Universitat Oberta de Catalunya

Trabajo Final de Grado (TFG)
Where's my Pet
Desarrollo de una aplicación con Java EE

Memoria

Autor: Alberto Muñoz Losada
Consultor: Albert Grau Perisé

Índice General

Índice de ilustraciones

Resumen

En la actualidad, y principalmente en los países denominados desarrollados, nos encontramos inmersos en la llamada sociedad de la información. Daniel Bell, sociólogo estadounidense, fue el primero en introducir este término de la siguiente forma: “Una sociedad post-industrial es básicamente una sociedad de la información. El intercambio de información en términos de varios tipos de procesamiento y almacenamiento de datos, investigación de mercado, etc… es la base de la mayoría de cambios económicos” (Bell, 1973).

En la era de Internet y de las comunicaciones, la necesidad de disponer de cierta información independientemente del lugar en el que nos encontremos ha hecho, que esta sociedad de la información haya ido evolucionando a lo que podríamos denominar, sociedad de la información móvil.

A su vez, la evolución de la tecnología, sobre todo desde finales del siglo XX ha permitido, que un usuario medio pueda tener a su alcance potentes equipos de computación sobre dispositivos de un tamaño infinitamente reducido, comparados con los ordenadores de primera generación de mediados del siglo pasado. Son, los llamados dispositivos móviles inteligentes (smartphones, tabletas, PDAs…).

Teniendo en cuenta lo anterior, este trabajo se centra en el análisis, diseño y desarrollo de una aplicación cliente para dispositivos móviles Android que permita, de forma remota y sencilla la identificación y localización de perros perdidos.

Asimismo, se desarrollará una aplicación servidor para atender las diferentes peticiones del cliente a través de servicios web, la cual accederá a una base de datos para añadir o recuperar información.

Además, y debido al gran auge del denominado “cloud computing”, estos servicios serán proporcionados desde la nube.

Licencia

1. INTRODUCCIÓN

En este capítulo se describirá brevemente el Trabajo Fin de Grado, los motivos por los que se llevará a cabo el mismo, los objetivos que se desean conseguir con este trabajo, el alcance del proyecto y se definirá una planificación para su correcta consecución.

1.1. Motivaciones

Desde la domesticación animal (según diferentes fuentes, alrededor del año 9000 a.C.), el ser humano ha convivido con infinitud de especies animales para diferentes propósitos. En ocasiones, alimenticios, en otras, únicamente como compañeros. Estos últimos, son los denominados animales de compañía. En la sociedad moderna, incluso tratamos a nuestros animales de compañía como a un ser querido. La pérdida de estos animales, conlleva en muchas ocasiones situaciones traumáticas para sus dueños.

A día de hoy, existen ciertos medios para ayudar a localización de estos animales, principalmente el llamado “microchip subcutáneo”. Este microchip es implantado de manera subcutánea y almacena diferente información (datos relativos al dueño, vacunas, historial veterinario…).

La legislación vigente ha hecho que la implantación de este dispositivo tenga carácter obligatorio, concretamente en gran parte del territorio nacional español, al tratarse de una competencia transferida a las diferentes Comunidades Autónomas, al igual que sucede en otros países occidentales. Sin embargo, la lectura de este tipo de chips, está restringida a un número limitado de puntos de lectura o clínicas veterinarias, lo que en ocasiones hace complejo que alguien que cree localizar a un animal de compañía extraviado pueda ponerse en contacto con sus dueños, dificultando la colaboración entre individuos.

El desarrollo de este trabajo pretende facilitar y a su vez agilizar la identificación y localización de estos animales de compañía, fomentando la colaboración entre individuos.

Otro factor que ha llevado a la realización de este trabajo es la gran expansión, aceptación y continua evolución que tienen en la actualidad los dispositivos móviles inteligentes (denominados smartphones) en el ámbito social. La potencia de cálculo de estos dispositivos, comparable en ocasiones a la de un ordenador de escritorio o portátil, el almacenamiento, así como las diferentes funciones que muchos de ellos combinan (conexión vía WIFI, cámara de fotos, navegador GPS…) los convierten en el dispositivo de movilidad por excelencia, permitiendo a cualquier usuario el acceso a todo tipo de información dondequiera que se encuentre.

1.2. Objetivos

Este trabajo surge por la necesidad de ofrecer a un gran número de usuarios una nueva forma ágil, intuitiva y sencilla para la identificación y localización de animales de compañía. Analizando el mercado existente de aplicaciones para dispositivos móviles, no se ha encontrado ninguna herramienta que proporcione características similares.

Para conseguir este objetivo principal, existen otros objetivos secundarios a conseguir:

- Análisis de requerimientos para los diferentes actores de la aplicación.

- Análisis, diseño e implementación de la información que se almacenará en la base de datos utilizada por el sistema.

- Análisis, diseño e implementación de una aplicación que será ejecutada en un equipo a modo de servidor del sistema, la cual será capaz de acceder a la información contenida en la base de datos, tratar dicha información según requerimientos y comunicarse de manera óptima y efectiva con los dispositivos móviles, recibiendo peticiones de estos y proporcionando la información requerida a los mismos.

- Conocer las diferentes arquitecturas para dispositivos móviles (especialmente, Android) así como adquirir los conocimientos necesarios para el desarrollo de aplicaciones para dichos dispositivos.

- Análisis, diseño e implementación de una aplicación Android que implemente los diferentes requerimientos previamente definidos, comunique con el servidor y presente la información al usuario.

- Realizar pruebas exhaustivas tras la finalización de la implementación de la aplicación, para asegurar la fiabilidad de la misma, la consecución de los objetivos y el funcionamiento correcto de los procesos definidos.

- Documentar el desarrollo de la aplicación y el proceso de realización del mismo con el objetivo de facilitar la comprensión de la herramienta desarrollada y su posible aplicación a diferentes ámbitos.

1.3. Alcance del proyecto

El desarrollo de la aplicación móvil contemplará inicialmente los siguientes requerimientos:

Existirán tres tipos de usuarios en el sistema: usuarios que denominaremos usuario básico, usuario registrado y usuario administrador.

El sistema ofrecerá al usuario un menú de navegación rápida para poder acceder de manera sencilla a todas las funcionalidades proporcionadas (en función del perfil del usuario). Se analizará la posibilidad de limitar el uso de la aplicación a usuarios registrados.

El sistema dará en todo momento la posibilidad de identificarse en la aplicación. Una vez identificado y validado el usuario, el sistema ofrecerá al usuario un menú de navegación rápida con las funcionalidades propias del perfil de usuario registrado, así como desconectarse de la aplicación.

El sistema proporcionará la funcionalidad de realizar una fotografía a un animal de compañía.

El sistema proporcionará la funcionalidad de localización geográfica en un momento determinado.

Un usuario no registrado tendrá la posibilidad de realizar la identificación de un animal y enviarla para su registro en el sistema, permitiendo la posibilidad de almacenar información adicional como la localización geográfica en la que se ha llevado a cabo la misma así como, información de contacto necesaria para la comunicación con el dueño del animal en caso de una futura identificación por parte de este último.

Un usuario registrado dispondrá de las mismas funcionalidades que uno no registrado, además, tendrá la posibilidad de registrar sus datos, registrar diferentes animales dentro de su ficha, gestionar dichos registros, marcar cualquiera de estos registros como “extraviado” (para optimizar el sistema, tan solo se realizarán búsquedas sobre animales marcados como extraviados), consultar los diferentes avisos para las posibles coincidencias de animales marcados como extraviados y confirmar la validez de dichos registros.

Un usuario administrador dispondrá además de funcionalidades administrativas para la propia herramienta en caso necesario.

El proceso tipo de uso de la herramienta sería el siguiente:

- En primer lugar, el usuario deberá descargar de la tienda oficial para dispositivos Android “Google Play” la aplicación e instalarla en el dispositivo móvil. Tan solo será posible disponer de una aplicación por dispositivo móvil.

- Un usuario se registra en la aplicación y da de alta sus datos y los de sus diferentes animales de compañía.

- En un momento determinado un usuario registrado podría marcar alguno de estos animales como extraviado permitiendo que este registro sea incluido en las diferentes búsquedas de la aplicación.

- Un usuario cualquiera realizaría una identificación de un animal localizado mediante una fotografía y la registraría en el sistema junto a diferentes datos adicionales como puede ser la forma de contacto en caso de identificación positiva por parte del posible dueño (de manera opcional se registraría información como, la localización, observaciones… los cuales facilitarían en algunos casos la verificación del animal localizado por parte de su posible dueño).

- Un usuario registrado, suponiendo que se produzca coincidencia entre la identificación registrada anteriormente por cualquier usuario, con un animal marcado como extraviado, dispondría de cierto tipo de aviso que le facilitase el acceso para verificar y validar la localización del mismo.

- El usuario registrado, accedería a las coincidencias con datos básicos (fotografía, observaciones, localización…) y en caso de tratarse del animal buscado (debido a la complejidad de la identificación de facciones animales, podrían darse varios posibles valores coincidentes), validaría o desecharía la localización (sería posible también cancelar posteriormente dicha localización en caso de falso positivo). Por motivos de confidencialidad de la información, solo entonces, se le mostraría todos los datos, como pueden ser los relativos al contacto con el usuario que ha realizado la localización.

- Una vez validada una localización de un animal, el sistema debería ser capaz de eliminar cualquier registro referente al mismo con una fecha anterior a la localización. Hay que tener en cuenta, que un mismo animal, puede extraviarse en fechas consecutivas.

1.4. Planificación

Para la planificación con hitos y temporalización del proyecto, se utilizará Microsoft Project, un software para la gestión de proyectos (planificación, asignación de recursos, temporalización, seguimiento…) el cual permite la realización de diferentes diagramas de análisis y seguimiento.

Resumen de las fechas clave de las fases del Trabajo Fin de Grado:

Fecha clave	Documento	Descripción
02/10/2014	PAC1	Elaboración del plan de trabajo, detalle de requisitos y funcionalidades y planificación del proyecto.
07/11/2014	PAC2	Estudio de la arquitectura, análisis general del proyecto, funcionalidades (diagrama de clases, diagrama de interrelación…), diseño de la interface.
16/12/2014	PAC3	Implementación y codificación del proyecto (versión Beta).
13/01/2014	PAC4	Fecha de finalización del proyecto con la entrega de la memoria, presentación final, vídeo y código depurado (producto final).

El diagrama de Gantt de la planificación temporal de las diferentes tareas que componen el Trabajo Fin de Grado es el siguiente:

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**Trabajo Fin de Grado**	**127 días**	**sáb 20/09/14**	**sáb 24/01/15**	
2		**Planificación**	**13 días**	**sáb 20/09/14**	**jue 02/10/14**	
3		Propuesta de TFG	1 día	sáb 20/09/14	sáb 20/09/14	
4		Aprobación de TFG	1 día	dom 21/09/14	dom 21/09/14	3
5		Introducción	1 día	lun 22/09/14	lun 22/09/14	4
6		Objetivos del TFG	1 día	mar 23/09/14	mar 23/09/14	5
7		Funcionalidades	3 días	mié 24/09/14	vie 26/09/14	6
8		Planificación temporal del TFG	2 días	sáb 27/09/14	dom 28/09/14	7
9		Elaboración de la documentación	3 días	lun 29/09/14	mié 01/10/14	8
10		Entrega PAC 1	1 día	jue 02/10/14	jue 02/10/14	9
11		**Análisis de requisitos y diagramas de clases**	**36 días**	**vie 03/10/14**	**vie 07/11/14**	**2**
12		Especificación de requisitos funcionales	7 días	vie 03/10/14	jue 09/10/14	
13		Esquematización de interfaz gráfica	7 días	vie 10/10/14	jue 16/10/14	12
14		Diagramas de clases	7 días	vie 17/10/14	jue 23/10/14	13
15		Diagramas de interrelación	7 días	vie 24/10/14	jue 30/10/14	14
16		Elaboración de la documentación	7 días	vie 31/10/14	jue 06/11/14	15
17		Entrega PAC 2	1 día	vie 07/11/14	vie 07/11/14	16
18		**Diseño**	**39 días**	**sáb 08/11/14**	**mar 16/12/14**	**11**
19		Diseño de arquitectura	3 días	sáb 08/11/14	lun 10/11/14	
20		Diseño de clases	3 días	mar 11/11/14	jue 13/11/14	19
21		Diseño de persistencia	3 días	vie 14/11/14	dom 16/11/14	20
22		**Implementación**	**20 días**	**lun 17/11/14**	**sáb 06/12/14**	
23		Desarrollo de clases	8 días	lun 17/11/14	lun 24/11/14	21
24		Desarrollo de interfaz gráfica	8 días	mar 25/11/14	mar 02/12/14	23
25		Elaboración de manuales	4 días	mié 03/12/14	sáb 06/12/14	24
26		**Verificación**	**6 días**	**dom 07/12/14**	**vie 12/12/14**	**25**
27		Pruebas unitarias	3 días	dom 07/12/14	mar 09/12/14	
28		Pruebas de integración	3 días	mié 10/12/14	vie 12/12/14	27
29		Elaboración de documentación	3 días	sáb 13/12/14	lun 15/12/14	28
30		Entrega PAC 3	1 día	mar 16/12/14	mar 16/12/14	29
31		**Depuración, Memoria y presentación**	**28 días**	**mié 17/12/14**	**mar 13/01/15**	**30**
32		Depuración de código	3 días	mié 17/12/14	vie 19/12/14	
33		Generación de aplicación definitiva	2 días	sáb 20/12/14	dom 21/12/14	32
34		Elaboración de la memoria del proyecto	11 días	lun 22/12/14	jue 01/01/15	33
35		Elaboración de la presentación del proyecto	6 días	vie 02/01/15	mié 07/01/15	34
36		Elaboración de vídeo	5 días	jue 08/01/15	lun 12/01/15	35
37		Entrega final (PAC 4)	1 día	mar 13/01/15	mar 13/01/15	36
38		**Tribunal**	**11 días**	**mié 14/01/15**	**sáb 24/01/15**	**37**
39		Preguntas del tribunal	8 días	mié 14/01/15	mié 21/01/15	
40		Entrega de respuestas	1 día	sáb 24/01/15	sáb 24/01/15	39

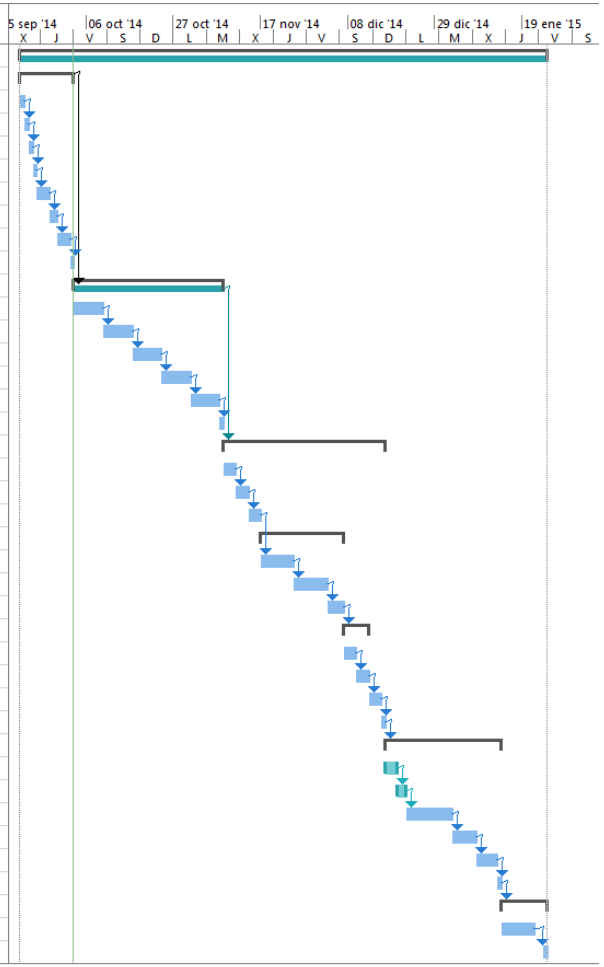

Ilustración 1 - Diagrama de Gantt de la planificación temporal

2. ESTADO DEL ARTE

2.1. Android

Android es un sistema operativo desarrollado inicialmente para teléfonos móviles, al igual que iOS, Symbian, Windows Mobile o Blackberry OS. La principal diferencia con estos últimos es, que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma. Y al igual que Linux, todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL**.**

El sistema Android permite programar aplicaciones en una variación de Java llamada Dalvik (máquina virtual que utiliza la plataforma para dispositivos móviles Android). El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, la gestión de llamadas, agenda, cámara, etc.) de manera simple mediante un lenguaje de programación altamente difundido como Java.

2.1.1. Historia

El sistema operativo Android fue desarrollado inicialmente por Android Inc. En el 2005, la empresa, fue comprada por Google. El 5 de noviembre de 2007 se lanzó la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio dedicados al desarrollo de estándares abiertos para dispositivos móviles y se proporcionó la primera versión de Android bajo la licencia Apache, una licencia libre y de código abierto, junto con el SDK dirigido a los desarrolladores para permitir crear aplicaciones para este sistema.

Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto a nivel mundial, con una cuota de mercado en torno al 40%. A nivel de España, en la actualidad, la venta de dispositivos móviles con sistema operativo Android está por encima del 90%.

Worldwide Device Shipments by Operating System (Thousands of Units)

Operating System	2012	2013	2014	2015
Android	503,690	877,885	1,102,572	1,254,367
Windows	346,272	327,956	359,855	422,726
iOS/Mac OS	213,690	266,769	344,206	397,234
RIM	34,581	24,019	15,416	10,597
Chrome	185	1,841	4,793	8,000
Others	1,117,905	801,932	647,572	528,755
Total	**2,216,322**	**2,300,402**	**2,474,414**	**2,621,678**

Source: Gartner (December 2013)

Ilustración 2 - Unidades vendidas 2012/13 y estimación de ventas 2014/15

Actualmente, Android dispone de una gran comunidad de desarrolladores programando aplicaciones para extender la funcionalidad de los dispositivos. A mediados del año 2013, la tienda de aplicaciones oficial de Android: Google Play, superó el millón de aplicaciones disponibles.

Google Play es la tienda de aplicaciones en línea administrada por Google, aunque también existe la posibilidad de obtener software en otros repositorios denominados markets. Las diferentes aplicaciones están escritos en el lenguaje de programación Java. No obstante, no es un sistema operativo libre de malware, aunque la mayoría de ello es descargado de sitios de terceros.

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (surface manager), un framework OpenCore, una base de datos relacional SQLite, una Interfaz de programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic. El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas de XML, 2,8 millones de líneas de lenguaje C, 2,1 millones de líneas de Java y 1,75 millones de líneas de C++.

2.1.2. Versiones

Desde la primera versión estable de Android, la 1.0, lanzada el 23 de septiembre de 2008 se han publicado otras 11 versiones, identificadas por un número y con un comercial de nombre de postre cuya inicial sigue un orden alfabético.

En la siguiente tabla se pueden comprobar las diferentes versiones del sistema operativo Android:

Id	Nombre de la versión	Fecha de lanzamiento
1.0	Apple pie	23 de septiembre de 2008
1.1	Banana Bread	9 de febrero de 2009
1.5	Cupcake	30 de abril de 2009
1.6	Donut	15 de septiembre de 2009
2.0/2.1	Éclair	26 de octubre de 2009
2.2	Froyo	20 de mayo de 2010
2.3	Gingerbread	6 de diciembre de 2010
3.0/3.1/3.2	Honeycomb	22 de febrero de 2011
4.0	Ice Cream Sandwich	19 de octubre de 2011
4.1/4.2/4.3	Jelly Bean	27 de junio de 2012
4.4	KitKat	31 de octubre de 2013
5	Lillipop	3 de noviembre de 2014

2.1.3. Características

Las principales características de Android son las siguientes:

- Diseño de dispositivos: dispone de una plataforma adaptable a diferentes tipos de pantallas, VGA, librerías de gráficos 2D y 3D.

- Almacenamiento de datos en SQLite, una base de datos liviana.

- Conectividad: Android soporta diferentes tecnologías de conectividad como GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, HSDPA, HSPA+, NFC, WiMAX, GPRS,UMTS y HSDPA+.

- Mensajería: SMS y MMS.

- Navegador web.

- Soporte de Java: aunque la mayoría de aplicaciones están escritas en Java, estas se compilan en un ejecutable Dalvik y se ejecutan en una máquina virtual Darvik.

- Soporte multimedia: Android soporta diferentes formatos multimedia como WebM, H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF y BMP.

- Soporte para streaming.

- Soporte para hardware adicional: cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros...

- Entorno de desarrollo: incluye un emulador de dispositivos, herramientas de depuración de memoria y análisis de rendimiento. Existe un plugin SDK Tools para el entorno de desarrollo de Eclipse.

- Google Play: market de aplicaciones oficial de Android.

- Multitáctil: soporte nativo para pantallas multitáctiles.

- Bluetooth.

- Video llamada.

- Multitarea.

- Características basadas en voz: como por ejemplo, para su uso en buscadores de internet.

- Tethering: permite al dispositivo ser usado como un punto de acceso alámbrico o inalámbrico.

2.1.4. Arquitectura

La plataforma Android dispone de una arquitectura interna básicamente formada por los siguientes componentes:

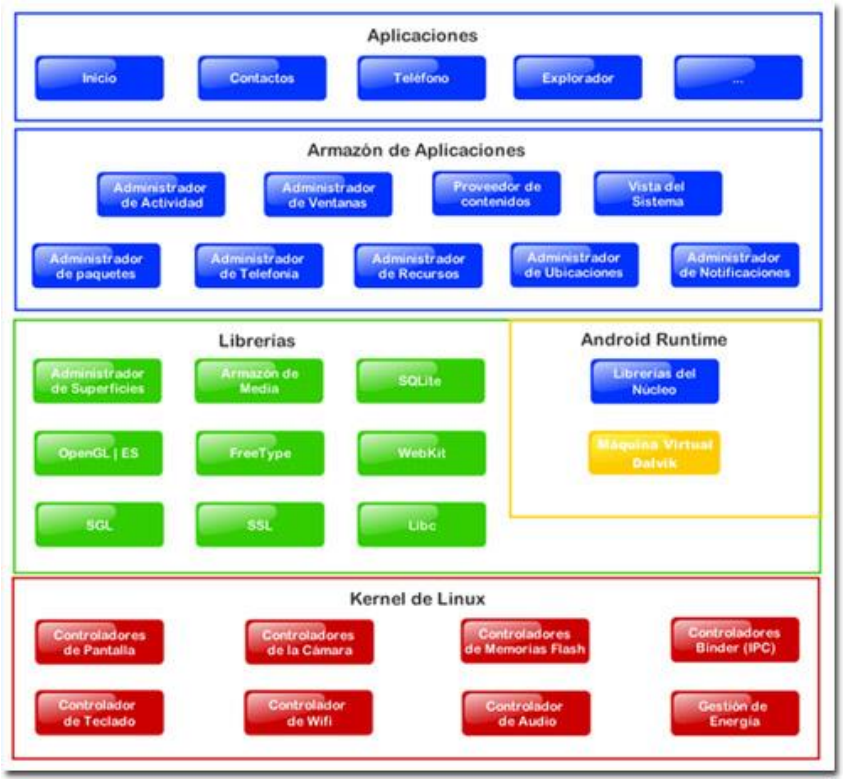

Ilustración 3 - Componentes de la plataforma Android

- Aplicaciones: todas las aplicaciones base creadas con la plataforma Android, incluyen un cliente de email (correo electrónico), programa de SMS, calendario, mapas, navegador, contactos, y algunos otros servicios. Todas las aplicaciones están escritas en el lenguaje de programación Java.

