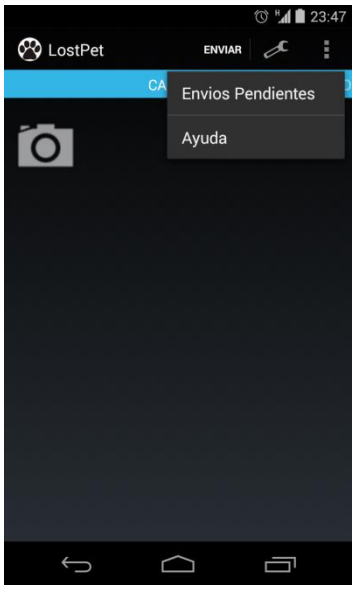
En lo que se refiere a la instalación, en el servidor no será necesario ya que este estará disponible desde el primer momento a los clientes.
En cuanto a la aplicación móvil será necesario obtener el paquete del programa apk e instalarlo en el móvil, esta se hace de forma automática.

Configuración

La configuración respecto al servidor, será necesario poner el servidor a la escucha de las peticiones de los clientes, en cuanto a los clientes, será necesario indicar la IP del servidor y el puerto a conectar, así como diferentes parámetros como el usuario y contraseña, si se posee una, o con que conexión móvil estará permitido enviar los datos, cualquiera o solo WIFI.

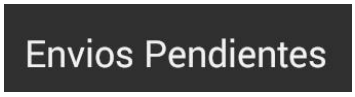
Manual de Usuario

Al iniciar la aplicación móvil, podremos ver una vista donde tendremos la opción de empezar a introducir datos, iniciar la configuración de la aplicación, y pasar directamente a envíos pendientes de realizar.



Botón de menú recoger los datos necesarios y enviar.

Botón de menú para acceder a preferencias.



Opción del menú ara ir directamente a la vista de envíos pendientes.

Ilustración 60: Interfaz de usuario imágenes tomadas inicio

En esta interfaz de usuario deslizante estará dividida en cuatro posibles vistas.

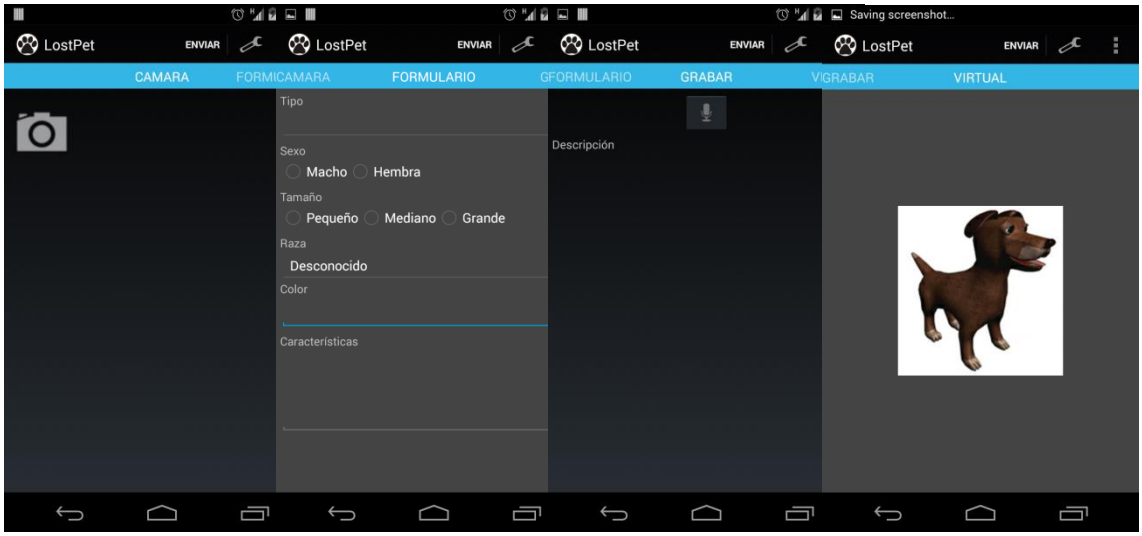
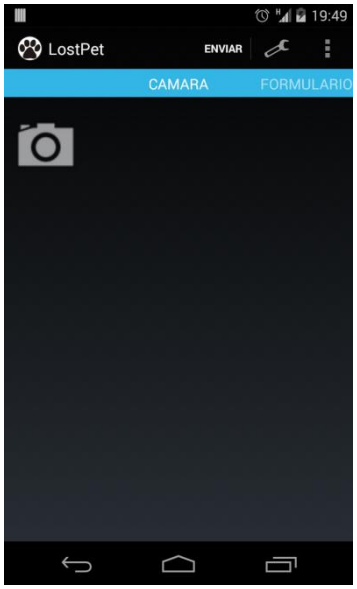