- Framework de aplicaciones: los desarrolladores de aplicaciones Android, tienen acceso total a los mismos API y código fuente utilizado en las aplicaciones base. El desarrollo de esta forma tiene como objetivo simplificar la reutilización de componentes, que respondan de la misma forma a la misma acción, dando la posibilidad de que los programas sean modificados o reemplazados por cualquier usuario sin tener que empezar a programar sus aplicaciones desde el principio. Cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

- Librerías o bibliotecas: Android incluye un conjunto de librerías C/C++ usadas por

diferentes componentes del sistema. Dichas características se exponen a todos los desarrolladores a través del framework de aplicaciones de Android como puede ser Android System C library (biblioteca C estándar), librerías de medios, librerías de gráficos, 3D o SQlite entre otras.

- Runtime de Android: Android incluye un conjunto de librerías que proporcionan la mayor parte de las funcionalidades disponibles en las librerías base del lenguaje de programación Java. Cada aplicación, ejecuta su propio proceso, con su propia máquina virtual de Dalvik. Dalvik permite la ejecución simultánea de varias máquinas virtuales de forma eficiente. Dalvik ejecuta ficheros con el formato Dalvik Executable (.dex). La Máquina Virtual está basada en registros, y corre clases compiladas por el compilador de Java que anteriormente han sido transformadas al formato .dex por la herramienta ''dx''.

- Núcleo Linux: Android depende de Linux para los servicios base del sistema como pueden ser la seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo Linux también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

2.1.5. Anatomía de una aplicación Android

Para la creación de aplicaciones Android, es esencial el uso de componentes. Cada componente es un punto diferente a través del cual el sistema puede entrar en la aplicación, existen como entidad propia y desempeñan un papel específico en el cual cada componente es un elemento único que ayuda a definir el comportamiento global de la aplicación.

Existen cuatro tipos distintos de componentes. Cada tipo es para un propósito en concreto y tiene un ciclo de vida distinto, que define cómo se crea y destruye el componente. Los componentes de construcción de una aplicación Android son:

- Actividad (Activity): representa una pantalla con una interfaz del usuario. El contenido visual de la ventana es proporcionado por una jerarquía de vistas (views), donde se lleva a cabo la interacción de la actividad con el usuario. Cada vista controla un espacio concreto de la ventana. Existen en Android una serie de vistas base listas para ser utilizadas: botones, campos de texto, menús, barras de desplazamiento...

- Servicio (Service): es un componente que se ejecuta en segundo plano durante un periodo indefinido de tiempo para llevar a cabo operaciones o realizar trabajos de procesos remotos. Por ejemplo, puede reproducir música en segundo plano mientras el usuario está navegando por internet.

- Proveedor de contenido (Content Provider): gestiona un conjunto de datos compartidos por una aplicación. Los datos se pueden almacenar en cualquier zona de almacenamiento persistente accesible por la aplicación. Mediante los

proveedores de contenido, otras aplicaciones pueden consultar o modificar la información compartida.

- Receptor de Notificaciones (Broadcast Receiver): es un componente que responde a la emisión de un mensaje en el sistema. Las aplicaciones Android pueden iniciar emisiones.

La comunicación entre los diferentes procesos se lleva a cabo mediante Intents. Los intents constituyen el core del sistema de mensajes en Android y definen un mensaje para activar un componente en particular. Por ejemplo, si se quiere invocar una nueva actividad desde la actividad actual, se necesita lanzar un intent especificando la nueva actividad.

No todas las aplicaciones tienen la necesidad de tener los cuatro componentes, pero cualquier aplicación estará formada por una combinación de estos. Una vez decididos los componentes necesarios para la aplicación, estos, deben enumerarse en un archivo XML llamado AndroidManifest, que, aunque no forma parte del código principal de la aplicación Android, es necesario para el correcto funcionamiento de la aplicación. En él, se declaran los componentes que pertenecen a la aplicación así como sus capacidades y requerimientos.

Ilustración 4 - Componentes de una aplicación Android

2.1.6. Ciclo de vida de una aplicación Android

Cada aplicación Android corre en su propio proceso, el cual es creado por la aplicación cuando se ejecuta y permanece hasta que la aplicación deja de trabajar o el sistema necesita memoria para otras aplicaciones y es liberado. Android coloca cada proceso en una jerarquía de importancia basada en los siguientes estados:

- Procesos en primer plano (Active process): es un proceso que aloja una Activity en la pantalla y con la que el usuario está interactuando (su método onResume() ha sido llamado) o que un IntentReceiver está ejecutándose. Este tipo de procesos serán eliminados como último recurso si el sistema necesitase memoria.

- Procesos visibles (Visible process): es un proceso que aloja una Activity pero no está en primer plano (como por ejemplo cuando la aplicación muestra al usuario un cuadro de diálogo). Este tipo de procesos no será eliminado en caso que sea necesaria la memoria para mantener a todos los procesos del primer plano corriendo.

- Procesos de servicio (Started service process): es un proceso que aloja un Service que ha sido iniciado con el método startService(). Este tipo de procesos no son visibles y suelen ser importantes para el usuario (conexión con servidores, reproducción de música).

- Procesos en segundo plano (Background process): Es un proceso que aloja una Activity que no es actualmente visible para el usuario. Normalmente la eliminación de estos procesos no suponen un gran impacto para la actividad del usuario. El sistema guarda una lista para asegurar que el último proceso visto por el usuario sea el último en eliminarse en caso de necesitar memoria.

- Procesos vacíos (Empty process): Es un proceso que no aloja ningún componente. El sistema elimina este tipo de procesos con frecuencia para obtener memoria disponible.

Según esta clasificación, Android prioriza los procesos existentes en el sistema y decide cuáles han de ser eliminados, con el fin de liberar recursos y poder lanzar la aplicación requerida.

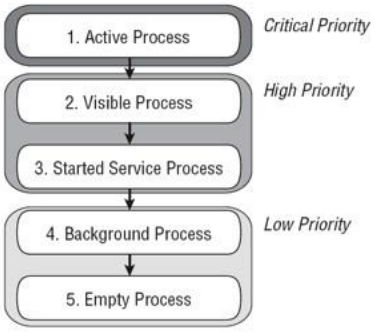

Ilustración 5 - Prioridad de procesos en Android

El hecho de que cada aplicación se ejecuta en su propio proceso aporta beneficios en cuestiones básicas como seguridad, gestión de memoria, o la ocupación de la CPU del dispositivo móvil. Android se ocupa de lanzar y parar todos estos procesos, gestionar su ejecución y decidir qué hacer en función de los recursos disponibles y de las órdenes dadas por el usuario.

Android es capaz de lanzar tantos procesos como permitan los recursos del dispositivo. Cada proceso, correspondiente a una aplicación, estará formado por una o varias actividades independientes (componentes Activity) de esa aplicación. Cuando el usuario se mueve entre las diferentes actividades, o abre una nueva aplicación, el sistema duerme dicho proceso y realiza una copia de su estado para poder recuperarlo más tarde. El proceso y la actividad siguen existiendo en el sistema, pero están dormidos y su estado ha sido guardado. Es entonces cuando crea, o despierta si ya existe, el proceso para la aplicación que debe ser lanzada, asumiendo que existan recursos para ello.

Cada uno de los componentes de Android tiene un ciclo de vida bien definido, por lo que el desarrollador puede controlar en cada momento en qué estado se encuentra dicho componente, pudiendo así programar las acciones que mejor convengan. El componente Activity, probablemente el más importante, tiene un ciclo de vida detallado en la siguiente imagen:

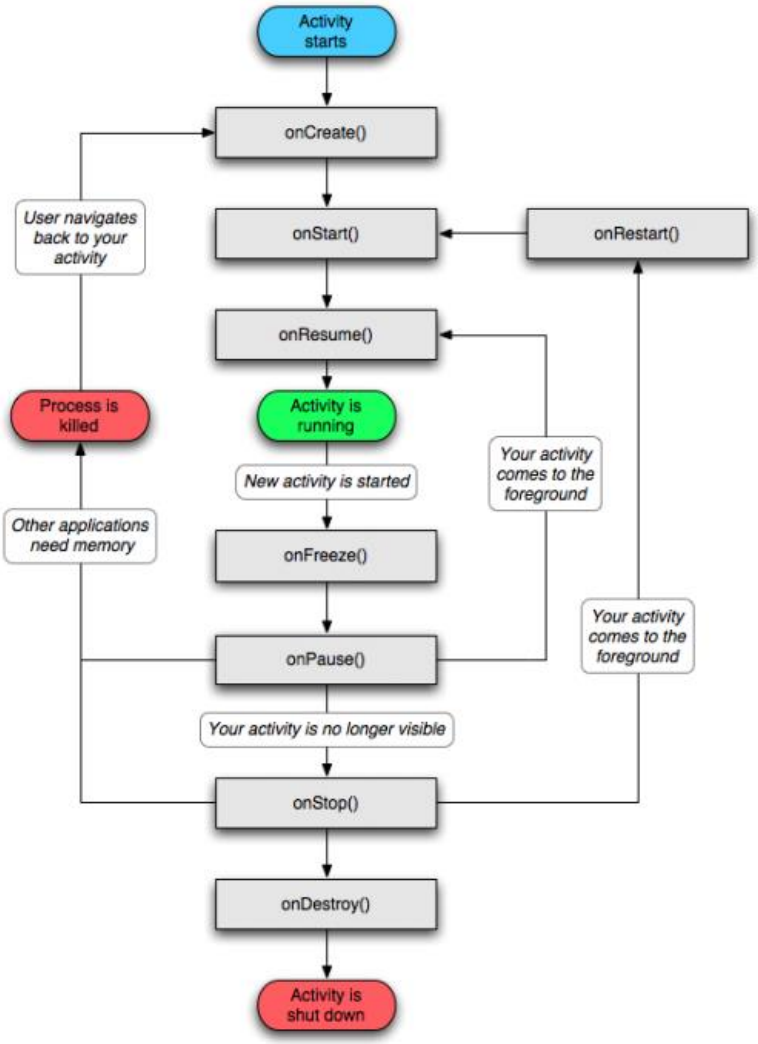

Ilustración 6 - Ciclo de vida de un Activity en Android

2.2. Eclipse

Eclipse es un entorno de desarrollo integrado para programación en Java (además de otros lenguajes de programación como C++, PHP… utilizando las extensiones apropiadas). Se trata de un sistema abierto, ya que su diseño permite que terceras partes lo extiendan fácilmente. Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) ya que proporciona herramientas con las que gestionar espacios de trabajo; construir, lanzar y depurar aplicaciones…

Eclipse es una plataforma, no es una aplicación terminada por sí misma, sino que está diseñada para ser extendida indefinidamente con herramientas y plugins. Se puede decir que Eclipse es un marco de trabajo para desarrollar aplicaciones, siendo los IDE (de cualquier lenguaje) una de estas aplicaciones.

El pilar fundamental de Eclipse es un mecanismo para descubrir, integrar y ejecutar distintos módulos llamados plugins a través de los cuáles va creciendo la plataforma.

Eclipse comenzó como un proyecto propietario, dirigido por OTI (Object Technology International), empresa adquirida por IBM quien pretendía reducir el gran número de entornos de desarrollo incompatibles que ofrecía a sus clientes, a la vez que incrementar el número de componentes reutilizables en esos mismos entornos. Al utilizar un marco de trabajo común los equipos de desarrollo podrían apoyarse en componentes proporcionados por otros, integrarlos y permitir que los desarrolladores pasaran de una herramienta o componente a otro sin ningún problema.

El proyecto Eclipse surgió a partir de una larga línea de productos de entornos de desarrollo, los primeros de los cuales son IBM VisualAge para Smalltalk e IBM VisualAge para Java, ambos escritos en Smalltalk.

Un pequeño grupo de expertos se dispuso entonces a recoger toda la experiencia de años diseñando e implementando entornos de desarrollo y plasmarlo en un nuevo concepto, solucionando los problemas anteriormente mencionados. El resultado fue Eclipse, una plataforma diseñada desde los cimientos como una plataforma de integración para herramientas de desarrollo. Permitió que los desarrolladores extendieran y ampliaran fácilmente productos basados en ella.

El proyecto de código abierto Eclipse fue anunciado en noviembre de 2001 y nació la licencia pública Eclipse. Líderes de la industria tales como Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft and Webgain formaron la lista inicial de organizadores eclipse.org.

Hacia finales de 2003, el consorcio contaba ya con más de 80 miembros. El 2 de febrero de 2004, la junta de administradores de Eclipse anunció la remodelación del consorcio como una corporación sin ánimo de lucro, en beneficio de los colaboradores de la plataforma y de los usuarios finales. Actualmente, Eclipse es una comunidad de código abierto cuyos proyectos se centran en construir una plataforma de desarrollo

extensible, y un marco de trabajo para aplicaciones que permitan el desarrollo, uso y administración del software (siendo éste de cualquier propósito, no solamente IDEs) durante todo su ciclo de vida.

La descripción más generalista de la arquitectura de Eclipse suele ser una visión abstracta del entorno de desarrollo y ejecución para la creación y despliegue de plugins, que proporcionan una cierta funcionalidad. Las nuevas herramientas intercambian información con la plataforma de cliente enriquecido a través de un mecanismo de plugins. Los plugins son componentes estructurados que se configuran utilizando un archivo de manifiesto (un archivo xml que contiene la descripción del plugin) en el sistema de ejecución de Eclipse.

El siguiente diagrama muestra los principales componentes y APIs de la plataforma Eclipse:

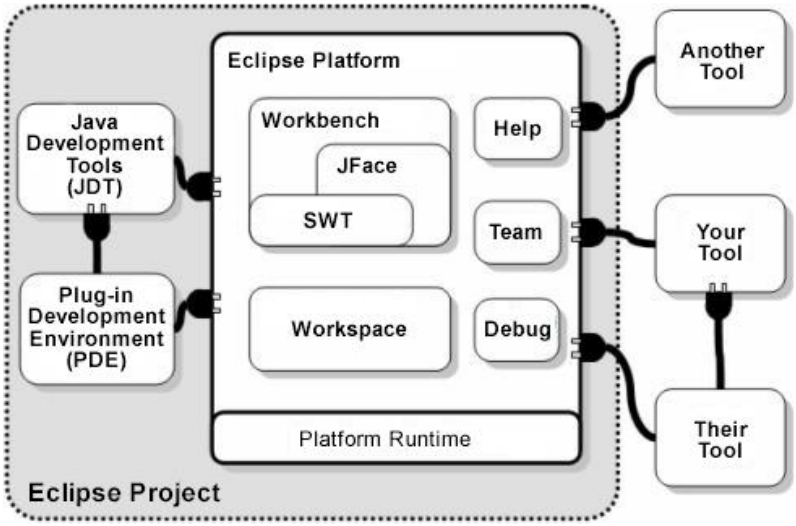

Ilustración 7 – Principales Componentes y APIs de Eclipse

Cada componente de la plataforma Eclipse proporciona una funcionalidad para que los desarrolladores se puedan concentrar en su área de competencia, al tiempo que se mantiene una arquitectura abierta y ampliable:

- El entorno de trabajo (Workbench), que incluye el kit de herramientas Standard Widget Toolkit y los widgets JFace, proporciona la interfaz de usuario.
- El sistema de ayuda ofrece documentos creados mediante plugins ampliables.
- El espacio de trabajo (Workspace) es el modelo de datos para la aplicación en desarrollo, y proporciona una API para crear y gestionar los recursos Eclipse.
- Características tales como la gestión de repositorios las proporciona también la plataforma.

La plataforma de ejecución descubre y carga plugins de forma dinámica, y mantiene información sobre los plugin y sus puntos de extensión.

2.3. MySQL

"MySQL es un sistema de administración de bases de datos relacional, RDBMS y utiliza el lenguaje de consulta estructurado (SQL) mediante el cual se llevarán a cabo diferentes acciones sobre una base de datos." (Gilgillan, 2003).

MySQL es un software de código abierto, licenciado bajo la GPL de la GNU y creado por la empresa MySQL AB. MySQL utiliza el lenguaje de programación Structured Query Language (SQL), el cual fue desarrollado por IBM en 1981 y desde entonces es utilizado de forma generalizada en las bases de datos relacionales.

MySQL surgió alrededor de la década del 90, Michael Windenis comenzó a usar mSQL para conectar tablas usando sus propias rutinas de bajo nivel (ISAM). Tras unas primeras pruebas, llegó a la conclusión de que mSQL no era lo bastante flexible ni rápido para lo que necesitaba, por lo que tuvo que desarrollar nuevas funciones. Esto resulto en una interfaz SQL a su base de datos, totalmente compatible a mSQL.

Las principales características de MySQL son:

- Velocidad: a la hora de ejecutar consultas SQL, MySQL es un sistema con una velocidad superior a sus diferentes rivales, incluido Oracle. MySQL aprovecha la potencia de sistemas multiproceso, gracias a su implementación multihilo.
- Portabilidad: Puede ejecutarse en la inmensa mayoría de plataformas y sistemas operativos, por lo que junto a un lenguaje de programación del lado del servidor de alta portabilidad como Java, PHP, Perl…, permite el desarrollo de aplicaciones Web fáciles de migrar y el acceso y copia de los datos desde cualquier sistema operativo.
- Seguridad: MySQL dispone de un sistema de privilegios y contraseñas muy flexible y seguro, que permite verificación basada en el Host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado una vez se conecta con un servidor.
- Escalabilidad y límites: MySQL soporta a grandes bases de datos. Se usa con bases de datos que tienen 50 millones de registros y con 60.000 tablas. Se permiten hasta 64 índices por tabla. Soporta gran cantidad de tipos de datos para las columnas.
- Conectividad: Los clientes pueden conectar con el servidor MySQL usando sockets TCP/IP en cualquier plataforma.
- Localización: El servidor soporta mensajes de error en diferentes idiomas.

2.4. JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition (Diciembre 1999). JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los

programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, se presentan de estas formas:

Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con { (llave de apertura) y termine con } (llave de cierre). Cada nombre es seguido por : (dos puntos) y los pares nombre/valor están separados por , (coma).

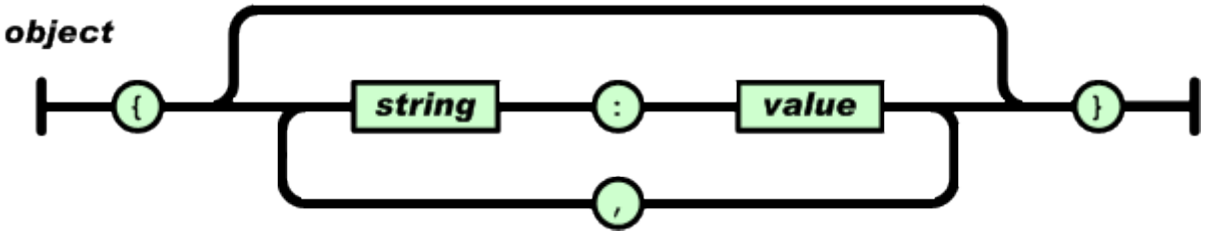

Ilustración 8 - Construcción de un objeto JSON

Un arreglo es una colección de valores. Un arreglo comienza con [(corchete izquierdo) y termina con] (corchete derecho). Los valores se separan por , (coma).

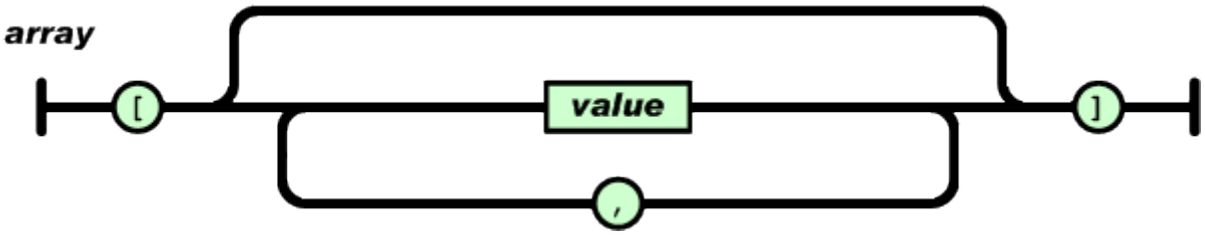

Ilustración 9 - Construcción de un array JSON

Un valor puede ser una cadena de caracteres con comillas dobles, o un número, o true o false o null, o un objeto o un arreglo. Estas estructuras pueden anidarse.

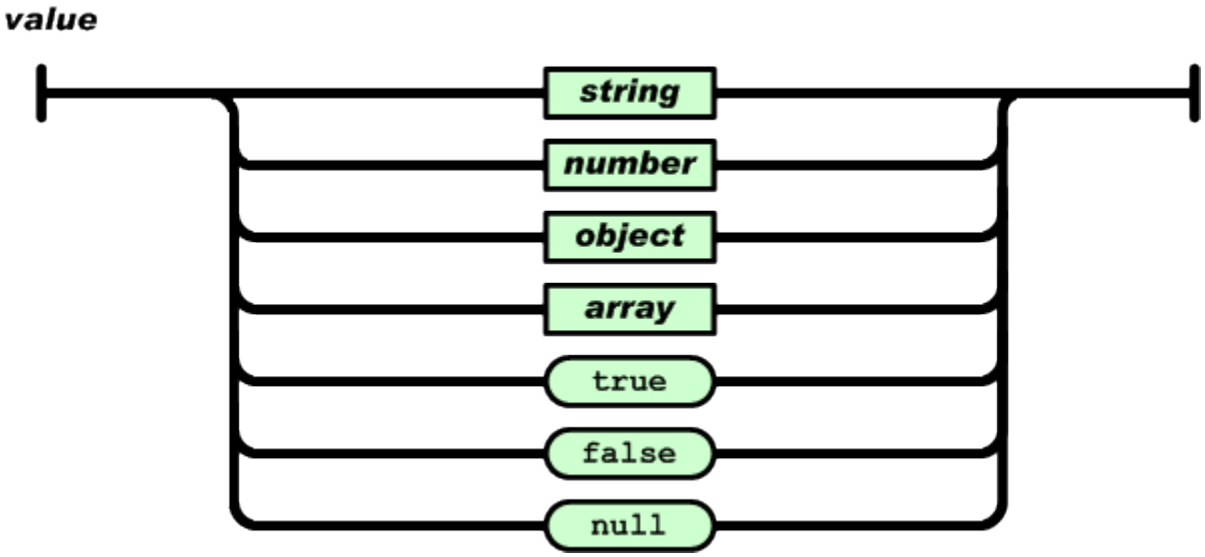

Ilustración 10 - Opciones válidas para la formación de un value en JSON

Una cadena de caracteres es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape. Un carácter está representado por una cadena de caracteres de un único carácter. Una cadena de caracteres es parecida a una cadena de caracteres C o Java.