Ilustración 61: Cámara Formulario Captura de voz Realidad Virtual

Cámara

En esta vista podremos apreciar un icono con forma de cámara que se utilizara para lanzar la aplicación de cámara disponible en el dispositivo y poder capturar imágenes con esta.



Botón para lanzar la cámara

Manteniendo este botón pulsado se cambia al botón de galería.

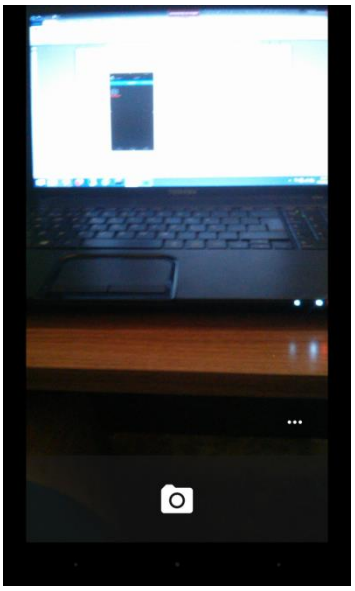
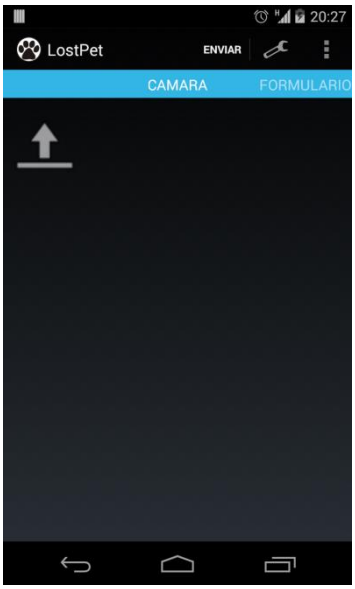


Ilustración 62: Interfaz de usuario icono cámara

Interfaz de usuario de la cámara

Si pulsamos el icono de la cámara durante un breve periodo de tiempo, esta cambia para proporcionar un acceso a la galería de imágenes disponible en él dispositivo.



Botón para lanzar la galería.

Manteniendo este botón pulsado se intercambia por el de cámara.

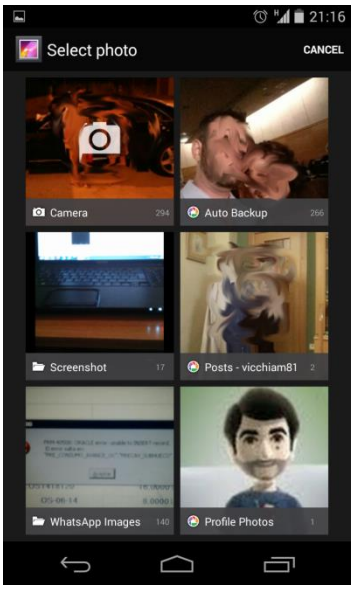
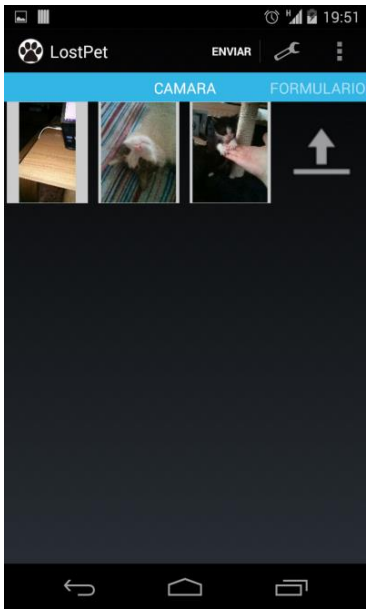


Ilustración 63: Interfaz de usuario icono galería

Interfaz de usuario de la galería

En cualquier caso la imagen capturada u obtenida a través de la galería pasará, en forma de miniatura, a la interfaz para visualizar el total de imágenes obtenidas en formato tabla.

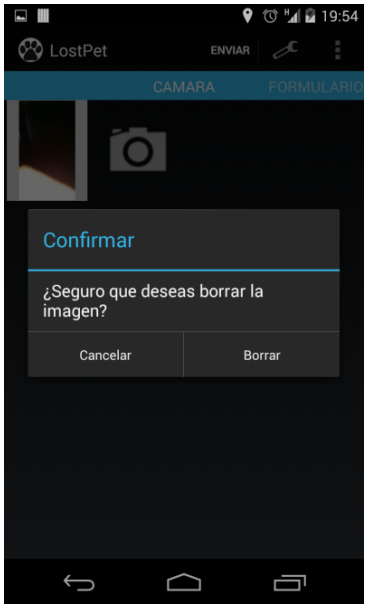


Se podrán añadir hasta un total de 12 imágenes.

Si se produce un error al intentar obtener la imagen, el sistema mostrará una advertencia indicándolo.

Ilustración 64: Interfaz de usuario imágenes tomadas

Las imágenes de la interfaz de usuario cámara se pueden tanto eliminar con una pulsación larga en la imagen, como visualizarlas en pantalla completa mediante una pulsación corta. Si se indica que una imagen se quiere eliminar el sistema pedirá confirmación para realizar la acción.

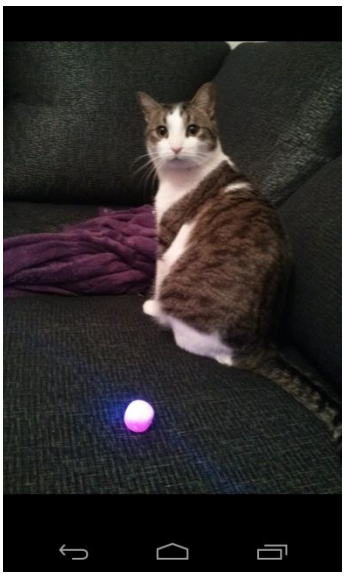


Las imágenes obtenidas de la galería no se eliminaran ya que la aplicación hace una copia.

Ilustración 65: Interfaz de usuario imágenes tomadas opción eliminar.

Visualizar Imágenes

Se podrá visualizar las imágenes obtenidas mediante desplazamiento horizontal pulsando sobre una imagen, todas las imágenes se mostraran ocupando la mayor parte posible de la pantalla del dispositivo.



Cuando se desee finalizar esta visualización solo bastará con pulsar el botón de volver para seguir con la captura de datos en la interfaz cámara.

Ilustración 66: Interfaz de usuario visor imágenes con alerta

Formulario

Vista donde podremos apreciar una serie de campos a rellenar por parte del cliente, estos campos son las propiedades más importantes del animal, es gato o perro, edad aproximada, raza, color, tamaño y una descripción más detallada de este. Algunos campos solo dejarán elegir entre una serie de valores, mientras que otros son de texto libre.