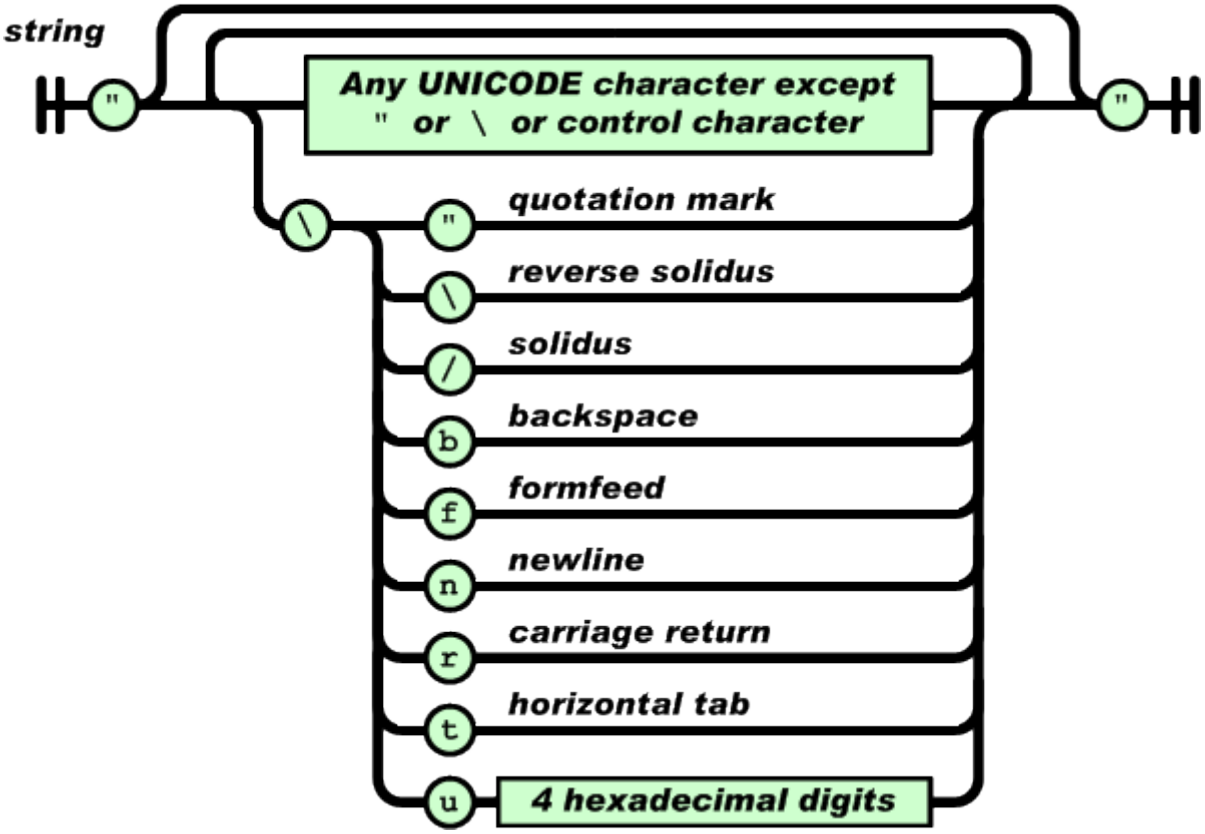

Ilustración 11 - Formación de un string en JSON

Un número es similar a un número C o Java, excepto que no se usan los formatos octales y hexadecimales.

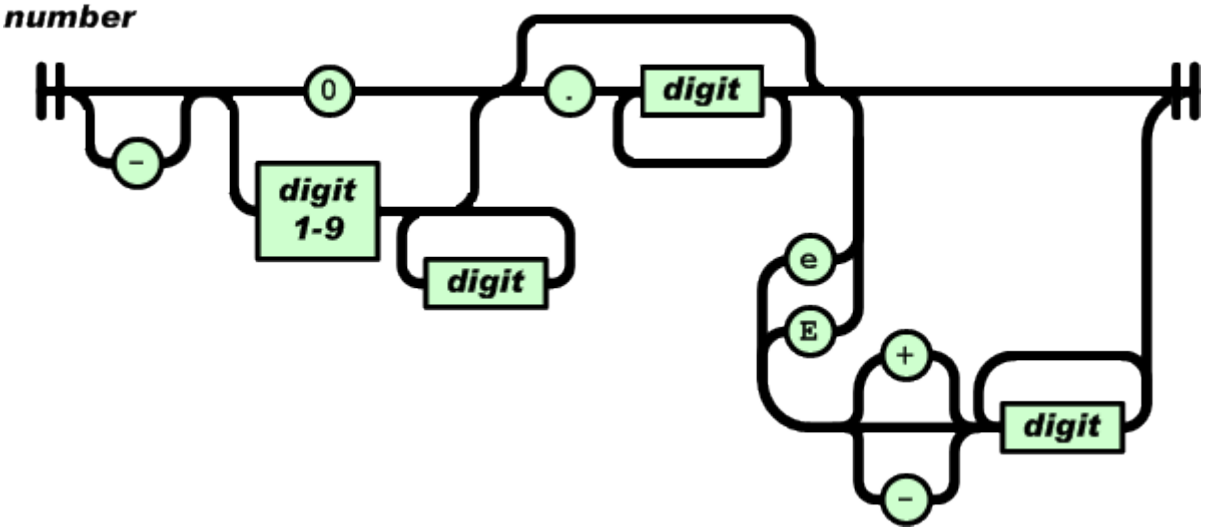

Ilustración 12 - Formación de un number en JSON

Los espacios en blanco pueden insertarse entre cualquier par de símbolos.

Exceptuando pequeños detalles de encoding, esto describe completamente el lenguaje.

2.5. REST Web Services

La Transferencia de Estado Representacional (Representational State Transfer) o REST es una familia de arquitecturas. Se puede considerar como arquitectura REST cualquier arquitectura de servicios distribuidos que cumpla con los siguientes requisitos:

- No se publican servicios RPC. En arquitecturas REST, los servicios no publican un conjunto arbitrario de métodos u operaciones.

- En REST lo que se publica son recursos. Un recurso se puede considerar como una entidad que representa un concepto de negocio que puede ser accedido públicamente.

- Cada recurso, de acuerdo a los principios de OO, posee un identificador único y global, que lo distingue de cualquier otro recurso, aunque ambos tuvieran exactamente los mismos datos.

- Cada recurso posee un estado interno, que no puede ser accedido directamente desde el exterior. Lo que sí es accesible desde el exterior es una o varias representaciones de dicho estado. Por representación se entiende un formato de datos concreto usado para la transferencia de una copia del estado público del recurso entre el cliente y el servidor. La implementación del recurso decide qué información es visible o no desde el exterior, y que representaciones de dicho estado se soportan.

- En REST todos los recursos comparten una interfaz única y constante. Todos los recursos tienen las mismas operaciones. Las operaciones nos permiten

manipular el estado público del recurso. En un sistema REST típico se definen cuatro operaciones.

- Create: el cliente manda al servidor una petición para crear un nuevo recurso. Opcionalmente el cliente puede mandar una representación del estado inicial de este recurso. El servidor responde con el identificador global del nuevo recurso.

- Delete: el cliente elimina un recurso del servidor. El cliente necesita saber el identificador del recurso.

- Read: el cliente puede leer una representación del estado de un recurso, identificado con su identificador global. El cliente puede especificar qué tipos de representaciones entiende. Se copia el estado del recurso en el servidor y se pega en el cliente. Ambas copias del estado no se mantiene sincronizadas. El servidor puede cambiar el estado real del recurso y el cliente, de forma independiente, puede modificar su copia local del estado del recurso.

- Update: como el servidor y el cliente tienen una copia diferente del estado, el cliente puede usar esta operación para sobrescribir o grabar su copia del estado en el servidor. De esta manera se puede actualizar el estado del recurso con las modificaciones hechas en el cliente.

La implementación del servicio es libre de prohibir alguno de estos métodos para un recurso en concreto. También debe definir el modelo de datos que se va a publicar y que representaciones soporta. Lo que no puede hacer es inventarse operaciones de forma arbitraria. Las operaciones son siempre las mismas.

Los distintos recursos se pueden interrelacionar y referenciar entre sí mediante sus identificadores globales.

2.6. Hibernate

Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones. Hibernate es software libre, distribuido bajo los términos de la licencia GNU GPL.

Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación

manipular los datos en la base de datos operando sobre objetos, con todas las características de la programación orientada a objetos.

Hibernate convierte los datos entre los tipos utilizados por Java y los definidos por SQL. Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

Hibernate ofrece también un lenguaje de consulta de datos llamado HQL (Hibernate Query Language), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "criteria").

Hibernate para Java puede ser utilizado en aplicaciones Java independientes o en aplicaciones Java EE, mediante el componente Hibernate Annotations que implementa el estándar JPA, que es parte de esta plataforma.

3. ANÁLISIS DEL SISTEMA

3.1. Descripción general

La aplicación desarrollada tiene como objetivo el registro y localización de mascotas extraviadas utilizando diferentes tecnologías disponibles hoy en día para un amplio número de usuarios. La aplicación se ejecutará en dispositivos móviles, smartphones o tablets, muy popularizados hoy en día, por lo que se trabajará con datos itinerantes a través de diferentes plataformas (wifi, red móvil…). Por tanto, se hace imprescindible tener en cuenta diferentes factores, como pueden ser, la optimización de las comunicaciones tanto a nivel de volumen como de velocidad (transfiriendo el menor número de información posible), la limitación del almacenamiento de los diferentes dispositivos tratando para ello de realizar una aplicación liviana y posibilitando con todo ello, un manejo de la aplicación fluido y eficiente.

Un esquema resumido de la estructura del sistema sería el siguiente:

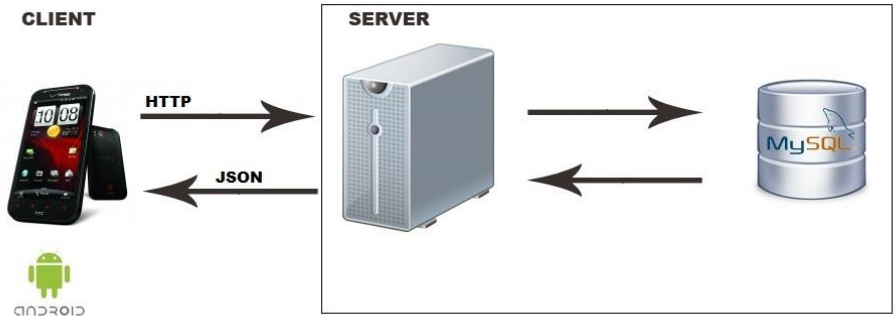

Ilustración 13 - Estructura del sistema

Como cliente, se dispondrá de un dispositivo Android. Dicho dispositivo, comunicará con la parte servidor mediante un protocolo de servicios web (utilizando para ello una comunicación itinerante como puede ser WIFI, red móvil, etc). En la parte servidor se ejecutará el core de la aplicación, el cual comunicará con una base de datos MySQL en la cual se almacenarán todos los datos necesarios para la herramienta.

Se dispondrá por tanto de una arquitectura cliente-servidor, la cual se describirá en mayor detalle en capítulos sucesivos.

3.2. Requisitos del sistema

3.2.1. Requisitos funcionales

Se ha planteado una aplicación que pueda ser utilizada tanto para usuarios registrados como para los no registrados. El objetivo de este planteamiento es poder alcanzar un mayor número de usuarios y por tanto la colaboración de todo usuario con acceso a la herramienta para alcanzar el objetivo principal de la misma, la localización de mascotas perdidas. Sin embargo, para los usuarios registrados se permitirá un mayor número de opciones que se detallarán más adelante.

3.2.1.1.GESTION DE USUARIOS

Alta de usuarios

El alta de usuarios convierte a un usuario común en un usuario registrado, y tener así acceso a características propias de la herramienta dedicadas a este tipo de usuarios. A partir de un formulario, se debe permitir registrar nuevos usuarios en la herramienta.

Baja de usuarios

Al igual que para el registro, la herramienta debe permitir al usuario eliminar todos sus datos de la misma y pasar a ser un usuario común. Por tanto, el sistema debe permitir dar de baja usuarios del sistema (extendiendo dicha baja a las posibles mascotas que este pudiera tener registradas).

Modificación de usuarios

Mediante el alta o registro de usuario (visto anteriormente), se debe permitir el acceso al sistema a un usuario como usuario registrado, sin embargo, se ha pensado que esto debe realizarse de la forma más rápida y simple posible. Una vez dado de alta un usuario, se estima necesario complementar dicha alta con el registro de un mayor número de datos que pudieran ser utilizados para facilitar la localización de un animal perdido (podría establecerse un criterio por proximidad, etc). Por tanto, se debe permitir tanto ampliar la información referente al usuario como modificar posteriormente los mismos (dirección, contraseña…). Para ello, se dispondrá de un apartado de modificación de datos de usuario.

3.2.1.2.GESTION DE MASCOTAS

Alta de mascotas

El objetivo principal de la aplicación es la localización de mascotas extraviadas, por ello, es necesaria la existencia de un apartado en el cual, un usuario registrado pueda registrar o dar de alta las diferentes mascotas que desee que puedan ser gestionadas por el sistema. A partir de un formulario, el usuario podrá crear las diferentes mascotas con los datos identificativos de estas últimas que permitan posteriores identificaciones en caso de pérdida.

Baja de mascotas

Al igual que el alta de la mascota, el sistema debe permitir que un usuario registrado gestione los diferentes animales que posee. Estos, pueden variar a lo largo del tiempo, por lo que se estima necesaria la gestión de bajas de mascotas. Los usuarios registrados por tanto, podrán dar de baja mascotas que anteriormente hayan registrado.

Modificación de datos de mascotas

Pese a que la mayoría de los datos de una mascota son los mismos a lo largo de su ciclo de vida, al igual que un ser humano, cualquier mascota cambia por ejemplo su fisionomía a lo largo del tiempo. El sistema debe permitir por tanto, no solo el alta y la baja de la mascota sino una modificación de sus datos (como pudiera ser la imagen de la misma, datos característicos que pudieran facilitar su localización, un collar, etc). Una de las funcionalidades que se incluirá en este apartado, será la de establecer si una mascota está en situación de perdida (entre otros motivos para permitir una gestión más sencilla y rápida desde la propia gestión de las mascotas).

Los usuarios registrados podrán visualizar y modificar los datos de las mascotas que hayan registrado anteriormente a partir de un formulario.

3.2.1.3. GESTION DE MENSAJES

Notificación de mascotas localizadas

La principal funcionalidad de la aplicación es, la localización de mascotas. Para ello, se tienen que dar varias premisas, en primer lugar, que una mascota se encuentre extraviada, en segundo lugar, que un actor notifique la localización de una mascota, la cual, coincida con la mascota perdida y en tercer lugar la validación de que dicha mascota es el animal buscado. En esta funcionalidad, el usuario puede registrar una localización de mascota.

Listado de mascotas localizadas

Los usuarios registrados pueden ver un listado con todas las posibles localizaciones de una mascota que haya sido definida como perdida. Para ello, desde el menú del usuario tendrá la posibilidad de acceder a un gestor de mensajes. Estos mensajes serán generados a partir de posibles coincidencias entre las mascotas notificadas como localizadas y las mascotas del usuario definidas como perdidas. En caso de existir coincidencias, se generará un mensaje asociado al usuario que permitirá una gestión individual del mismo para su posterior validación, notificar un falso positivo…

Selección de mascota

Los usuarios registrados podrán seleccionar entre los diferentes resultados de posibles localizaciones de una mascota definida como perdida para su validación, en el mensaje, el usuario podrá confirmar la identidad de la mascota perdida, con lo que se tendrá acceso a la información compartida por el usuario que generó la notificación. En otro caso, el usuario podrá marcar el mensaje como falso positivo.

3.2.1.4.SISTEMA DE LOGIN

Login / Logout

Tras haber realizado un registro en la aplicación, el sistema debe permitir a cada usuario el acceso con seguridad a la aplicación y garantizar que cada usuario solo puede ver su área privada correspondiente. Para ello, se dispondrá de un sistema de acceso o login a la aplicación que identifique que el usuario que accede a la misma es único.

Además del acceso, la herramienta debe permitir a los usuarios poder realizar un logout de la aplicación y limpiar por tanto los datos referentes a la sesión iniciada en la herramienta.

3.2.1.5.ESTADÍSTICAS

Se establecerá un área de estadísticas de la aplicación con información útil al actor Administrador del sistema donde poder ejecutar estadísticas y poder analizar y conocer el estado del funcionamiento de la aplicación.

3.2.2. Requisitos no funcionales

"Los usuarios tienen expectativas implícitas acerca de cómo debe funcionar un software. Esas características incluyen cuán sencillo tiene que ser el uso del software, cómo de rápido se ejecuta, cómo es de fiable y cómo se comporta cuando se producen condiciones inesperadas." (Stellman-Greene, 2006).

Los requisitos no funcionales son aquellos que debe cumplir un proyecto para garantizar aspectos que no tienen que ver directamente con el funcionamiento de las diferentes opciones de la aplicación. Una aplicación no sólo debe realizar aquello para lo que se diseñó, sino que tiene que obtener estos resultados teniendo en cuenta diversos aspectos como por ejemplo su usabilidad, su seguridad, su eficiencia, etc.

Los requisitos más importantes que deberá cumplir la aplicación serán:

- Usabilidad: resulta imprescindible para la aplicación el hecho de que sea clara, intuitiva y que tenga un uso fácil de comprender y utilizar por los usuarios potenciales, en caso contrario, la consecuencia es el abandono de la herramienta. El tiempo de aprendizaje de la aplicación debe ser el menor posible.

- Eficiencia: la eficiencia especifica en qué medida el software es capaz de utilizar correctamente los diferentes recursos (CPU, uso de memoria, espacio en disco, ancho de banda…). Será necesario por tanto que la respuesta por parte de la aplicación sea lo más rápida y fiable posible, garantizando un uso fluido de la misma, lo cual tendrá como consecuencia la satisfacción del usuario.

- Integridad: los requerimientos de integridad definen la seguridad de los diferentes atributos del Sistema, la restricción del acceso a diferentes

características o datos para diferentes usuarios y la protección de los datos introducidos en el sistema. Será necesario disponer de un acceso a los datos de usuario. Cada usuario, solo debe tener acceso a sus propios datos y a los datos de terceros únicamente tras haber sido aceptados por los mismos. Será necesario por tanto cumplir las leyes de protección de datos (Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal, LOPD).

- Escalabilidad: un software escalable tiene la posibilidad de adaptarse a una amplia variedad de configuraciones. En caso de que el número de usuarios fuese elevado, sería necesario que pudiera ser ampliada y escalada de forma sencilla sin necesidad de modificar el código de la aplicación.

3.3. Casos de uso

"Un caso de uso captura un contrato entre los stakeholders del sistema acerca de su comportamiento. El caso de uso describe el comportamiento del sistema bajo varias condiciones en respuesta a una petición de uno de los stakeholders, llamado actor principal. El actor principal inicia una interacción con el sistema para cumplir algún objetivo. El sistema responde, protegiendo los intereses de todos los stakeholders. Diferentes comportamientos o escenarios, pueden ir apareciendo poco apoco, dependiendo de las peticiones particulares realizadas y de las condiciones alrededor de las peticiones. Un caso de uso recopila estos diferentes escenarios.

Los casos de uso son especificados fundamentalmente en forma textual, aunque también pueden ser descritos como diagramas de flujo, de secuencia, redes de Petri o mediante diferentes lenguajes de programación. Bajo circunstancias normales, sirven para transmitir información entre dos personas, incluso entre personas sin un conocimiento específico." (Cockburn, 2000).

Para la representación de los casos de uso, utilizaremos por una parte la descripción textual, para la especificación de la información de cada funcionalidad y por otra parte, para la representación gráfica del diagrama de casos de uso, utilizaremos el lenguaje UML.

El diagrama de casos de uso de la aplicación sería el siguiente:

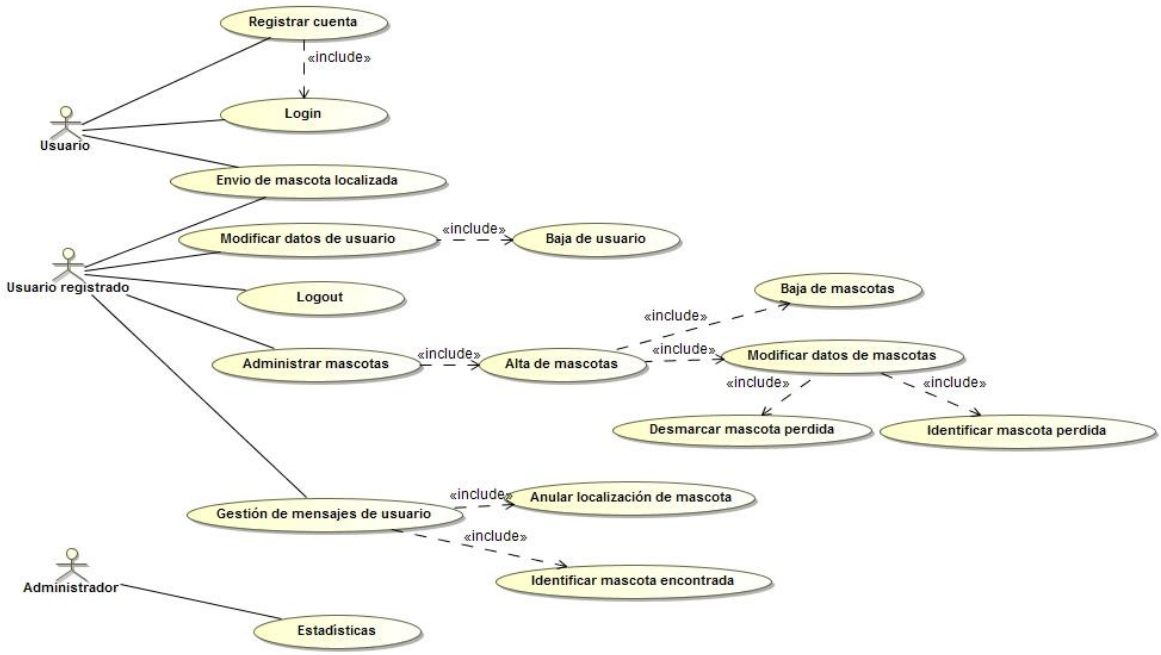

Ilustración 14 - Diagrama de casos de uso

En cuanto a la representación textual, se realizará una especificación de los detalles de cada una de las funcionalidades, utilizando para ello, tablas de representación con campos referidos a aspectos o características de la naturaleza del caso de uso. El significado de cada uno de los campos de las diferentes tablas sería el siguiente:

- **Id**: identificador único asociado al caso de uso. La nomenclatura vendrá dada por CU-XX, siendo CU las siglas correspondientes a "Caso de Uso" y XX un número entero incremental cuyo primer valor será 01. De este modo, conseguiremos un identificador único para cada caso de uso.