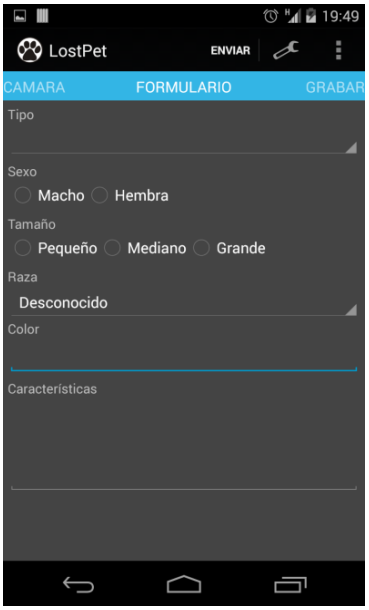
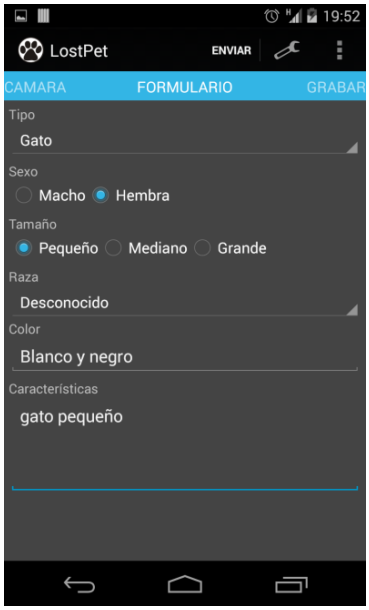


Ilustración 67: Formulario sin rellenar



Formulario rellenado

Si algún campo obligatorio del formulario no se cumplimenta se mostrará un mensaje recordando que el campo debe de rellenarse.

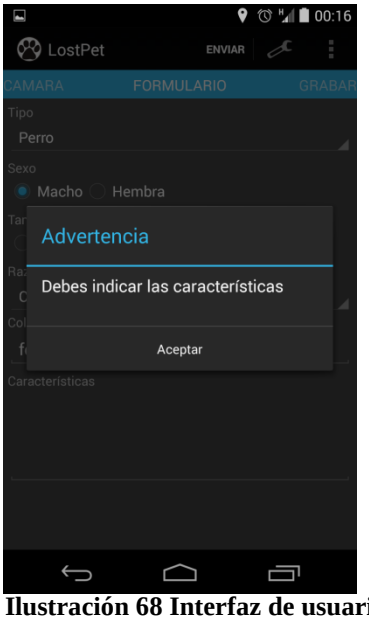
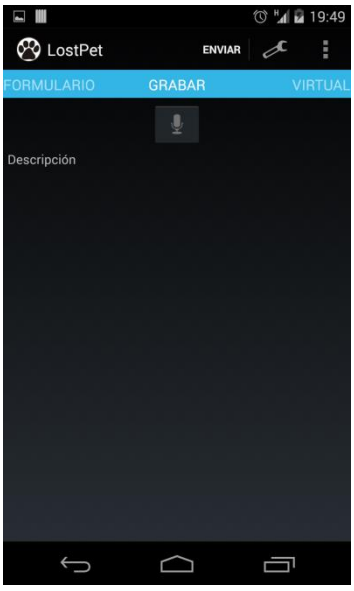


Ilustración 68 Interfaz de usuario Formulario con alerta

Sonido

Esta vista permite mediante reconocimiento de voz y una conexión a internet capturar la descripción que el usuario realizará mediante su voz y trasladarla a texto en la interfaz de usuario.



Para tener disponible la opción de reconocimiento de voz es necesaria una conexión a internet.

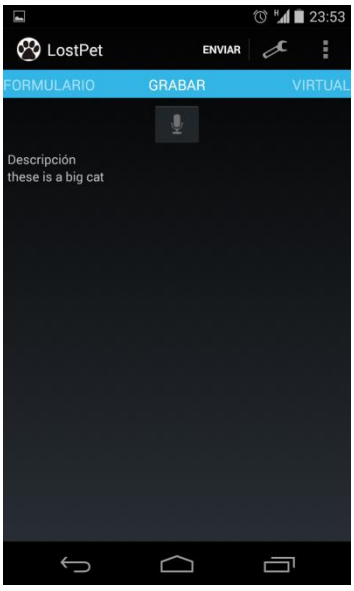


Ilustración 69: Interfaz de usuario sonido vacía **Interfaz sonido rellena**

Realidad virtual

Apoyándose con la realidad virtual el usuario podrá hacer la descripción visual más aproximada de la mascota

Ilustración 70: Interfaz de usuario realidad virtual vacía

Preferencias

Si se elige la opción de editar las preferencias, se visualizará una vista donde podremos elegir los parámetros relevantes de la aplicación, servidor y puerto donde escuchará el servidor, así como elegir con qué tipo de conexión a la red se enviarán imágenes, cualquier o WIFI.

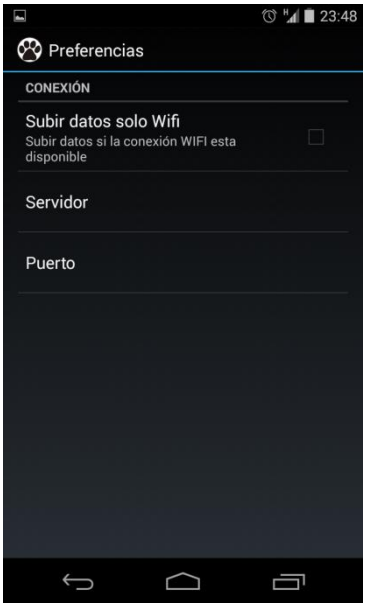
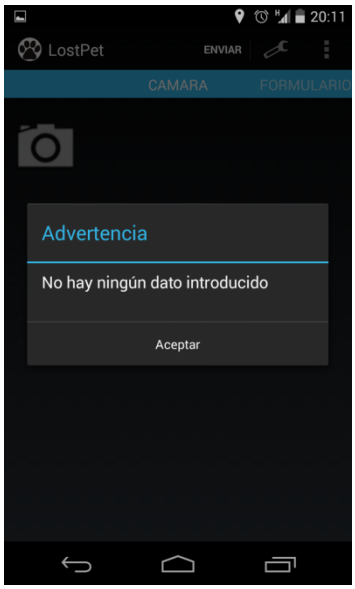


Ilustración 71: Interfaz de usuario preferencias.

Todas estas vistas son complementarias entre sí, ya que se pueden capturar los datos de cualquiera de las cuatro formas pudiendo utilizar varias a la vez, con la única condición de que al menos el formulario esté debidamente cumplimentado.
Para avanzar en el proceso una vez se tengan los datos capturados se pulsará sobre el botón siguiente.

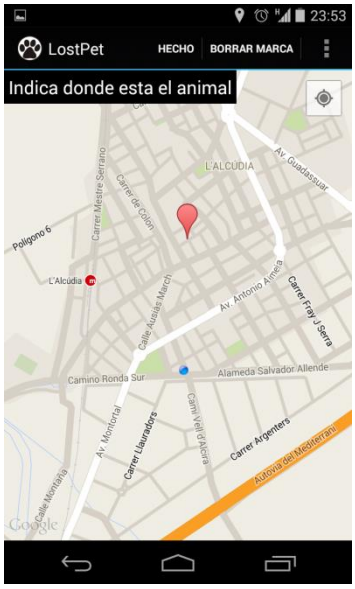


Si no se cumplen los requisitos mencionados anteriormente el sistema mostrará una advertencia.

Ilustración 72: Interfaz de usuario visor imágenes con alerta

Mapa

Esta vista mostrará un mapa de GoogleMaps donde automáticamente se intentará obtener la ubicación actual y más precisa disponible del usuario y se mostrará una marca en esta, el usuario podrá introducir la ubicación manualmente si así lo desea, con tan solo desplazar el mapa a la ubicación deseada y mantener pulsado sobre este para poner una marca indicando la ubicación real donde se encontró al animal.