- **Caso de uso**: Nombre descriptivo del caso de uso.

- **Casos de uso relacionado**: Casos de uso que tienen una interrelación con el caso de uso analizado.

- **Actores:** Roles de usuario implicados en el caso de uso y que por tanto, tienen la posibilidad de ejecutarlo.

- **Objetivo:** Descripción de la funcionalidad atribuida al caso de uso, y la razón de su existencia.

- **Precondiciones:** Condiciones previas necesarias que deben cumplirse para poder llevar a cabo el caso de uso.

- **Postcondiciones:** Efectos y consecuencias que provoca la ejecución del caso de uso en el sistema.

- **Escenario básico:** Descripción del flujo principal del caso de uso en relación a la interacción entre el actor y el sistema.

- **Escenario alternativo:** Alternativas al caso de uso, diferentes al escenario básico, que se pueden llevar a cabo debido a la evaluación de condiciones de error u otros factores.

3.3.1. Actores

"Un actor conforma un rol jugado por entidades externas (como usuarios humanos u otros sistemas) que interactúan con el sistema intercambiando señales y datos. En cualquier caso, la entidad representada por un actor es externa al sistema. Aunque los actores existen en el modelo, representan entidades que no son parte del sistema modelado. Son usados en el modelo de análisis para definir los entornos del sistema.

Un actor representa un rol, no necesariamente a una entidad física específica." (Milicev, 2009).

En la aplicación, se definen tres tipos de actores:

- Usuario: será toda persona que ejecuta la aplicación en un dispositivo móvil.

- Usuario registrado: serán aquellas personas que se han registrado en el sistema creando una cuenta y además acceden a la aplicación mediante su correo electrónico y contraseña.

- Administrador: serán aquellas personas definidas internamente en la aplicación como administradores del sistema.

3.3.2. Casos de uso del módulo de conexión

Caso de uso: Registrar cuenta

Id	CU-01	**Caso de uso**	Registrar cuenta
Casos de uso relacionados			
Actores		Usuario	
Objetivo		Permite al usuario registrarse en la aplicación y crear una nueva cuenta.	
Precondiciones		No puede existir una cuenta con el mismo correo electrónico (username).	
Postcondiciones		Se crea la cuenta de usuario y la aplicación redirigirá al usuario a su pantalla inicial de la aplicación.	
Escenario básico		1. El sistema muestra la pantalla principal. 2. El usuario selecciona la opción "Register" 3. El sistema muestra la pantalla "Register" con el formulario. 4. El usuario introduce su nombre, dirección de correo electrónico y contraseña y pulsa "Register". 5. El sistema crea la cuenta y redirige al usuario	

	a la página “My profile”.
Escenario alternativo	2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 4a. El usuario introduce algún dato incorrecto o algún campo vacío. 4a1. El sistema muestra un mensaje identificando el error. 5a. El sistema no puede conectar al servidor. 5a1. El sistema muestra un mensaje indicando que no se ha podido crear la cuenta. 5b. El sistema detecta que ya existe una cuenta relacionada con ese email. 5b1. El sistema muestra un mensaje indicando que ya existe la cuenta.

Caso de uso: Login de la aplicación

Id	CU-02	**Caso de uso**	Login en la aplicación
Casos de uso relacionados		Registrar cuenta	
Actores		Usuario	
Objetivo		Permite al usuario acceder a las funcionales privadas de la aplicación asociadas a su usuario.	
Precondiciones		Debe existir una cuenta registrada	
Postcondiciones		El usuario accede y tiene visibilidad de la parte privada asociada a su usuario en la aplicación.	
Escenario básico		1. El sistema muestra la pantalla principal. 2. El usuario selecciona la opción “Login”. 3. El sistema muestra la pantalla “Login” con el formulario. 4. El usuario introduce su nombre de usuario (dirección de correo electrónico) y la contraseña y pulsa “Login”. 5. El sistema redirige al usuario a la página “My profile”.	
Escenario alternativo		2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 4a. El usuario introduce algún dato incorrecto. 4a1. El sistema muestra un mensaje de error. 5a. El sistema no puede conectar al servidor. 5a1. El sistema muestra un mensaje indicando que no se ha podido crear la cuenta.	

Caso de uso: Modificar datos

Id	CU-03	**Caso de uso**	Modificar datos de usuario
Casos de uso relacionados			
Actores		Usuario registrado	
Objetivo		Permite al usuario modificar sus datos personales.	
Precondiciones		Debe existir el usuario	
Postcondiciones		Se modifican los datos personales del usuario y la aplicación redirigirá al usuario a su pantalla inicial de la aplicación.	
Escenario básico		1. El sistema muestra la pantalla de menú de usuario. 2. El usuario pulsa el botón “Modify User Data”. 3. El sistema muestra la pantalla con el formulario y los datos existentes. 4. El usuario introduce nuevos datos o modifica los ya existentes y pulsa “Save”. 5. El sistema modifica los datos y muestra al usuario un mensaje por pantalla.	
Escenario alternativo		2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 4a. El usuario introduce algún dato incorrecto. 4a1. El sistema muestra un mensaje de error. 4b. El usuario pulsa el botón “Reset”. 4b1. El sistema resetea los campos con los valores iniciales. 5a. El sistema no puede conectar al servidor. 5a1. El sistema muestra un mensaje indicando que no se ha podido realizar la operación.	

Caso de uso: Administrar mascotas

Id	CU-04	**Caso de uso**	Administrar mascotas
Casos de uso relacionados			
Actores		Usuario registrado	
Objetivo		Permite al usuario gestionar las diferentes mascotas.	
Precondiciones			
Postcondiciones			
Escenario básico		1. El sistema muestra la pantalla de menú de usuario. 2. El usuario pulsa el botón “Pet Manager”. 3. El sistema muestra la pantalla con un listado de las mascotas dadas de alta así como las	

	diferentes opciones.
Escenario alternativo	2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 3a. No existen mascotas. 3a1. El sistema muestra un listado vacío de mascotas así como las diferentes opciones.

Caso de uso: Alta de mascotas

Id	CU-05	**Caso de uso**	Alta de mascotas
Casos de uso relacionados			
Actores		Usuario registrado	
Objetivo		Permite al usuario dar de alta una mascota.	
Precondiciones			
Postcondiciones		Se crea el registro de la mascota y la aplicación redirigirá al usuario a la pantalla de gestión de mascotas.	
Escenario básico		1. El sistema muestra la pantalla de administración de mascotas. 2. El usuario pulsa el botón “Add Pet”. 3. El sistema muestra la pantalla “Add Pet” con el formulario. 4. El usuario introduce los diferentes datos referentes a la mascota y pulsa “Add Pet”. 5. El sistema crea la mascota y redirige al usuario a la página “Pet Manager” mostrando la nueva mascota en el listado.	
Escenario alternativo		2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 4a. El usuario no introduce ningún dato. 4a1. El sistema muestra un mensaje indicando que no se ha introducido ningún valor. 4b. El usuario pulsa “Cancel”. 4b1. El sistema vuelve a la pantalla de gestión de mascotas. 5a. El sistema detecta que ya existe una mascota con los mismos valores que la mascota introducida. 5a1. El sistema muestra un mensaje indicando que ya existe la mascota. 5b. El sistema no puede conectar al servidor. 5b1. El sistema muestra un mensaje indicando que no se ha podido realizar la operación.	

Caso de uso: Modificar datos de mascotas

Id	CU-06	**Caso de uso**	Modificar datos de mascotas
Casos de uso relacionados			
Actores		Usuario registrado	
Objetivo		Permite al usuario modificar los datos de una mascota.	
Precondiciones		Debe existir al menos una mascota registrada.	
Postcondiciones		Se modifican los datos pertenecientes a la mascota y la aplicación redirigirá al usuario a la pantalla de gestión de mascotas.	
Escenario básico		1. El sistema muestra la pantalla de administración de mascotas con el listado de mascotas dadas de alta. 2. El usuario selecciona una mascota pulsando sobre el elemento del listado. 3. El sistema muestra la pantalla con los datos de la mascota. 4. El usuario modifica alguno de los datos y pulsa “Save”. 5. El sistema guarda los datos de la mascota y redirige al usuario a la página “Pet Manager” mostrando la nueva mascota en el listado.	
Escenario alternativo		2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 4a. No se modifica ningún dato. 4a1. El sistema no registra cambios. 4b. El usuario pulsa “Cancel”. 4b1. El sistema vuelve a la pantalla de gestión de mascotas. 5a. El sistema no puede conectar al servidor. 5a1. El sistema muestra un mensaje indicando que no se ha podido realizar la operación.	

Caso de uso: Baja de mascotas

Id	CU-07	**Caso de uso**	Baja de mascotas
Casos de uso relacionados			
Actores		Usuario registrado	
Objetivo		Permite al usuario eliminar una mascota.	
Precondiciones		Debe existir al menos una mascota registrada.	
Postcondiciones			
Escenario básico		1. El sistema muestra la pantalla de	

	administración de mascotas con el listado de mascotas dadas de alta. 2. El usuario pulsa sobre una mascota del listado. 3. El sistema muestra la pantalla con los datos de la mascota. 4. El usuario pulsa sobre el botón “Delete Pet”. 5. El sistema solicita confirmación sobre el borrado de la mascota. 6. El usuario confirma el borrado de la mascota. 7. El sistema borra la mascota y en caso de existir mensajes relacionados con la mascota, los elimina y redirige al usuario a la pantalla “Pet Manager” mostrando el listado de mascotas dadas de alta y las diferentes opciones.
Escenario alternativo	2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 6a. El usuario cancela el borrado de la mascota. 6a1. El sistema vuelve a la pantalla con los datos de la mascota. 7a. El sistema no puede conectar al servidor. 7a1. El sistema muestra un mensaje indicando que no se ha podido realizar la operación.

Caso de uso: Logout de la aplicación

Id	CU-08	**Caso de uso**	Logout de la aplicación
Casos de uso relacionados			
Actores		Usuario registrado	
Objetivo		Permite al usuario realizar un logout de la aplicación con su id de usuario.	
Precondiciones		El usuario está logueado.	
Postcondiciones		El usuario no está logueado.	
Escenario básico		1. El sistema muestra la pantalla del perfil de usuario. 2. El usuario selecciona la opción “Logout”. 3. El sistema muestra una pantalla de confirmación. 4. El usuario confirma el logout de la aplicación. 5. El sistema realiza el logout del usuario y lo redirige a la página principal de la aplicación.	
Escenario alternativo		2a. El usuario elige salir de la aplicación.	

	2a1. Finaliza el caso de uso 4a. El usuario cancela el logout de la aplicación. 4a1. El sistema vuelve a la pantalla principal del usuario. 5a. El sistema no puede conectar al servidor. 5a1. El sistema muestra un mensaje indicando que no se ha podido realizar la operación.

Caso de uso: Envío de mascota localizada

Id	CU-09	**Caso de uso**	Envío de mascota localizada
Casos de uso relacionados			
Actores		Usuario, Usuario registrado	
Objetivo		Permite al usuario enviar información acerca de una mascota localizada.	
Precondiciones			
Postcondiciones		Se registra la alerta y se generan los posibles mensajes para las mascotas coincidentes.	
Escenario básico		1. El sistema muestra la pantalla inicial (tanto para el usuario registrado como no registrado). 2. El usuario selecciona la opción “Notify Finding”. 3. El sistema muestra un formulario con los diferentes datos. 4. El usuario completa el formulario, añade una foto y de manera opcional, registra una localización. 5. El sistema registra la localización de una mascota, comprueba las posibles coincidencias y genera los diferentes mensajes de alerta mostrando un mensaje al usuario.	
Escenario alternativo		2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 5a. El sistema no puede conectar al servidor. 5a1. El sistema muestra un mensaje indicando que no se ha podido realizar la operación.	

Caso de uso: Gestión de mensajes de usuario

Id	CU-10	**Caso de uso**	Gestión de mensajes de usuario

Casos de uso relacionados	
Actores	Usuario registrado
Objetivo	Permite al usuario gestionar mensajes relacionados con mascotas perdidas
Precondiciones	Existe al menos una mascota marcada como perdida.
Postcondiciones	
Escenario básico	1. El sistema muestra la pantalla del perfil de usuario. 2. El usuario selecciona la opción “Manage Messages”. 3. El sistema muestra un formulario con la lista de diferentes notificaciones. En el caso de no existir notificaciones, muestra un mensaje.
Escenario alternativo	2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso

Caso de uso: Identificar mascota como perdida

Id	CU-11	**Caso de uso**	Identificar mascota como perdida
Casos de uso relacionados			
Actores		Usuario registrado	
Objetivo		El usuario marca una mascota como perdida.	
Precondiciones		Existe al menos una mascota dada de alta.	
Postcondiciones		La mascota pasa al estado perdida.	
Escenario básico		1. El sistema muestra la pantalla de administración de mascotas con el listado de mascotas dadas de alta. 2. El usuario pulsa sobre una mascota del listado. 3. El sistema muestra la pantalla con los datos de la mascota. 4. El usuario marca la opción “Lost”. 5. El sistema solicita confirmación sobre el estado de la mascota. 6. El usuario confirma el estado de la mascota. 7. El sistema marca la mascota como perdida.	
Escenario alternativo		2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 6a. El usuario cancela el cambio de estado de la mascota. 6a1. El sistema vuelve a la pantalla con los datos de la mascota. 6b. El sistema no puede conectar al servidor. 6b1. El sistema muestra un mensaje	

	indicando que no se ha podido realizar la operación.

Caso de uso: Desmarcar mascota como perdida

Id	CU-12	**Caso de uso**	Desmarcar mascota como perdida
Casos de uso relacionados			
Actores		Usuario registrado.	
Objetivo		El usuario desmarca una mascota como perdida.	
Precondiciones		Existe al menos una mascota dada de alta y marcada como perdida.	
Postcondiciones		La mascota no está registrada como perdida.	
Escenario básico		1. El sistema muestra la pantalla de administración de mascotas con el listado de mascotas dadas de alta. 2. El usuario pulsa sobre una mascota del listado. 3. El sistema muestra la pantalla con los datos de la mascota. 4. El usuario marca la opción “Lost”. 5. El sistema solicita confirmación sobre el estado de la mascota. 6. El usuario confirma el estado de la mascota. El sistema desmarca la mascota como perdida y en caso de existir notificaciones coincidentes con la mascota perdida elimina las posibles notificaciones.	
Escenario alternativo		2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 6a. El usuario cancela el cambio de estado de la mascota. 6a1. El sistema vuelve a la pantalla con los datos de la mascota. 6b. El sistema no puede conectar al servidor. 6b1. El sistema muestra un mensaje indicando que no se ha podido realizar la operación.	

Caso de uso: Identificar mascota como encontrada

Id	CU-13	**Caso de uso**	Identificar mascota como encontrada
Casos de uso relacionados			
Actores		Usuario registrado	
Objetivo		El sistema permite al usuario validar una mascota perdida como localizada.	
Precondiciones		Existe la mascota marcada como perdida y un mensaje de localización relacionado a la mascota.	
Postcondiciones		La mascota deja de estar registrada como perdida.	
Escenario básico		1. El sistema muestra la pantalla de administración de notificaciones. 2. El usuario pulsa sobre una de las notificaciones del listado. 3. El sistema muestra la pantalla con los datos de la notificación. 4. El usuario marca la opción “Confirm”. 5. El sistema solicita confirmación sobre la notificación. 6. El usuario confirma el estado de la notificación. El sistema desmarca la mascota como perdida y en caso de existir notificaciones coincidentes con la mascota perdida elimina las posibles notificaciones.	
Escenario alternativo		2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 6a. El usuario cancela la localización de la mascota. 6a1. El sistema vuelve a la pantalla con los datos de la notificación. 6b. El sistema no puede conectar al servidor. 6b1. El sistema muestra un mensaje indicando que no se ha podido realizar la operación.	

Caso de uso: Anular localización de mascota

Id	CU-14	**Caso de uso**	Anular localización de mascota
Casos de uso relacionados			
Actores		Usuario registrado	
Objetivo		El sistema permite al usuario anular un aviso de mascota encontrada debido a un falso positivo.	
Precondiciones		Existe la mascota marcada como perdida y un mensaje de localización relacionado a la mascota.	
Postcondiciones		El mensaje deja de estar asociado a la mascota.	

Escenario básico	1. El sistema muestra la pantalla de administración de notificaciones. 2. El usuario pulsa sobre una de las notificaciones del listado. 3. El sistema muestra la pantalla con los datos de la notificación. 4. El usuario marca la opción “Deny”. 5. El sistema solicita confirmación sobre la notificación. 6. El usuario confirma la notificación. El sistema desmarca la mascota como perdida y en caso de existir notificaciones coincidentes con la mascota perdida elimina las posibles notificaciones.
Escenario alternativo	2a. El usuario elige salir de la aplicación. 2a1. Finaliza el caso de uso 6a. El usuario cancela la localización de la mascota. 6a1. El sistema vuelve a la pantalla con los datos de la notificación. 6b. El sistema no puede conectar al servidor. 6b1. El sistema muestra un mensaje indicando que no se ha podido realizar la operación.

Caso de uso: baja de usuario

Id	CU-15	**Caso de uso**	Baja de usuario
Casos de uso relacionados			
Actores		Usuario registrado	
Objetivo		Permite al usuario eliminar sus datos de registro de la aplicación.	
Precondiciones		Existe el usuario como registrado.	
Postcondiciones		El usuario deja de estar registrado en la aplicación.	
Escenario básico		1. El sistema muestra la pantalla de menú de usuario. 2. El usuario pulsa el botón “Modify User Data”. 3. El sistema muestra la pantalla con el formulario y los datos existentes. 4. El usuario pulsa el botón “Delete User”. 5. El sistema solicita la confirmación del borrado del usuario. 6. El usuario confirma el borrado. 7. El sistema elimina el usuario como usuario registrado y todos los registros relacionados.	
Escenario alternativo		2a. El usuario elige salir de la aplicación.	

[redacted]	2a1. Finaliza el caso de uso 6a. El usuario cancela el borrado de usuario. 6a1. El sistema vuelve a la pantalla con los datos de usuario. 7a. El sistema no puede conectar al servidor. 7a1. El sistema muestra un mensaje indicando que no se ha podido realizar la operación.

3.4. Modelo conceptual

El modelo conceptual está orientado a la descripción de estructuras de datos y restricciones de integridad, representando los elementos que intervienen en el problema y sus relaciones. Representaremos el modelo conceptual de la aplicación mediante diagramas entidad-relación.

3.4.1. Modelo entidad-relación UML

En el siguiente diagrama se puede ver el modelo entidad-relación previo a la normalización.

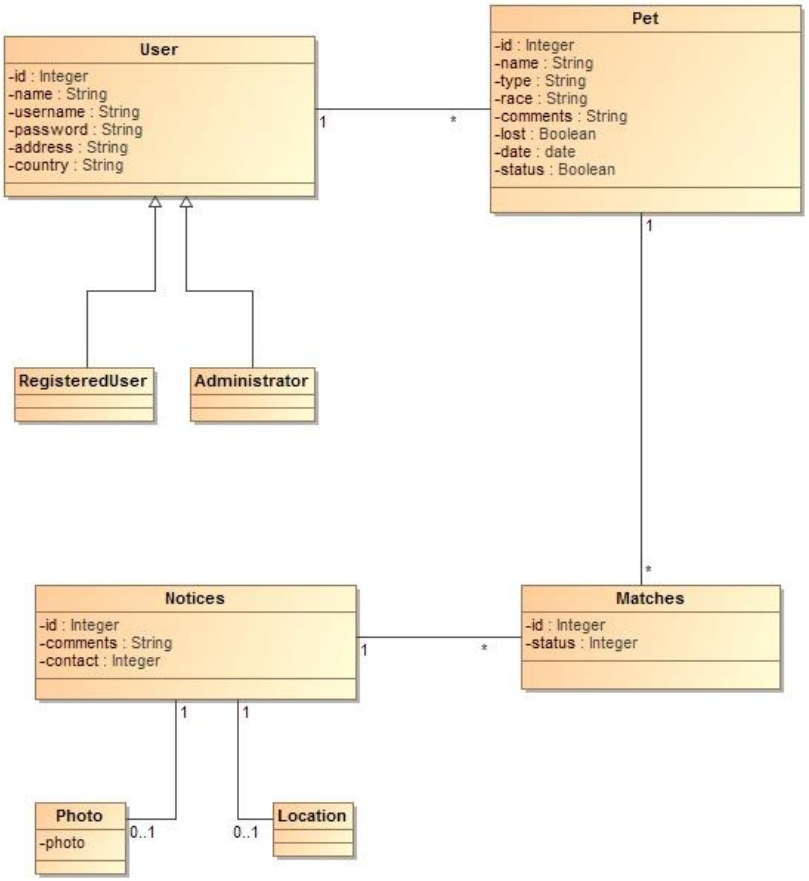

Ilustración 15 - Modelo de datos UML no normalizado

3.4.1. Modelo de datos normalizado

Una vez normalizado el modelo entidad relación, el modelo de datos quedaría de la siguiente forma:

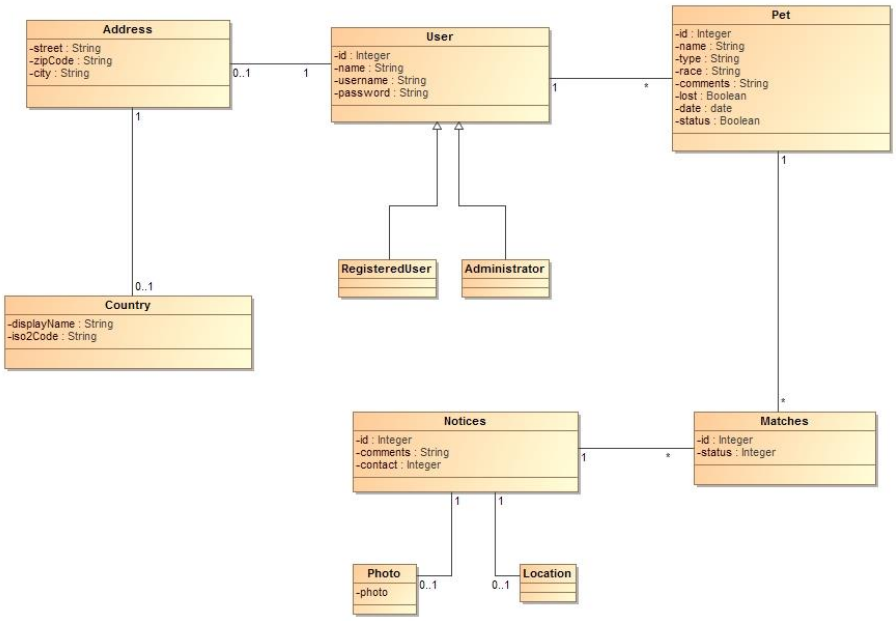

Ilustración 16 - Modelo de datos UML normalizado

En la siguiente ilustración se puede ver el modelo detallado:

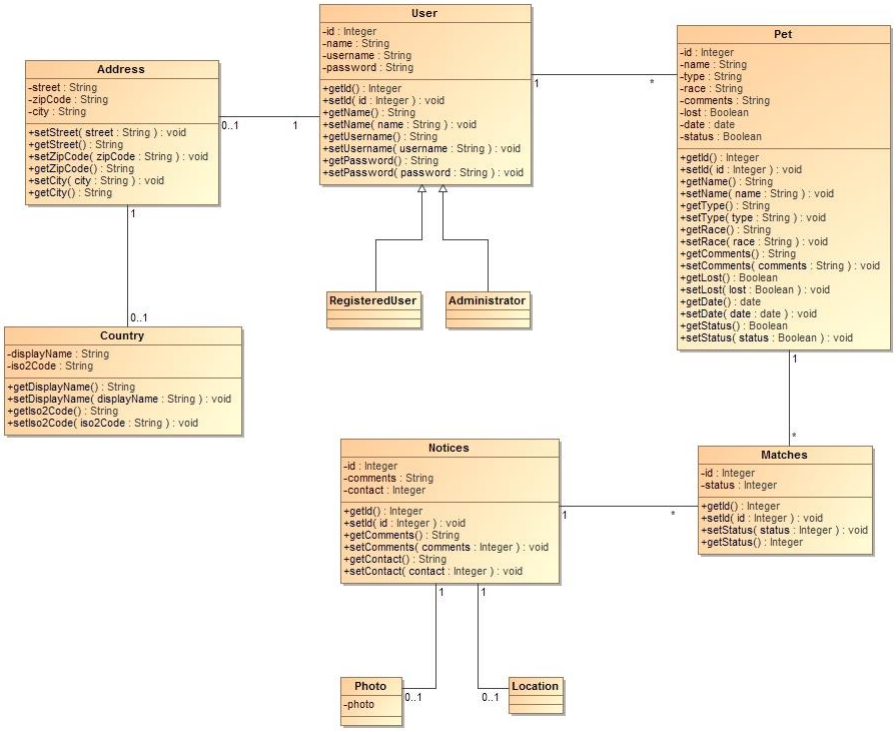

Ilustración 17 - Modelo de datos UML normalizado detallado

4. DISEÑO

4.1. Arquitectura de la aplicación

La aplicación web se construirá a partir del framework Spring MVC.

4.1.1. Patrón Modelo-Vista-Controlador

MVC es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Se fundamenta en la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores, o lo que es lo mismo, Model, Views and Controllers (MVC), en inglés.

- Capa modelo: es la capa que representa la información específica del dominio de la aplicación, en la que se trabaja con los datos. Contiene mecanismos para acceder a la información y para actualizar su estado mediante una subcapa.
- Capa vista: contiene el código que produce la visualización de las interfaces de usuario. También llamada capa de presentación, permite la interacción con el usuario. En la vista se trabaja con los datos, sin embargo, no se realiza un acceso directo a ellos. Las vistas requieren los datos a los modelos y se encargan de generar la salida.
- Capa controlador: contiene el código necesario para responder y procesar las diferentes acciones que solicita la aplicación, como visualizar elementos, realizar búsquedas… Se trata de la capa de enlace entre la capa controlador y la capa modelo. No manipula directamente los datos ni muestra ningún tipo de salida, sino que sirve de enlace entre ambas capas para implementar las necesidades del desarrollo.

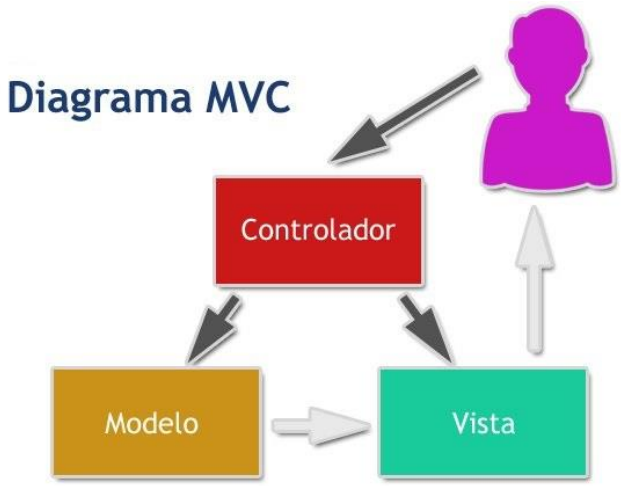

Ilustración 18 - Diagramas MVC

Como se puede ver, los controladores, con su lógica de negocio, hacen de puente entre los modelos y las vistas. Pero además en algunos casos los modelos pueden enviar datos a las vistas. La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

Existe un conjunto de reglas que se siguen en el software para reaccionar ante distintas situaciones conocidas como “lógica de negocio”. En una aplicación el usuario se comunica con el sistema por medio de una interfaz, pero cuando acciona esa interfaz para realizar acciones con el programa, se ejecutan una serie de procesos que se conocen como la lógica del negocio. Aparte de marcar un comportamiento cuando ocurren cosas dentro de un software, también tiene normas sobre lo que se puede hacer y lo que no se puede hacer. Eso también se conoce como reglas del negocio. En el MVC la lógica del negocio queda del lado de los modelos. Ellos son los que deben saber cómo operar en diversas situaciones y las cosas que pueden permitir que ocurran en el proceso de ejecución de una aplicación.

Existe otro concepto que se usa en la terminología del MVC que es la lógica de aplicación, algo que pertenece a los controladores. Todo el conjunto de acciones que se realizan invocando métodos de los modelos y mandando datos a las vistas forman parte de la lógica de la aplicación.

4.1.2. Diseño de la arquitectura del sistema

La arquitectura del sistema es una arquitectura cliente-servidor, donde la aplicación se divide en un componente servidor, encargado de ofrecer los servicios y un conjunto de clientes que usan dichos servicios.

El sistema está compuesto por un servidor Microsoft Windows con la aplicación servidor instalada soportada por Java, la cual incluye todas las dependencias necesarias mediante la utilización de Maven. Una de las dependencias es un servidor Apache, el cual, corre para dar servicio a las diferentes peticiones por parte del cliente. En el componente servidor se proporcionan los servicios de autenticación y las diferentes gestiones relacionadas con las mascotas solicitadas por los clientes.

Esta aplicación conecta con un servidor de base de datos MySQL Server, en cuya base de datos persiste la información utilizando Hibernate para ello.

En cuanto al software cliente, existe una aplicación desarrollada en Java bajo la API Android para permitir su uso en dispositivos con este sistema operativo.

El esquema puede verse en la siguiente ilustración:

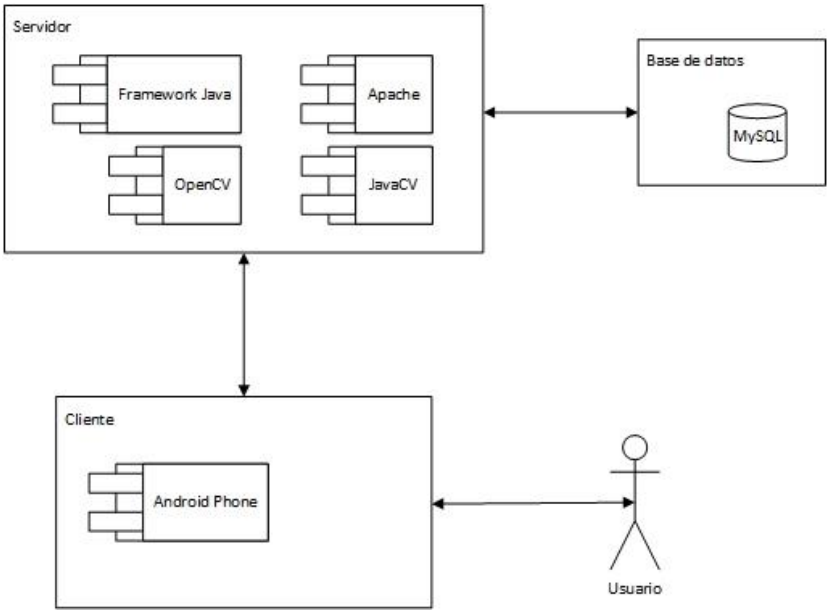

Ilustración 19 - Infraestructura del sistema

La utilización de este modelo permite la distribución de sus componentes de forma natural y una mejor escalabilidad, lo que permite que se puedan incorporar nuevos recursos de forma transparente a la aplicación y al usuario, como pueden ser servidores en el caso de necesitar una mayor potencia de proceso.

En este esquema, los servicios ofrecidos por los servidores se llaman servicios web (o webservices) y estos pueden ser:

- Servicios web basados en SOAP (Simple Object Access Protocol), un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Se pueden implementar en varios mensajes y siguen las especificaciones SOAP, WSDL y UDDI
- Servicios web basados en REST (Representational State Transfer o Transferencia de Estado Representacional): se trata de cualquier interfaz web simple que utiliza XML y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web SOAP. REST se basa en la arquitectura orientada a recursos (ROA) que está basada en acciones sobre dichos recursos para permitir la interacción entre el cliente y el servidor, mediante métodos get, post, put y delete ya incluidos en las llamadas HTTP. Mediante REST, no se requiere que los mensajes sea XML, lo que hace a REST más eficiente que los basados en SOAP, sobre todo para las aplicaciones para dispositivos móviles donde las respuestas al usuario deben ser siempre lo más rápidas posibles.

Para este trabajo, se utilizarán servicios web basados en REST, y para el formato de intercambio datos se utilizará un formato ligero como JSON.

4.1.3. Diseño de la base de datos

Para la capa de persistencia se parte del modelo de datos visto anteriormente. El esquema de base de datos obtenido es el siguiente:

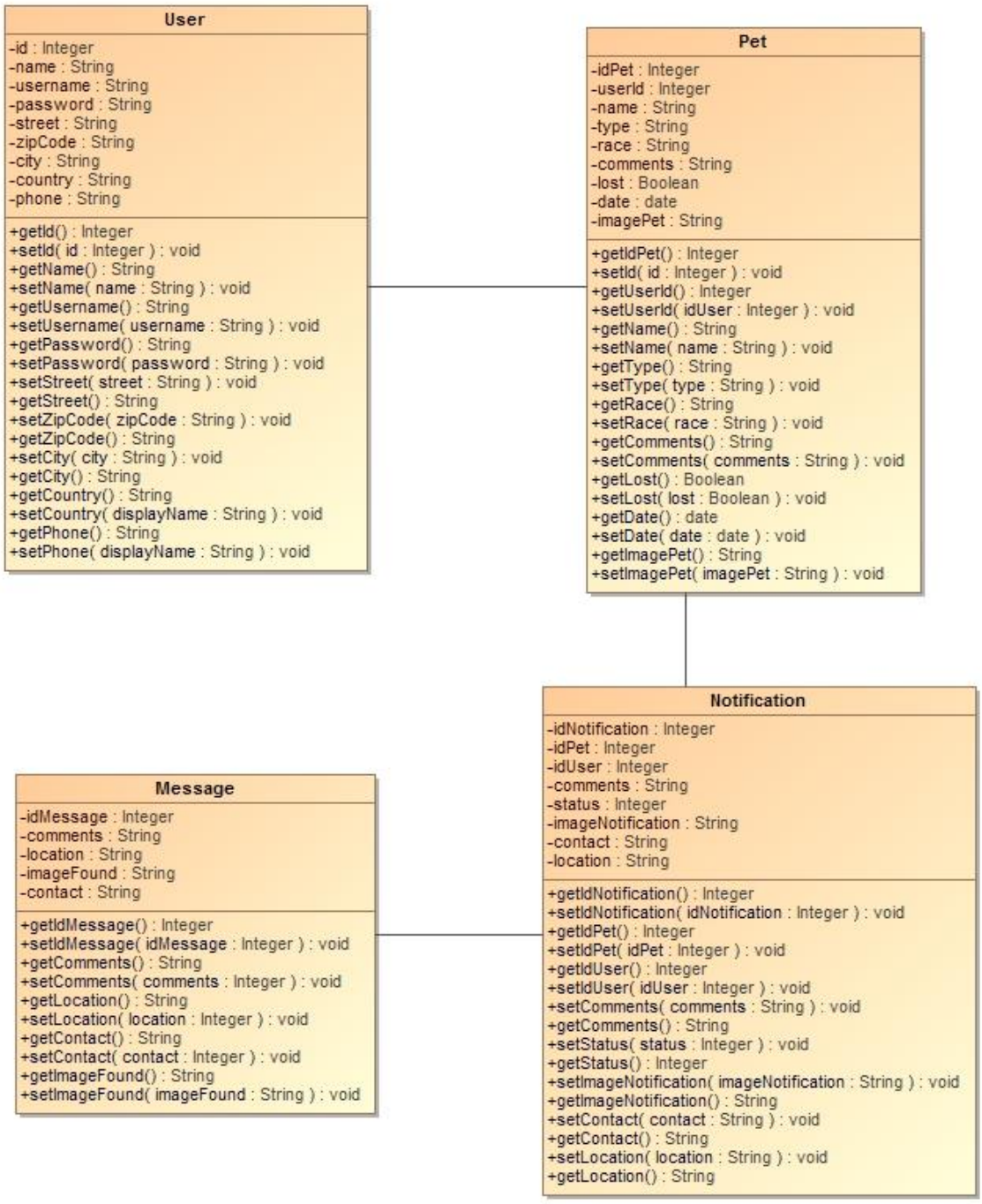

Ilustración 20 - Modelo de Base de Datos de la aplicación

5. IMPLEMENTACIÓN

Finalizadas las tareas de análisis y diseño se aplican las diferentes directrices de fases anteriores para dar forma al proyecto de software. El código se ha desarrollado con la ayuda del IDE Eclipse y ADT (Android Development Tools).

5.1. Estructura y componentes de la aplicación cliente

Al crear una aplicación Android mediante Eclipse, se crea una estructura de carpetas por defecto, las cuales serán ampliadas para incluir las clases necesarias de la aplicación. La estructura generada puede verse en la siguiente ilustración:

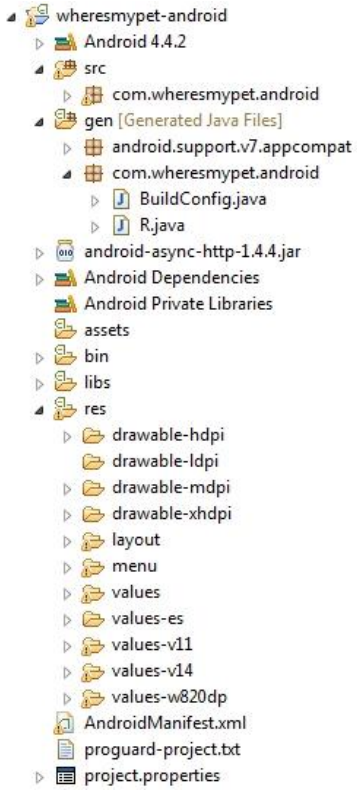

Ilustración 21 - Estructura de componentes de la aplicación cliente

En la estructura anterior, se pueden ver las siguientes carpetas importantes para el proyecto:

- /src: en la que se incluye el código fuente de la aplicación y todas las clases auxiliares.

- /gen: contiene los diferentes elementos generados automáticamente por Eclipse al compilar el proyecto. Uno de los ficheros, "R.java" contiene una serie de constantes con los identificadores de todos los recursos que están en la carpeta /res y permite que sean accesibles desde el código.

- /assets: contiene el resto de los ficheros necesarios para la aplicación como pueden ser los ficheros de datos, de configuración, etc. No se genera un identificador para acceder a ellos, sino que se accede directamente con la ruta.

- /bin: en esta carpeta se generan los diferentes elementos compilados, los ficheros binarios, incluido el ejecutable de la aplicación para ser posteriormente instalado en el dispositivo móvil (fichero .apk).

- /libs: contiene todas las librerías necesarias para la aplicación. Generalmente, estas se almacenan en archivos con formato .jar.

- /res: ruta en la que se incluyen los diferentes recursos utilizados por la aplicación como imágenes, cadenas de texto, etc así como las diferentes interfaces gráficas utilizadas por la aplicación dentro de la carpeta layout.

- AndroidManifest.xml: es quizá, el fichero más importante de cualquier aplicación Android. En él se definen mediante XML los aspectos más importantes de la aplicación como la identificación (nombre, versión, icono, versiones soportadas…), los componentes incluidos (pantallas, cadenas de texto…), los permisos necesarios para la ejecución de la aplicación y las diferentes librerías utilizadas.

La estructura de clases de la aplicación, como se ha comentado anteriormente, dentro de la carpeta /src, dónde se incluyen las clases Activity para representar las pantallas, queda de la siguiente forma:

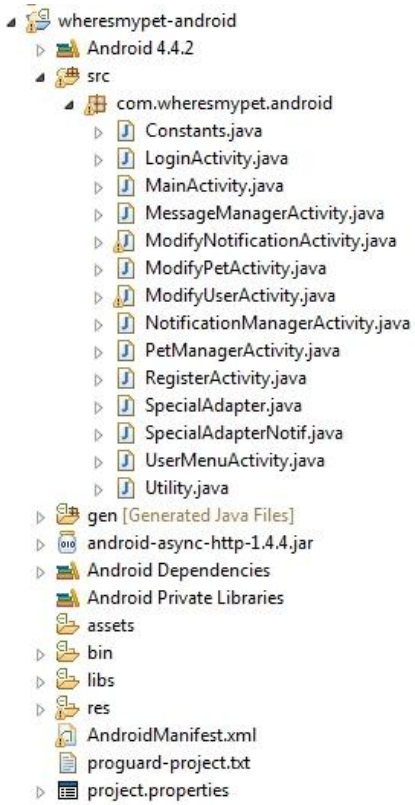

Ilustración 22 - Contenido de la carpeta /src

En la estructura se pueden observar las siguientes clases:

- Constants: clase que almacena información común para la aplicación. Uno de los datos que se almacena es la ruta del servidor al que la aplicación debe conectar, facilitando de esta manera la posibilidad de cambiar la aplicación de ruta de ejecución. En el caso de estudio, se ha configurado una ruta para una máquina virtual.

```
package com.wheresmypet.android;

// Parameters according to your DB
public class Constants {
    // To connect into AVD it must be configured as 10.0.2.2
    // Server address
    public static String server = "http://10.0.2.2:8080/wheresmypet";
}
```

Ilustración 23 - Clase Constants

- LoginActivity: Pantalla de login de usuario.
- MainActivity: Pantalla inicial de la aplicación.
- MessageManagerActivity: Pantalla de gestión de mensajes para mascotas encontradas.
- ModifyNotificationActivity: Pantalla de gestión de notificaciones de usuario.
- ModifyPetActivity: Pantalla de gestión de mascotas.
- ModifyUserActivity: Pantalla de gestión de datos de usuario.
- NotificationManagerActivity: Pantalla que muestra un listado con todas las notificaciones del usuario.
- PetManagerActivity: Pantalla que muestra un listado con todas las mascotas del usuario.
- RegisterActivity: Pantalla para el registro de usuarios.
- SpecialAdapter y SpecialAdapterNotif: Adaptadores utilizados para identificar mediante colores valores en las listas de mascotas y notificaciones respectivamente.
- UserMenuActivity: Pantalla de menú de usuario.
- Utility: clase que incluye diferentes utilidades usadas por cualquier parte de la aplicación como conversión de imágenes, comprobación de campos, etc.

5.2. Estructura y componentes de la aplicación servidor

Mediante Eclipse se crea un proyecto tipo Maven que, como se ha comentado anteriormente, facilita entre otras cosas la integración de las diferentes librerías durante el desarrollo de la aplicación. Un proyecto Maven crea una estructura de carpetas por defecto, las cuales serán ampliadas para incluir las clases necesarias de la aplicación. La estructura generada puede verse en la siguiente ilustración:

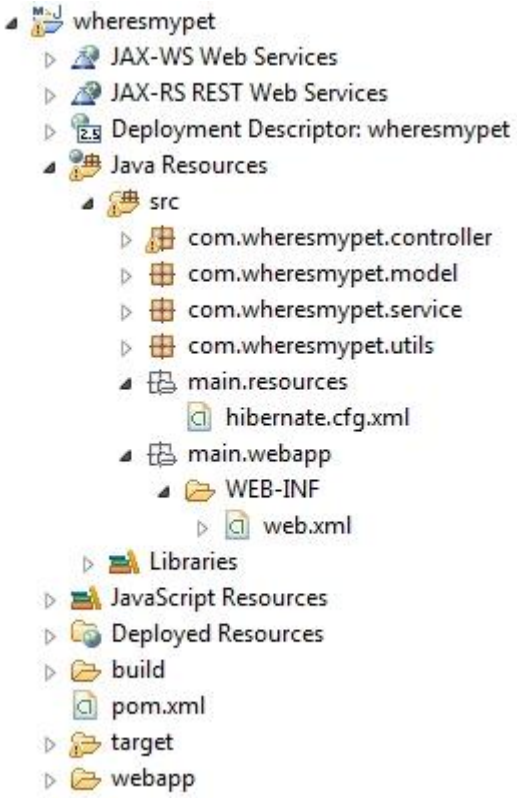

Ilustración 24 - Estructura de proyecto Maven servidor

En la estructura se pueden observar diferentes elementos:

- /src: en la que se incluye el código fuente de la aplicación y todas las clases auxiliares.

- hibernate.cfg.xml: es un fichero utilizado para configurar hibernate (persistencia de datos). La información que este contiene se refiere a las propiedades de configuración y las clases que se quieren mapear.

- pom.xml: Maven utiliza un fichero pom.xml (Project Object Model) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Por lo tanto, es el fichero más importante para un proyecto Maven ya que define cómo se interrelacionan los diferentes componentes de la aplicación.

Se han creado diferentes packages para agrupar las diferentes clases:

- com.wheresmypet.controller: en el que se guardan todas las clases que tienen la función de controlador dentro de la aplicación. Cabe destacar la clase “Constants”, la cual almacena variables relativas a la ruta utilizada para almacenar las fotografías utilizadas por el sistema. De esta forma se permite una fácil modificación en caso necesario.

- com.wheresmypet.model: en el que se almacena el modelo de datos de la aplicación.

- com.wheresmypet.service: clases utilizadas para la persistencia y que ofrecen servicio al modelo de datos.

- com.wheresmypet.utils: en el que se almacenan clases auxiliares utilizadas por el conjunto de la aplicación.