Una vez obtenida la ubicación de forma automática o manual se procederá a obtener las coordenadas, y con estas y los datos introducidos anteriormente, se lanzara la siguiente interfaz.

Circulo azul, ubicación actual.
Marca roja, ubicación indicada.

Ilustración 73: Interfaz de usuario mapa

Envíos pendientes

En esta interfaz de usuario se podrá visualizar todos los envíos pendientes que aún no se han envido completamente al servidor, se mostraran en forma de listado, pudiendo elegir eliminarlas si así se desea.

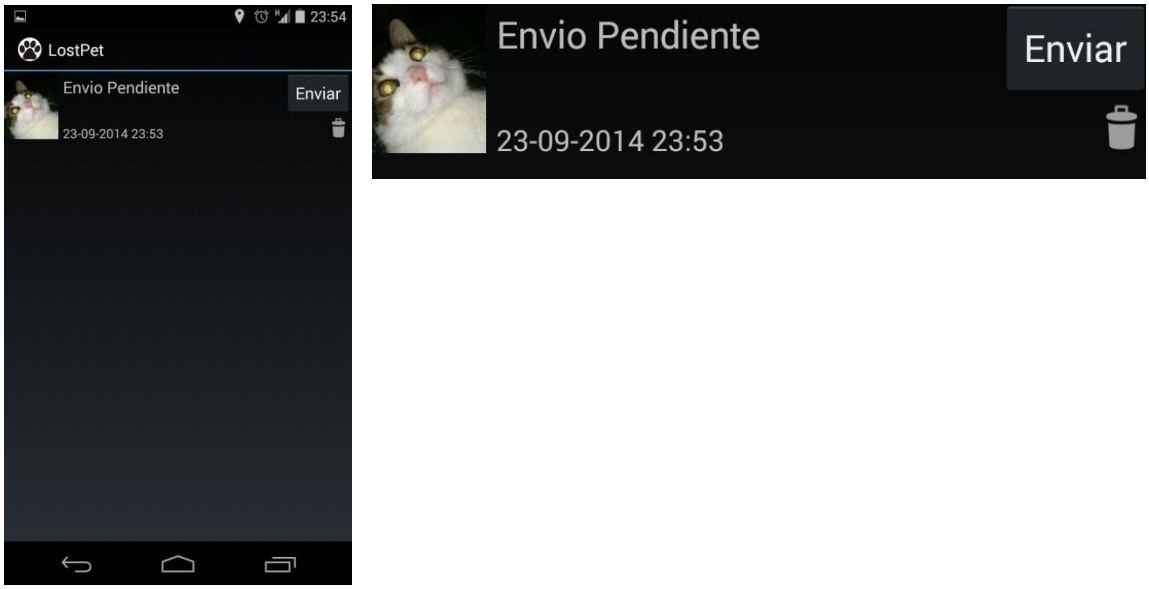


Ilustración 74: Interfaz de usuario envíos pendientes

Si el envío se hace correctamente y de forma completa se eliminará de la lista el envío, y se mostrara un mensaje de envío realizado correctamente, en caso contrario se indicar al usuario que el envío no se realizó correctamente.

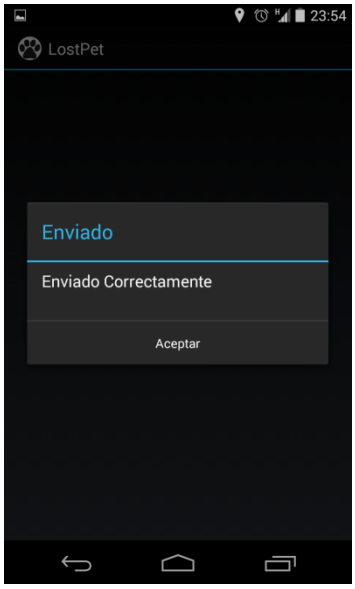


Ilustración 75: Interfaz de usuario envíos pendientes