En la siguiente ilustración se puede ver cómo quedan los diferentes packages:

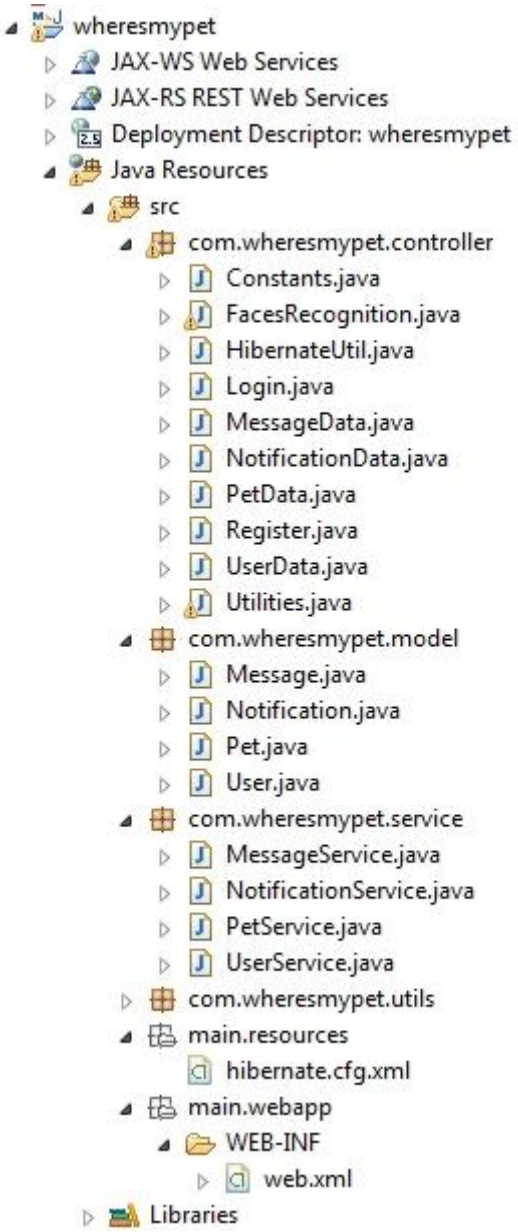

Ilustración 25 - Packages aplicación servidor

6. TECNOLOGÍAS EMPLEADAS

Este apartado se resume las tecnologías utilizadas para la aplicación:

- Java: tanto como para la parte cliente como para el entorno servidor, se utiliza la plataforma de desarrollo Java.

- Apache HTTPComponents: conjunto de herramientas de componentes Java de bajo nivel, enfocados para trabajar con el protocolo HTTP. Aunque el paquete de java.net proporciona la funcionalidad básica para los recursos que tienen acceso vía *HTTP*, no proporciona la flexibilidad o la funcionalidad completa necesitada por muchos usos. El componente HttpComponents de Apache intenta llenar este vacío proporcionando un paquete eficiente, actualizado, y abundantes características que pone el lado del cliente en ejecución de los estándares más recientes y de las recomendaciones del *HTTP*.

- JSON: formato ligero para el intercambio de datos para las respuestas de los servicios REST. La aplicación almacena las respuestas en este formato para convertirlas en clases Java (POJOs) que representen el modelo de dominio. En este trabajo, las respuestas del servidor web seguirán el formato JSON.

- MySQL, gestor de bases de datos utilizado para crear una base de datos en el servidor y almacenar los datos que deben ser persistentes.

6.1. Herramientas de software y recursos hardware

6.1.1. Herramientas de software

- ADT Bundle para Windows: se utilizará el paquete completo ADT Bundle (Android Developer Tools), el cual incluye una versión de eclipse, los plugins de ADT, diferentes herramientas y la plataforma de desarrollo Android así como una serie de emuladores para simular dispositivos móviles en el PC mediante la ejecución de AVD (Android Virtual Device). En los diferentes anexos, se describe tanto la instalación del paquete como detalles acerca de la configuración del entorno de desarrollo.
- MySQL: en el servidor se almacenan los datos persistentes necesarios para la aplicación, generados desde los diferentes clientes así como con tareas propias del servidor. Se ha optado por utilizar MySQL versión 5.6.21.1 para el desarrollo de la aplicación.
- OpenCV: se trata de una biblioteca de libre distribución y bajo licencia BSD para la visión artificial originalmente desarrollada por Intel. Se trata de una biblioteca desarrollada en C++, por lo que se hace necesario su uso junto a JavaCV. Para el desarrollo de la aplicación, se utiliza la versión 2.4.9
- JavaCV: Esta biblioteca se utiliza como interfaz OpenCV para Java. Para el desarrollo de la aplicación, se utiliza la versión 0.9.
- Maven: es una herramienta de software para la gestión y construcción de proyectos Java similar en funcionalidad a Apache Ant, pero tiene un modelo de configuración de construcción más simple, basado en un formato XML. Maven

utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores. Todo esto, lo convierte en una herramienta que facilita enormemente las tareas de desarrollo y posterior mantenimiento de la aplicación.

6.1.2. Recursos Hardware

La aplicación está claramente dividida en dos partes. Una parte cliente orientada a dispositivos móviles Android y otra parte servidor en el que se gestionan las diferentes peticiones de los clientes y se almacenan los datos persistentes de la aplicación.

Para la parte cliente, como se ha comentado anteriormente, la aplicación se ha desarrollado para el sistema operativo Android. Aunque se ha desarrollado para ser compatible con dispositivos con versión de sistema operativo superior a la 2.2, se ha optimizado para la versión de Android 4.4. Se han utilizado dos dispositivos para el desarrollo del trabajo:

- Samsung Galaxy S4: versión de Android 4.4.2
- Samsung Galaxy Tab S: versión de Android 4.4.2

Para la parte servidor se ha optado por un entorno con sistema operativo Windows. Para la realización del trabajo, se han utilizado dos infraestructuras distintas. Por una parte, una máquina virtual con sistema operativo Windows 7 sobre la cual se instalan todas las herramientas necesarias vistas anteriormente. Por otro lado, esa misma infraestructura se lleva a la nube mediante un VPS (Virtual Private Server). En los diferentes anexos, se describe tanto la instalación como detalles acerca de la configuración del entorno.

7. PRUEBAS

Los errores humanos dentro de la programación de los computadores son muchos y aumentan considerablemente con la complejidad del problema, siendo estos no solo errores de programación sino una implementación incorrecta de los requisitos o incluso la no implementación de los mismos. Por tanto, es necesario realizar las debidas pruebas que garanticen el correcto funcionamiento de dicho programa bajo el mayor número de situaciones posibles a las que se pueda enfrentar. Esto, se realiza en el proceso de pruebas o testing con el objetivo de detectar estos errores y solucionarlos y mejorar por tanto la calidad del producto final.

7.1. Niveles de pruebas

Existen diferentes tipos de pruebas para el software que se clasifican de la siguiente forma:

- En función del ámbito del sistema. Una aplicación, sobre todo aquellas desarrolladas en lenguajes orientados a objetos constan de diferentes elementos, estos, pueden funcionar por sí solos o bien estar interrelacionados con otros objetos del sistema, por lo que es necesaria la realización de diferentes pruebas, las cuales se clasifican en:
    - Pruebas unitarias: las cuales verifican la funcionalidad de cada uno de los componentes de la aplicación independientemente del resto.
    - Pruebas de integración: en las cuales se verifica la interacción de los diferentes componentes que forman parte de la aplicación.
    - Pruebas de sistema: las cuales verifican el funcionamiento del conjunto del sistema con todos los componentes integrados.
- En función del objetivo de la prueba. Se puede querer probar un requerimiento, la respuesta de la aplicación en cuanto a tiempo, consumo de recursos, usabilidad, etc. Este conjunto de pruebas se puede clasificar de la siguiente forma:
    - Pruebas de aceptación: en las cuales se comprueban el funcionamiento de los diferentes requisitos de funcionalidad planteados en apartados anteriores. Este tipo de pruebas implica definir todos los escenarios posibles para todas las situaciones que pueden producirse. Este tipo de pruebas suele realizarse por parte del usuario final, por lo que también son conocidas como “Pruebas de Aceptación de Usuario”.
    - Pruebas de instalación: una vez validadas las diferentes pruebas de aceptación, es necesario comprobar el funcionamiento del software bajo diferentes entornos.
    - Pruebas Alpha: aquellas ejecutadas por los usuarios desarrolladores antes de la publicación del producto.
    - Pruebas Beta: ejecutadas por un conjunto de usuarios predefinidos.
    - Pruebas de fiabilidad: las cuales verifican que el software cumpla con la especificación realizada.
    - Pruebas de regresión: las cuales consisten en probar que los test realizados previamente a la aplicación son aplicables una vez introducidos

cambios en la misma.
- Pruebas de rendimiento: en las que se comprueba el rendimiento de los diferentes sistemas sobre los que se ejecuta la aplicación.
- Pruebas de seguridad: en las que se comprueba la protección de la aplicación frente a ataques externos o bien cómo se protege la confidencialidad, integridad y disponibilidad del sistema y de los datos que este maneja.
- Pruebas de estrés: en las que se chequea el comportamiento de la aplicación y los diferentes sistemas bajo situaciones límite como pueden ser los estados de la red, número de usuarios, etc.
- Pruebas de recuperación: analizando el comportamiento del sistema tras una situación de desastre.
- Pruebas de usabilidad: las cuales analizan la facilidad con la que los usuarios finales aprenden a utilizar la aplicación.

Dependiendo de la aplicación desarrollada, es necesario llevar a cabo las pruebas aplicables a la misma.

7.2. Pruebas para la aplicación

Los tipos de pruebas realizados para la aplicación han sido los siguientes:

- Pruebas unitarias: realizadas durante el desarrollo de la aplicación definido en el Anexo.
- Pruebas de sistema: se ha instalado y probado la aplicación en los siguientes dispositivos definidos en el Anexo:
  - AVD (Android Virtual Device) bajo Eclipse.
  - Samsung Galaxy S4 Android 4.4.2
  - Samsung Galaxy Tab S Android 4.4.2
- Pruebas de aceptación: comprobando el correcto cumplimiento de los requerimientos definidos inicialmente.

7.2.1. Pruebas de aceptación

En esta sección se presentarán las diferentes pruebas de aceptación realizadas sobre la aplicación y que han servido para para validar la lista de requisitos definidos anteriormente así como verificar la corrección de todas las funcionalidades del sistema.

Las diferentes pruebas realizadas se presentarán en forma de tabla. Los campos de cada tabla serán los siguientes:

- Id: identificador del tipo PA-XY, siendo XY un número correlativo comenzando en “01” (PA indicará “prueba de aceptación”).
- Descripción: descripción de la prueba realizada en la aplicación.
- Resumen: descripción resumida de la prueba realizada.
- Resultado: resultado de la prueba realizada.

Prueba de aceptación: Login de la aplicación

Id	PA-01	**Resumen**	Login en la aplicación
Descripción		Se va a comprobar que el proceso de login de la aplicación es correcto. Las pruebas realizadas han sido las siguientes: 1. Se introducen los datos correctos de un perfil usuario. 2. Se introduce un identificador de usuario inexistente. 3. Se introduce un identificador de usuario correcto pero con contraseña incorrecta. 4. Se deja alguno de los campos vacíos.	
Resultado		Los resultados de las distintas pruebas han sido las esperadas: 1. Se muestra el menú principal del usuario. 2. Sobre la pantalla principal de la aplicación, se muestra el mensaje “Incorrect email or password”. 3. Sobre la pantalla principal de la aplicación, se muestra el mensaje “Incorrect email or password”. 4. Sobre la pantalla principal de la aplicación, se muestra el mensaje “Please fill the form, don't leave any field in blank”.	

Prueba de aceptación: Registrar cuenta de usuario

Id	PA-02	**Resumen**	Registrar cuenta de usuario
Descripción		Se va a comprobar que el proceso añadir un nuevo usuario funciona de manera correcta. Las pruebas realizadas han sido las siguientes: 1. Se introducen correctamente todos los datos en el formulario y se confirma el registro. 2. Se deja algún campo vacío y se confirma el registro. 3. Se introduce un usuario existente y se confirma el registro.	
Resultado		Los resultados de las distintas pruebas han sido las esperadas: 1. Sobre la pantalla de registro de usuario, se muestra el mensaje “You are successfully registered” y se pasa a la pantalla de login de	

	usuario. 2. Sobre la pantalla de registro de usuario, se muestra el mensaje “Please fill the form, don't leave any field in blank”. 3. Sobre la pantalla de registro de usuario, se muestra el mensaje “Email already registered”.

Prueba de aceptación: Modificar datos de usuario

Id	PA-03	**Resumen**	Modificar datos de usuario
Descripción		Se va a comprobar que el proceso modificar datos de usuario funciona de manera correcta. Las pruebas realizadas han sido las siguientes: 1. No se realizan cambios y se graba. 2. Se deja vacío algún campo requerido y se graba. 3. Se introduce una contraseña que no coincide con la confirmación de contraseña y se graba. 4. Se introducen de forma correcta los datos del formulario (introduciendo los campos requeridos y el resto de los campos de forma indiferente).	
Resultado		Los resultados de las distintas pruebas han sido las esperadas: 1. Sobre la pantalla de datos de usuario, se muestra el mensaje “No changes done”. 2. Sobre la pantalla de datos de usuario, se muestra el mensaje “Name can not be empty” o bien “Password can not be empty” dependiendo del campo dejado en blanco. 3. Sobre la pantalla de datos de usuario, se muestra el mensaje “Password and confirmation are not equal”. 4. Sobre la pantalla de datos de usuario, se muestra el mensaje “User successfully updated”, se actualizan los datos de usuario y se pasa al menú de usuario.	

Prueba de aceptación: Administrar mascotas

Id	PA-04	**Resumen**	Administrar mascotas
Descripción		Se va a comprobar que el proceso administrar mascotas funciona de manera correcta. Las pruebas	

	realizadas han sido las siguientes: 1. Sin existir mascotas se accede a la opción “Pet Manager”. 2. Teniendo el usuario mascotas creads, se accede a la opción “Pet Manager”.
Resultado	Los resultados de las distintas pruebas han sido las esperadas: 1. Se muestra la pantalla de “Pet Manager” sin mascotas en el listado. 2. Se muestra la pantalla de “Pet Manager” con un listado con únicamente las mascotas dadas de alta anteriormente para el usuario.

Prueba de aceptación: Alta de mascotas

Id	PA-05	**Resumen**	Alta de mascotas
Descripción		Se va a comprobar que el proceso de alta de datos de mascotas funciona de manera correcta. Desde la pantalla de “Pet Manager” se accede al alta de mascotas mediante el botón “Add Pet”. Las pruebas realizadas han sido las siguientes: 1. No se introducen todos los campos requeridos y se graba. 2. Se introducen los datos requeridos y se marca la mascota como perdida pero no se indica fecha y se graba. 3. Se introducen los datos requeridos, se marca la mascota como perdida y se indica fecha pero no se incluye foto y se graba. 4. Se introducen los datos correctos de la mascota y se graba.	
Resultado		Los resultados de las distintas pruebas han sido las esperadas: 1. Sobre la pantalla de datos de mascota, se muestra el mensaje “Please fill the form, don't leave any field in blank”. 2. Sobre la pantalla de datos de mascota, se muestra el mensaje “Please insert date when pet was lost”. 3. Sobre la pantalla de datos de mascota, se muestra el mensaje “If pet is lost, image is needed”. 4. Sobre la pantalla de gestión de mascotas, se	

	muestra el mensaje “Pet successfully created”, mostrando a su vez la lista incluyendo la nueva mascota (en caso de haber sido marcada como perdida, se muestra bajo color amarillo, en caso contrario, en blanco).

Prueba de aceptación: Modificar datos de mascotas

Id	PA-06	**Resumen**	Modificar datos de mascotas
Descripción		Se va a comprobar que el proceso modificar datos de mascota funciona de manera correcta. Las pruebas realizadas han sido las siguientes: 1. No se realizan cambios y se graba. 2. Se deja vacío algún campo requerido y se graba. 3. Se introducen los datos requeridos y se marca la mascota como perdida pero no se indica fecha y se graba. 4. Se introducen los datos requeridos, se marca la mascota como perdida y se indica fecha pero no se incluye foto y se graba. 5. Se introducen los diferentes campos de la mascota de forma correcta y se graba.	
Resumen		Los resultados de las distintas pruebas han sido las esperadas: 1. Sobre la pantalla de datos de mascota, se muestra el mensaje “No changes done”. 2. Sobre la pantalla de datos de mascota, se muestra el mensaje “Please fill the form, don't leave any field in blank”. 3. Sobre la pantalla de datos de mascota, se muestra el mensaje “Please insert date when pet was lost”. 4. Sobre la pantalla de datos de mascota, se muestra el mensaje “If pet is lost, image is needed”. 5. Sobre la pantalla de gestión de mascotas, se muestra el mensaje “Pet successfully updated”, mostrando a su vez la lista incluyendo la mascota (en caso de haber sido marcada como perdida, se muestra bajo color amarillo, en caso contrario, en blanco).	

Prueba de aceptación: Baja de mascotas

Id	PA-07	**Resumen**	Baja de mascotas
Descripción		Se va a comprobar que el proceso baja de mascotas funciona de manera correcta. Para realizar la baja de mascotas es necesario acceder a los datos de una mascota y pulsar el botón delete. Las pruebas realizadas han sido las siguientes: 1. El usuario pulsa el botón delete y no confirma el borrado. 2. El usuario pulsa el botón delete y confirma el borrado.	
Resultado		Los resultados de las distintas pruebas han sido las esperadas: 1. Se vuelve a la pantalla de datos de mascota. 2. Sobre la pantalla de gestión de mascotas, se muestra el mensaje “Pet successfully deleted” y sobre la lista de mascotas no aparece la mascota anteriormente eliminada.	

Prueba de aceptación: Logout de la aplicación

Id	PA-08	**Resumen**	Logout de la aplicación
Descripción		Se va a comprobar que el proceso logout de usuario funciona de manera correcta. Para realizar el logout es necesario pulsar el botón “Logout” del menú de usuario. Las pruebas realizadas han sido las siguientes: 1. El usuario pulsa el botón Logout y no confirma la salida del sistema. 2. El usuario pulsa el botón Logout y confirma la salida del sistema.	
Resultado		Los resultados de las distintas pruebas han sido las esperadas: 1. Se vuelve a la pantalla de menú de usuario. 2. Sobre la pantalla principal de la aplicación, se muestra el mensaje “Successfully logged out”.	

Prueba de aceptación: Envío de mascota localizada

Id	PA-09	**Resumen**	Envío de mascota localizada

Descripción	Se va a comprobar que el proceso de envío de mascotas funciona de manera correcta. Las pruebas realizadas han sido las siguientes: 1. No se introduce ningún dato y se graba. 2. No se introducen todos los datos del formulario y se graba. 3. Se introducen los datos correctamente, se incluye una foto y se graba.
Resultado	Los resultados de las distintas pruebas han sido las esperadas: 1. Sobre la pantalla “Notify Pet Found”, se muestra el mensaje “Please fill the form, don't leave any field in blank”. 2. Sobre la pantalla “Notify Pet Found”, se muestra el mensaje “Please fill the form, don't leave any field in blank”. 3. Sobre la pantalla “Notify Pet Found”, se muestra el mensaje “Message successfully created” y pasa a la pantalla principal de la aplicación. Si el sistema detecta alguna coincidencia, almacena la notificación para la mascota coincidente.

Prueba de aceptación: Gestión de mensajes de usuario

Id	PA-10	**Resumen**	Gestión de mensajes de usuario
Descripción		Se va a comprobar que el proceso de gestión de notificaciones funciona de manera correcta. Las pruebas realizadas han sido las siguientes: 1. Se comprueba en el menú de usuario que no existen notificaciones (el indicador de mensajes muestra 0) y se accede a la gestión de notificaciones. 2. Se comprueba en el menú de usuario que existen notificaciones (el indicador de mensajes muestra un valor distinto de 0) y se accede a la gestión de notificaciones.	
Resultado		Los resultados de las distintas pruebas han sido las esperadas: 1. Se muestra la pantalla de gestión de notificaciones y la lista no incluye ninguna notificación. 2. Se muestra la pantalla de gestión de	

	notificaciones y la lista incluye un número de notificaciones igual al número mostrado en el menú principal.

Prueba de aceptación: Identificar mascota como encontrada

Id	PA-11	**Resumen**	Identificar mascota como encontrada
Descripción		Se va a comprobar que el proceso de validación de notificaciones funciona de manera correcta. Las pruebas realizadas han sido las siguientes: 1. Se accede a una mascota no validada y sin modificar los datos del formulario, se graba. 2. Se accede a una mascota no validada, se confirma y se graba.	
Resultado		Los resultados de las distintas pruebas han sido las esperadas: 1. Desde la pantalla de la notificación se muestra el mensaje “No changes done”. 2. Desde la pantalla de la notificación se muestra el mensaje “Notification successfully updated”. Se muestran los campos correspondientes a los datos del usuario que localizó la mascota. En la pantalla de gestión de notificaciones se comprueba que la notificación ha quedado marcada en color verde. En la gestión de mascotas se comprueba que la mascota aparece bajo fondo blanco. En la ficha de la mascota se comprueba que ya no está marcada como perdida.	

Prueba de aceptación: Anular localización de mascota

Id	PA-12	**Resumen**	Anular localización de mascota
Descripción		Se va a comprobar que el proceso de cancelación de notificaciones funciona de manera correcta. Las pruebas realizadas han sido las siguientes: 1. Se accede a una mascota no validada y sin modificar los datos del formulario, se graba. 2. Se accede a una mascota no validada, se marca “Reject” y se graba. 3. 7. El usuario marca la opción “Deny”. 8. El sistema solicita confirmación sobre la	

	notificación. 9. El usuario confirma la notificación. El sistema desmarca la mascota como perdida y en caso de existir notificaciones coincidentes con la mascota perdida elimina las posibles notificaciones.
Resultado	Los resultados de las distintas pruebas han sido las esperadas: 1. Desde la pantalla de la notificación se muestra el mensaje "No changes done". 2. Desde la pantalla de la notificación se muestra el mensaje "Notification successfully updated". En la pantalla de gestión de notificaciones se comprueba que la notificación ha quedado marcada en color rojo. En la gestión de mascotas se comprueba que la mascota continúa apareciendo amarillo. En la ficha de la mascota se comprueba que continúa estando marcada como perdida.

Una vez realizadas las diferentes pruebas, podemos compararlo con los requisitos definidos inicialmente.

	CU01	CU02	CU03	CU04	CU05	CU06	CU07	CU08	CU09	CU10	CU11	CU12	CU13	CU14	CU15
PA01		X													
PA02	X														
PA03			X												
PA04				X											
PA05					X						X	X			
PA06						X					X	X			
PA07							X								
PA08								X							
PA09									X						
PA10										X					
PA11													X		
PA12														X	

Tabla 1 - Pruebas de aceptación vs. Requisitos

A la vista de los resultados obtenidos en las pruebas y teniendo en cuenta que la baja de usuario se desechó durante el desarrollo debido a la falta de tiempo, comprobando que no existía ningún error, se puede afirmar que la aplicación estaba lista para ser puesta en producción como la primera versión.

8. OBJETIVOS CONSEGUIDOS

Con los resultados obtenidos, los objetivos de este Trabajo Fin de Grado, tanto generales como específicos se han cubierto satisfactoriamente. Los objetivos relacionados con la tecnología Java así como los relacionados con Android han sido totalmente satisfechos mientras que los objetivos relacionados con la aplicación a desarrollar han sido implementados.

A continuación se describen los diferentes objetivos planteados en el Trabajo Fin de Grado y comentarios relacionados con la forma de alcanzar los mismos:

Se ha cumplido el **objetivo general** gracias al desarrollo de una aplicación ágil, intuitiva y sencilla que permite la identificación y localización de animales de compañía.

Se han cumplido los diferentes **objetivos específicos** planteados al inicio del trabajo:

- Realizando diferentes análisis de requerimientos para los actores de la aplicación.

- Realizando el diseño e implementación de la base de datos utilizada por el sistema para la persistencia de datos.

- Profundizando en el conocimiento de la arquitectura para dispositivos móviles Android así como en el desarrollo mediante tecnologías Java para dicha plataforma, para el cual no se poseía conocimiento alguno anterior.

- Realizando diferentes baterías de pruebas asegurando la fiabilidad del sistema y correcto funcionamiento de los procesos definidos previamente.

- Cubriendo los diferentes requisitos definidos en la etapa previa de análisis de la herramienta.

9. CONCLUSIONES

En este apartado se presentarán las conclusiones posteriores a la realización del proyecto.

Uno de los motivos que me llevó a hacer este proyecto era la posibilidad de programar una aplicación para la plataforma Android junto a un conjunto de tecnologías como Java, bases de datos, etc. El incesante uso de dispositivos móviles y tablets en los últimos años, ha convertido a esta tecnología en la más usada y con alcance a un mayor número de usuarios, incluso aquellos que nunca han usado un ordenador personal, como se ha podido ver en capítulos anteriores. Sumado a ello, la tecnología que se ha ido incorporando en los últimos años a estos dispositivos (desde cámaras de foto y vídeo, GPS, inclinómetro…) ofrece un gran potencial a los desarrolladores de aplicaciones.

Uno de los objetivos que me había planteado incluso antes de la elección del tema del proyecto era el de obtener un “producto” práctico, que mejorase o facilitase la tarea a realizar. No se trataba solo de algo que funcionase y que visualmente fuera atractivo, pero al fin y al cabo, otro “programa más de bibliotecas”, “un programa más bonito de gestión de habitaciones de hotel”, etc.

La realización de este proyecto, me ha permitido conocer una gran variedad de tecnologías con las que nunca había trabajado. Desde Android, pasando por Java, con el que no había profundizado hasta el nivel actual, las diferentes APIs utilizadas, las diferentes posibilidades que ofrecen los dispositivos móviles inteligentes actuales, etc. He de remarcar, que la curva de aprendizaje por estos motivos, fue muy intensa, pero muy productiva posteriormente. Se han cumplido los requisitos propuestos al inicio del trabajo. Se ha trabajado utilizando metodologías de trabajo y bajo una planificación inicial, lo cual ha permitido realizar diferentes puntos de control para analizar el estado de la aplicación.

La mayor dificultad con la que me he encontrado en el proyecto ha sido, el poner en funcionamiento la API de reconocimiento facial, ya que salvo la documentación de los desarrolladores, no se ha encontrado información muy práctica en Internet. Por tanto, en ocasiones ha habido que analizar el código de la propia API (una API desarrollada en C++). Esto, me llevó gran tiempo durante el desarrollo del proyecto y conllevó que se desechasen ciertas funcionalidades adicionales que se hubiera deseado incluir, como por ejemplo, la geolocalización por medio de dispositivos GPS para identificar puntos de localización, direcciones, etc.

Las conclusiones que puedo sacar de la realización del proyecto por tanto, son muy positivas.

10. TRABAJO FUTURO

Toda aplicación es susceptible de mejora y ampliaciones, más si cabe, teniendo en cuenta el avance de las tecnologías en las últimas décadas, de las que hacen uso las diferentes aplicaciones integrándolas en nuevos servicios, dotándolas de mayor capacidad de proceso, etc.

Una vez desarrollada la primera versión de la aplicación, en este apartado se plantearán diferentes propuestas de mejora para versiones posteriores.

- Mejora de la experiencia de usuario: incluye el desarrollo para diferentes plataformas (iOS, Windows Mobile…) así como la adaptación a diferentes tipos de dispositivos mediante la adaptación de pantallas, resolución, botones, etc. Para ello, sería necesario crear nuevas plantillas así como incluir imágenes a diferente resolución.

- Incluir geolocalización para indicar los diferentes lugares o direcciones que aparecen en la aplicación. Debido al limitado tiempo para el desarrollo de esta aplicación, no se ha incluido en la primera versión. Para ello se podría incluir la API de geolocalización y adaptar las diferentes pantallas para acceder al mismo.

- Adaptar la lógica de API OpenCV con sus diferentes actualizaciones para optimizar la localización de mascotas. Actualmente, debido al limitado tiempo para el desarrollo de la aplicación, la búsqueda de mascotas perdidas se realiza únicamente cuando un usuario introduce una posible localización (podría realizarse también cuando el usuario marca una mascota como perdida o bien hacerse periódicamente en la parte del servidor mediante threads o mediante timers).

- Adaptar la aplicación a diferentes idiomas. Debido al limitado tiempo para el desarrollo del trabajo se ha optado únicamente por el idioma inglés. Para ello sería necesario modificar los diferentes ficheros de cadenas de texto las cuales serían accesibles según la configuración local del dispositivo.

- Administración de la aplicación: se puede realizar un sistema de gestión de la aplicación por parte de usuarios administradores (para gestión de usuarios, gestión de mensajes, etc).

- Gestiones adicionales como el envío automático de mensajes o correo electrónico cuando el usuario se registra o bien cuando se registra una notificación asociada a su cuenta podrían analizarse para ampliaciones futuras.

- Incluir gestiones sobre los usuarios como la baja de usuarios permitiendo de esta manera desligar al usuario de la aplicación eliminando todos los datos del mismo (mascotas, notificaciones, etc).

- Limitar los valores introducidos en la aplicación creando listas desplegables que

el usuario debe seleccionar, facilitando y agilizando de este modo la introducción de datos en la aplicación por parte del usuario y a su vez optimizando la información contenida en la base de datos (por ejemplo mediante listas desplegables de países para la dirección, razas, etc).

Glosario de términos

Cloud computing	Consiste en la posibilidad de ofrecer servicios a través de Internet. La computación en nube es una tecnología nueva que busca tener todos nuestros archivos e información en Internet y sin depender de poseer la capacidad suficiente para almacenar información.
GPL	Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License (GNU GPL) es la licencia más ampliamente usada en el mundo del software y garantiza a los usuarios finales la libertad de usar, estudiar, compartir y modificar el software.
HTTP	Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web.
JSON	JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.
Nube	Metáfora empleada para hacer referencia a servicios que se utilizan a través de Internet.
POJO	Un POJO (acrónimo de Plain Old Java Object) enfatiza el uso de clases simples y que no dependen de un framework en especial.
SQL	Lenguaje de consulta estructurado o en inglés, Structured Query Language es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

Bibliografía

(Amaro, 2013)	**Amaro, José Enrique.** (2013) El gran libro de programación avanzada con Android, México: Alfaomega Grupo Editor, S.A., 2013, ISBN 978-607-707-556-6
(Bell, 1973)	**Bell, Daniel.** (1973) The Coming of Post-Industrial Society: A Venture in Social Forecasting, New York: Basic Books, 1973, ISBN 0-465-01281-7
(Cockburn, 2000)	**Cockburn, Alistair.** (2000) Writing Effective Use Cases, Addison-Wesley, 2000, ISBN 0-201-70225-8
(Eckel, 2007)	**Eckel, Bruce.** (2007) Piensa en Java, Pearson Educación, S.A., 2007, ISBN 978-84-8966-034-2
(Gargenta, 2011)	**Gargenta, Marko.** (2011) Learning Android, O'Really Media, Inc., 2011, ISBN 978-1-449-39050-1
(Gilgillan, 2003)	**Gilfillan, Ian.** (2003) La Biblia de MySQL, Anaya Multimedia, 2003, ISBN 978-8-441-51558-1
(Milicev, 2009)	**Milicev, Dragan.** (2009) Model-Driven Development With Executable UML, Wiley Publishing, Inc., 2009, ISBN 978-0-470-48163-9
(Murphy, 2011)	**Murphy, Mark L.** (2011) Android Programming Tutorials, CommonsWare, 2011, ISBN 978-0-981-67804-7
(Stellman-Greene, 2006)	**Stellman, Andrew and Green, Jennifer.** (2006) Applied Software Project Management, O'Reilly Media, Inc., 2006, ISBN 978-0-596-00948-9
(Young, 2004)	**Young, Ralph R.** (2004) The Requirements Engineering Handbook, Artech House, 2004, ISBN 1-58053-266-7

Webgrafía

[1] **Arxiu d'Identificació d'Animals de Companyia (AIAC)**
http://www.veterinaris.cat/CONSELL/cat/aiac.asp

[2] **Introducing JSON**
http://www.json.org/

[3] **Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal**
http://www.agpd.es/portalwebAGPD/canaldocumentacion/legislacion/estatal/common/pdfs/LOPD_consolidada.pdf

[4] **Maven**
http://maven.apache.org

[5] **MySQL**
http://www.mysql.com/

[6] **OpenCV**
http://opencv.org/

[7] **Registro Informático Valenciano de Identificación Animal**
http://www.rivia.org

ANEXOS

Anexo I. Configuración del entorno de desarrollo

En primer lugar, ya que se va a utilizar un desarrollo realizado para plataforma Android, es necesario descargar el ADT Bundle for Windows desde el enlace http://developer.android.com/sdk/index.html de la versión de Microsoft Windows correspondiente, 32-bit o 64-bit, la cual permite disponer de las librerías de Android, plugins, AVD, etc. Para el desarrollo del trabajo se ha utilizado la versión del fichero adt-bundle-windows-x86_64-20140624.

Descomprimiremos el fichero en el disco, a ser posible en el directorio raíz para posteriormente facilitar el trabajo.

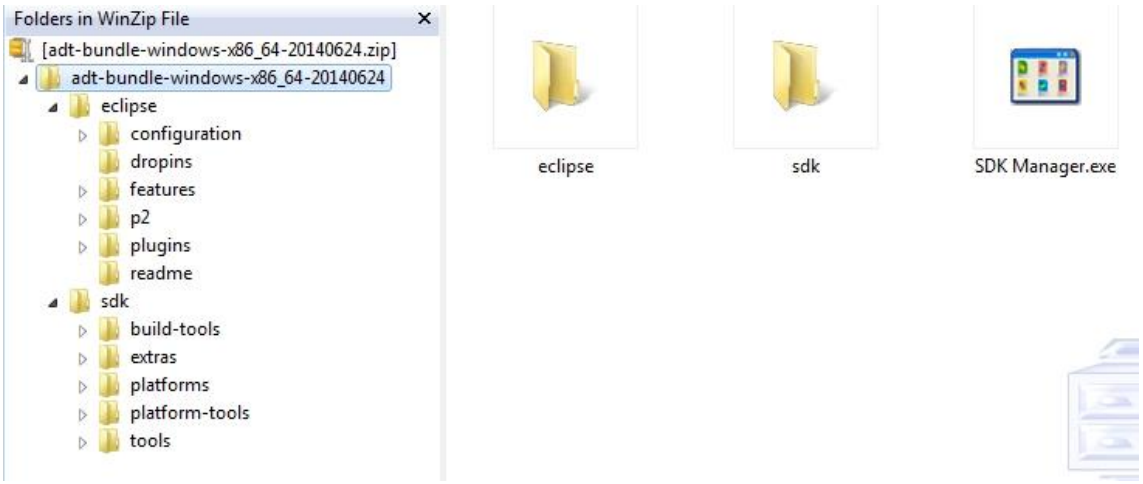

Ilustración 26 - Contenido del ADT Bundle

Ejecutamos el IDE Eclipse, el cual contiene los plugins y herramientas de desarrollo Android. Al ejecutar el IDE Eclipse por primera vez, se le solicita al usuario el directorio workspace, en el cual se crearán todas las carpetas necesarias por el proyecto. Configuramos una carpeta de nuestro equipo como workspace.

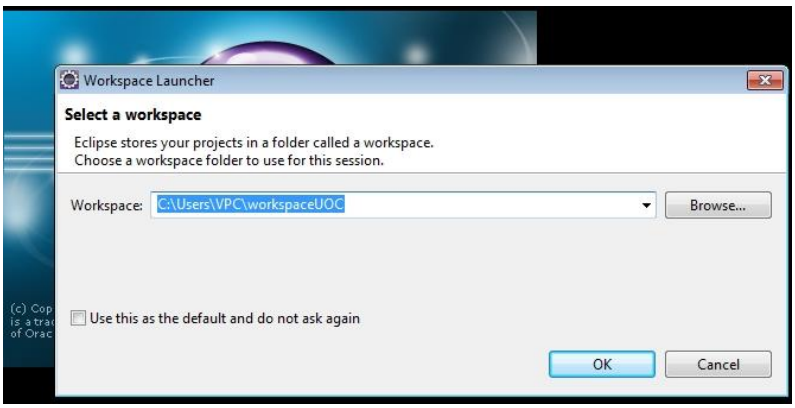

Ilustración 27 - Selección de workspace

En la pantalla principal de Eclipse, tenemos la opción “Android SDK Manager”, desde donde podemos instalar los paquetes necesarios de Android, plugins, herramientas e incluso plantillas de dispositivos virtuales que utilizaremos como si se tratase de un dispositivo real para realizar el desarrollo. En nuestro caso, trabajaremos

con la API 19, la cual corresponde con la versión 4.4.2 de Android.

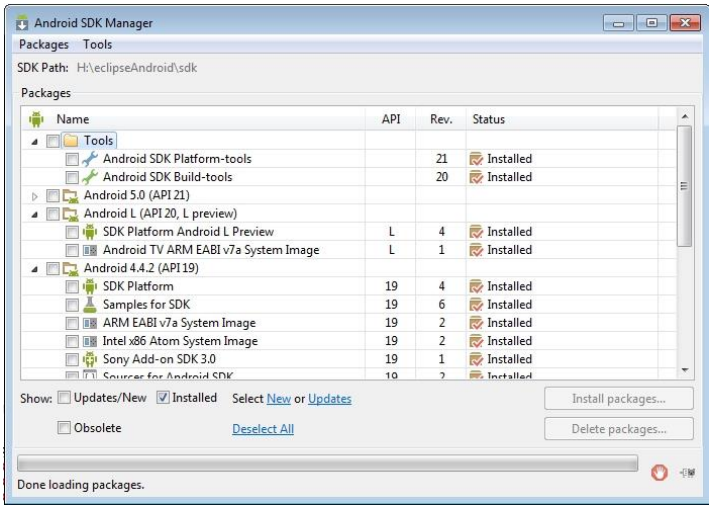

Ilustración 28 - Gestión de paquetes Android SDK Manager

Para crear una aplicación Android, pulsaremos sobre File – New – Other y seleccionaremos Android Application Project (también podríamos seleccionar la opción Android Sample Project, en este caso, nos crea un código de ejemplo "Hello World").

Una vez seleccionado, se le solicitan al usuario varios datos que permitirán definir posteriormente el proyecto.

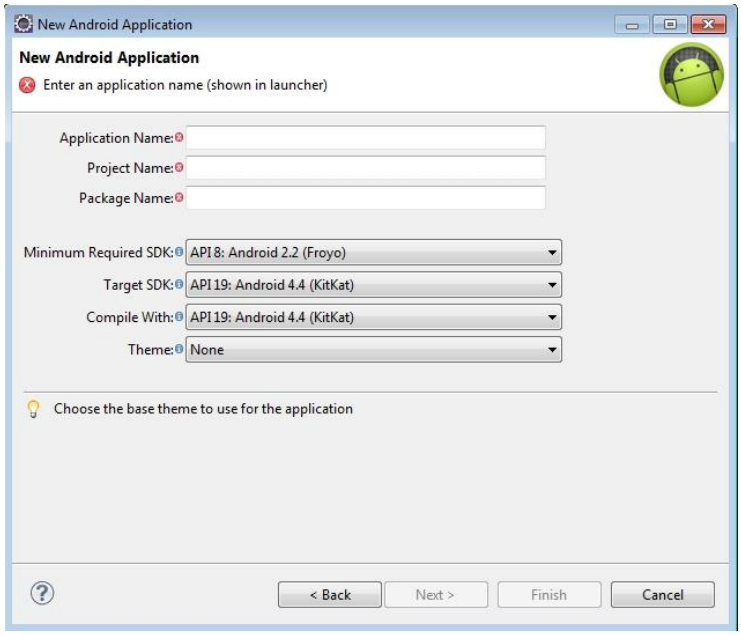

Ilustración 29 - Datos para la creación de proyecto Android

Los datos solicitados al usuario y necesarios para la generación del proyecto son los siguientes:

- Application name: nombre que se le dará a la aplicación (en nuestro caso lo llamaremos wheresmypet-android).

- Project name: nombre de la carpeta del proyecto (le daremos el mismo nombre).

- Package name: se trata del paquete en el que generaremos nuestra aplicación y que posteriormente utilizaremos para referenciar desde nuestras clases (en nuestro caso lo hemos llamado com.wheresmypet.android).

- Minimum Required SDK: se trata de la versión mínima que soportará la aplicación. En este punto, es aconsejable seleccionar la menor versión, ya que de esta forma, la aplicación estará soportada por el mayor número de dispositivos. Sin embargo, las continuas actualizaciones del sistema Android hace que aparezcan nuevas funcionalidades, las cuales solo son soportadas por versiones posteriores a esta, por lo tanto, la elección también dependerá de las funciones que queramos incluir en nuestra aplicación. En nuestro caso, usaremos la API 8, versión 2.2 de Android como mínimo SDK requerido.

- Target SDK: indica la versión máxima sobre la que la aplicación ha sido testeada. Por tanto, junto con la opción anterior, se establece un entorno de APIs sobre las cuales la aplicación debe correr sin problema alguno. Para nuestro trabajo se utilizó la versión más reciente en el momento de comenzar el desarrollo, esta era la versión 19, versión de Android 4.4 (sin embargo, durante la realización del trabajo aparecieron las versiones API 20 y API 21, esta última, correspondiente a la versión 5.0 de Android).

- Compile With: se trata de la versión con la que se compilará la herramienta. Para la práctica, se selecciona la misma versión que el punto anterior.

- Theme: especifica el estilo de la interface de usuario que utilizará la aplicación por defecto. En nuestro caso, seleccionaremos “None” pare crear nuestra propia interface.

Una vez seleccionadas las diferentes opciones, se le solicitará al usuario el icono que tendrá la aplicación y si crear o no una actividad principal. En nuestro trabajo, no crearemos por defecto el icono de la aplicación pero sí una actividad principal, la cual llamaremos activity_main.

Una vez creado el proyecto, para poder ejecutar nuestro desarrollo, crearemos un dispositivo virtual de Android (AVD).

Para ello, deberemos acceder a la opción de Eclipse “Android Device Manager”. El gestor de dispositivos de Android, nos permite utilizar diferentes plantillas de dispositivos predefinidos (en la pestaña “Device Definitions”), los cuales podremos descargarlos desde el “Android SDK Manager” que hemos visto anteriormente.

En la pestaña Android Virtual Devices, seleccionaremos la opción “Create”, con

lo que se nos abre un nuevo formulario en el que se introducirán las características del dispositivo que queremos crear. Pare el desarrollo de la práctica, hemos utilizado un dispositivo con las características que pueden verse en la siguiente ilustración:

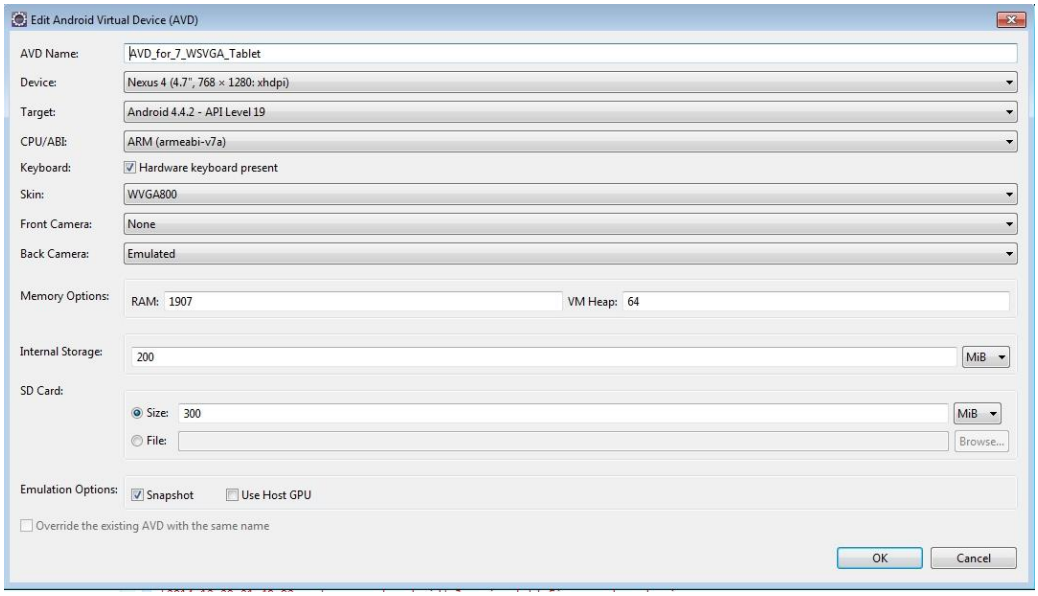

Ilustración 30 -Creación de dispositivo virtual de Android

En la imagen anterior podemos ver que en este punto se introducen datos como la memoria que tendrá el dispositivo, la emulación de otros dispositivos como la cámara, etc.

Con esto, tendremos creado nuestro entorno de desarrollo en Eclipse con soporte para Android. Para ejecutar nuestro proyecto, bastará con colocarnos encima de la raíz del mismo, pulsar el botón derecho del ratón y seleccionar Run As – Android Application.

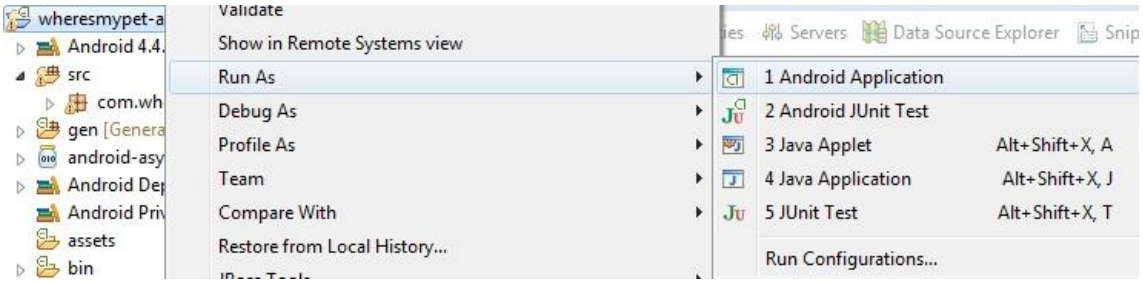

Ilustración 31 - Ejecución de aplicación Android desde Eclipse

Una vez ejecutada la aplicación, esta será instalada en el dispositivo virtual, con lo que podremos seguir la ejecución de la misma dentro del dispositivo virtual de Android.

Anexo II. Manual de instalación

Proyecto cliente

Para instalar la aplicación cliente sobre el entorno de desarrollo se puede importar el proyecto Android. Como se ha comentado en el Anexo I, el proyecto Android es wheresmypet-android. Eclipse añade también de forma automática el proyecto appcompat_v7 para dar soporte a las bibliotecas por defecto de Android, por lo que también formará parte del proyecto de Android. De esta manera se puede acceder al código de la aplicación.

Instalación sobre dispositivo virtual de Android (AVD)

Como se ha comentado al final del Anexo I, al ejecutar el proyecto Android sobre un dispositivo previamente configurado, Eclipse instala la aplicación en el dispositivo y realiza la ejecución de la aplicación.

Una vez ejecutada la aplicación, podremos seguir la ejecución de la misma dentro del dispositivo virtual de Android.

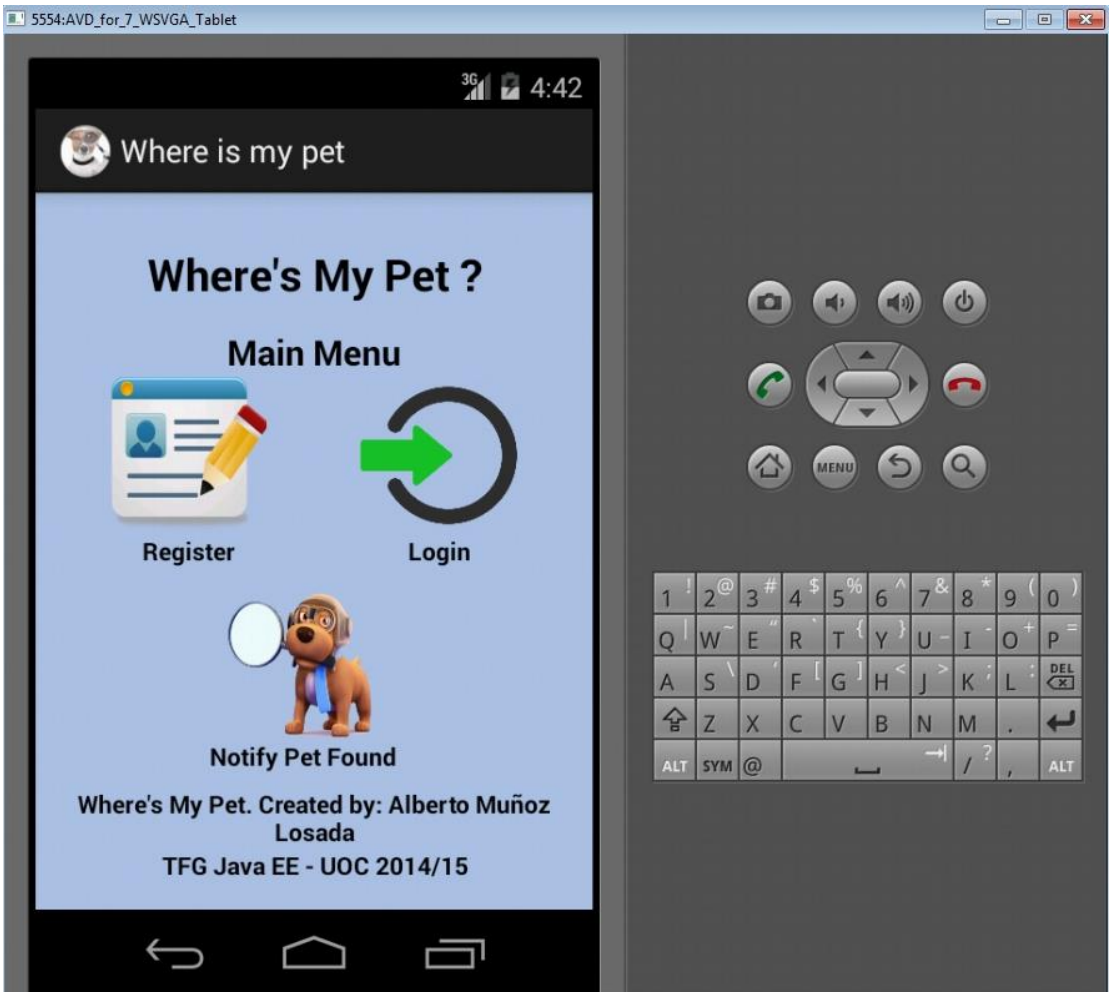

Ilustración 32 - Ejecución de la aplicación bajo AVD

Instalación en dispositivo físico

Dentro de la carpeta bin del proyecto cliente de Where's My Pet para Android, se deja el archivo ejecutable "WheresMyPet.apk" al compilar la aplicación desde Eclipse.

Para la instalación de la aplicación cliente sobre un dispositivo físico, será necesario copiar el fichero a la tarjeta de memoria del dispositivo conectando este último al PC mediante un cable USB, para posteriormente ejecutarlo desde el dispositivo.

NOTA IMPORTANTE: *En el caso de instalar el software en un dispositivo virtual y querer acceder a otro servidor, será necesario modificar el fichero Constants.java localizado dentro de la carpeta src/com/wheresmypet/android, en el que se almacena la información acerca del servidor al que conectar antes de realizar la compilación del fichero .apk (para el desarrollo se utilizó una máquina virtual con dirección IP 10.0.2.2 mientras que para la puesta en producción y compilación que adjunta el proyecto se utilizó la dirección pública del servidor en la nube).*

Instalación de la parte servidor

Se opta por un servidor Windows Server 2012 en la nube para la instalación del proyecto, en el que se instalarán las diferentes herramientas y sobre el que se ejecutará posteriormente la aplicación. Esta, en la puesta en producción se configurará para que corra en modo servicio.

Para la parte del servidor necesitaremos instalar las siguientes herramientas:

MySQL Server

Se trata del servidor de base de datos bajo el que funcionará la aplicación. Para ello, desde la página http://dev.mysql.com/downloads/ descargaremos la versión MySQL Community Server (versión bajo licencia GPL). En nuestro caso, la versión de la herramienta es la 5.6.22. También se puede descargar e instalar la herramienta MySQL Workbench (herramienta que permite la gestión, diseño, etc de MySQL de forma visual) de forma independiente o bien incluirlo en las opciones de instalación de MySQL Community Server. Para el trabajo se utiliza la versión 6.2.4 de MySQL Workbench.

La instalación de MySQL Server no implica ninguna dificultad. Al usuario se le solicitan diferentes datos. Se optará por la versión "Developer Default" instalando previamente los requerimientos necesarios indicados por la aplicación.

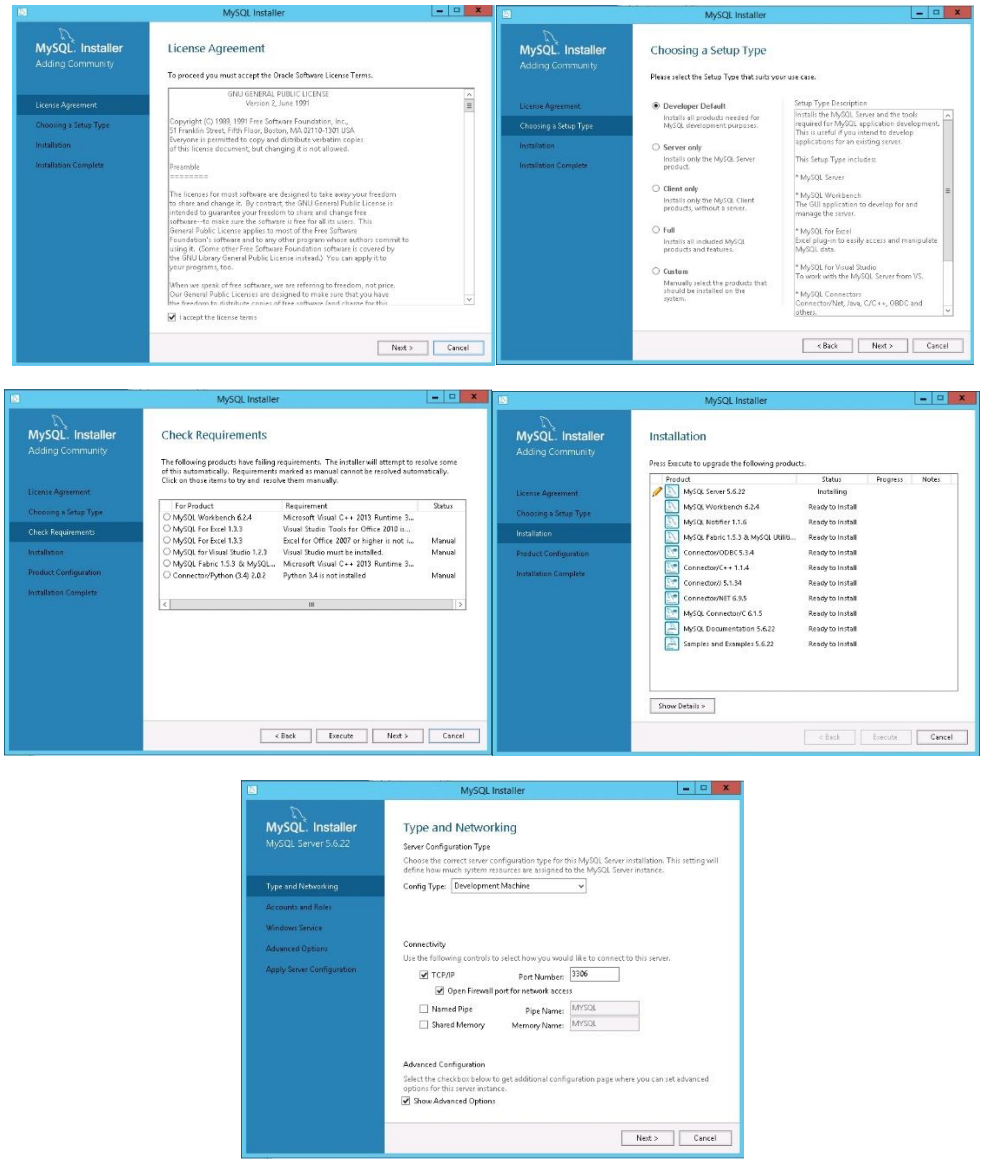

Ilustración 33- Instalación de MySQL Server

Es necesario prestar especial atención al momento de crear la contraseña de usuario root así como los posibles usuarios de la base de datos (aunque estos podrán ser configurados más tarde). El usuario root, tiene acceso absoluto sobre la base de datos y en nuestro proyecto Java, deberemos configurar este (algo no aconsejable en entornos de producción) o bien un usuario con permisos sobre la base de datos para tener acceso a la misma.

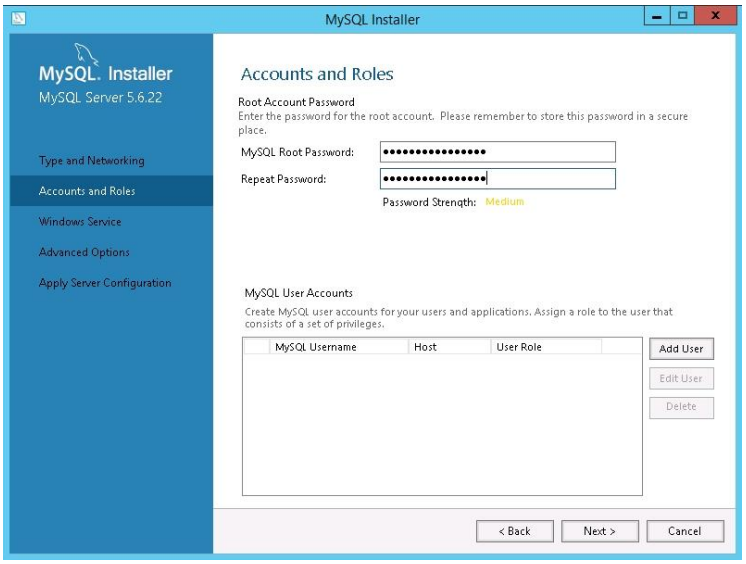

Ilustración 34 - Pantalla de creación de usuarios de BBDD y contraseña root

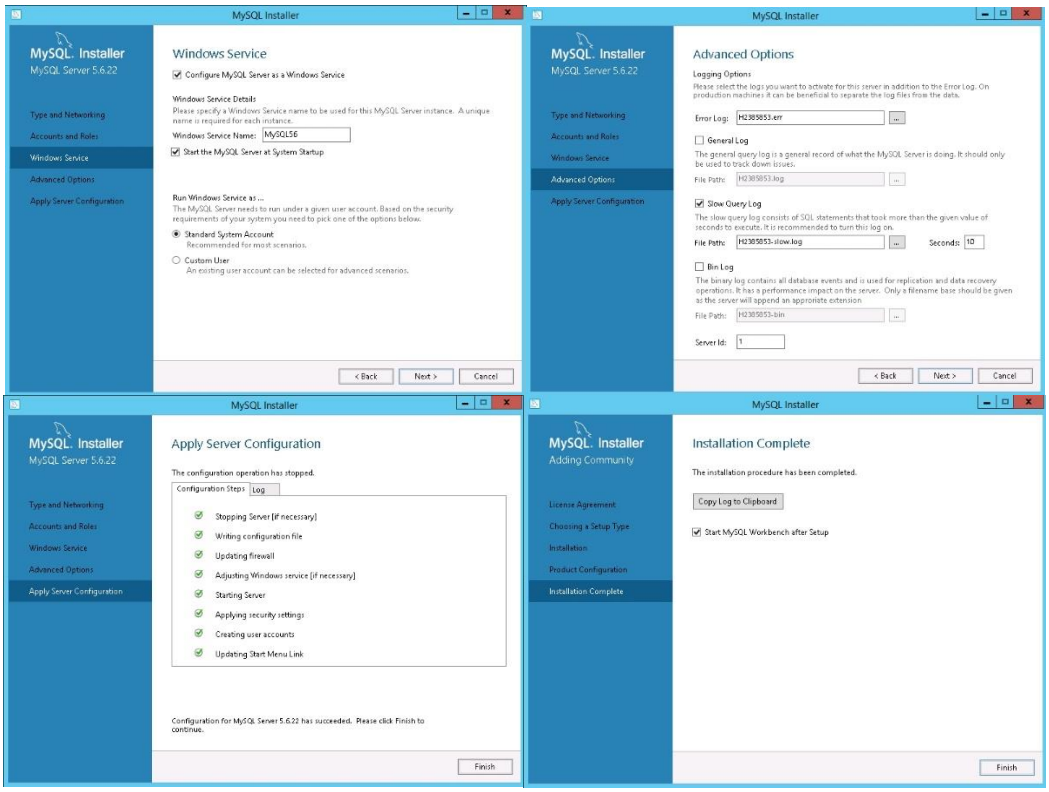

Ilustración 35 - Instalación de MySQL Server

No instalaremos los ejemplos que se adjuntan en el último paso y con esto, finaliza la instalación de MySQL Server. Para comprobar el correcto funcionamiento, podremos ejecutar MySQL Workbench y tras validar usuario y contraseña, tener acceso a MySQL.

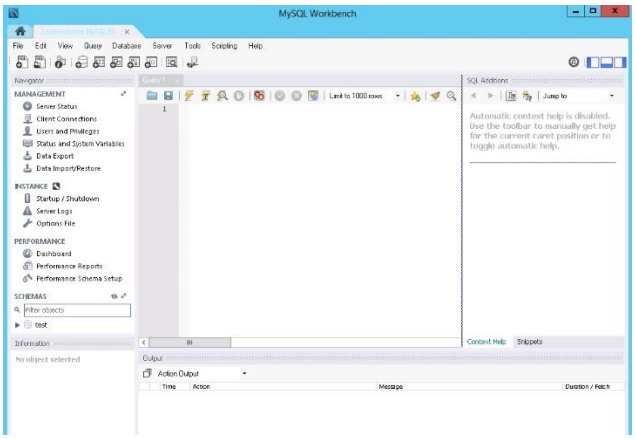

Ilustración 36 - Acceso a MySQL mediante MySQL Workbench

Por último, será necesario importar la estructura de BBDD incluida en el trabajo para crear el esquema y las diferentes tablas necesarias para la aplicación. La estructura se encuentra en un fichero de texto en la carpeta raíz del proyecto.

Java SE Development Kit 7

Instalaremos el runtime de Java, el cual permite la ejecución de programas en Java. Es conveniente instalar la última versión del runtime de Java que puede descargarse desde http://www.oracle.com/technetwork/java/javase/downloads/index.html.

En nuestro caso, instalaremos la versión 7u72 del SDK (herramientas de desarrollo para entorno Java) para 32 bits, el cual incluye el runtime de Java. Lo instalaremos con las opciones que se muestran por defecto, teniendo en cuenta la carpeta de instalación que posteriormente deberemos incluir en el path de nuestro sistema.

Ilustración 37 - Instalación de Java SDK

Una vez instalado, crearemos dos variables de sistema:

- JAVA_HOME: que apuntará a la ruta donde hemos instalado Java.

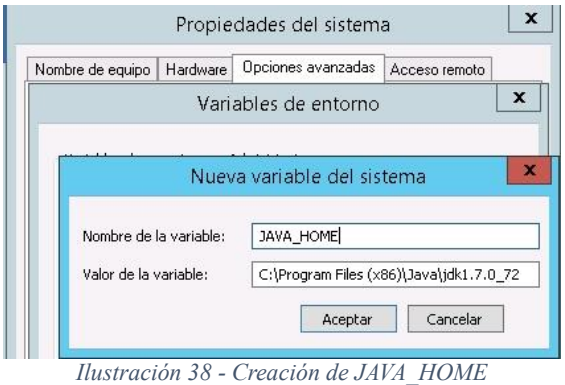

Ilustración 38 - Creación de JAVA_HOME

- En segundo lugar añadiremos en la variable PATH la ruta a la carpeta "bin" del runtime para acceder fácilmente a los ejecutables de Java.

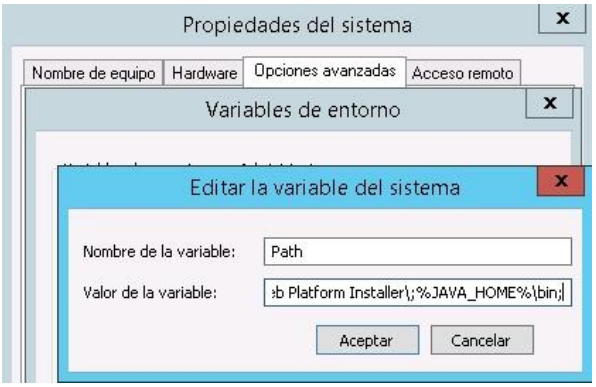

Ilustración 39 - Inserción de ruta en el PATH de sistema

Con esto, concluiríamos la instalación de Java (una comprobación que podemos hacer es abrir una ventana de comandos y ejecutar el comando java –version, lo cual nos debería devolver la versión instalada de Java).

Ilustración 40 - Versión instalada de Java

OpenCV

Se trata de una biblioteca publicada bajo licencia BSD, de uso libre para la visión artificial originalmente desarrollada por Intel. Pese a que OpenCV se incluye en las dependencias del proyecto Maven, los desarrolladores sugieren que esta sea instalada sobre el sistema que corre la aplicación. A su vez, también se indica en su web la incompatibilidad con diferentes versiones de JavaCV (que veremos posteriormente), por lo que instalaremos para nuestra aplicación la versión 2.4.9 para Windows de la biblioteca.

Para ello, la descargaremos desde la página http://opencv.org/downloads.html. Para instalar la biblioteca, es necesario ejecutar el fichero descargado y extraer el contenido. Los desarrolladores, aconsejan que sea extraído sobre la raíz de la unidad. En

nuestro caso lo instalaremos sobre C:\OpenCV. La ruta será necesario ser incluída posteriormente en nuestro path del sistema.

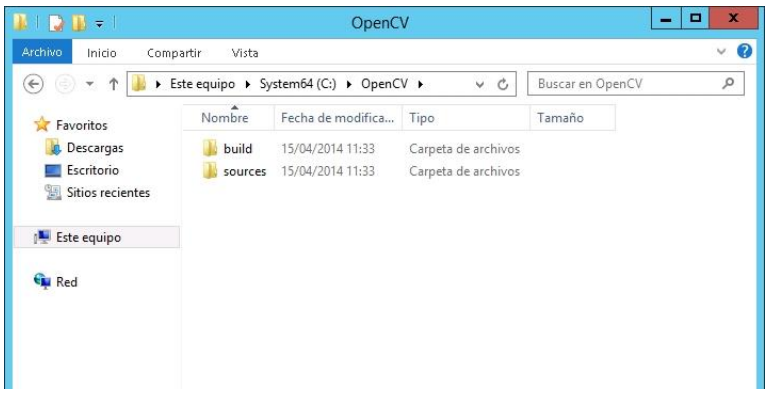

Ilustración 41 - Contenido de la carpeta OpenCV

Una vez instalado, crearemos dos variables de sistema:

- OPENCV_HOME: que apuntará a la carpeta build de nuestra instalación de OpenCV.

Ilustración 42 - Creación de variable OPENCV_HOME

- En segundo lugar añadiremos en la variable PATH la ruta a la carpeta en la que se encuentran las diferentes librerías de OpenCV (%OPENCV_HOME%\x86\vc10\bin) para hacerlas accesible a nuestra herramienta.

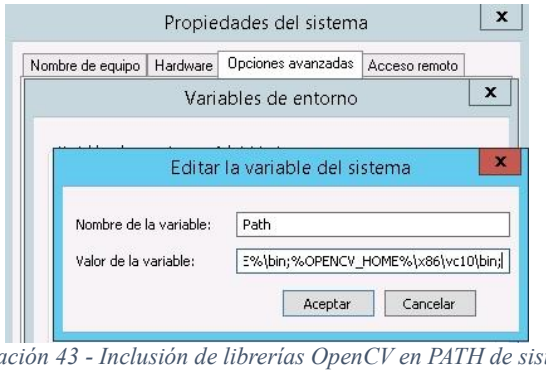

Ilustración 43 - Inclusión de librerías OpenCV en PATH de sistema

Maven

Nuestro proyecto, está empaquetado con Maven, una herramienta de software que facilita la gestión y construcción de proyectos Java. Un proyecto Maven incluye un fichero pom.xml el cual describe el proyecto de software a generar, las dependencias de otros módulos y componentes externos y el orden de construcción de elementos.

Maven, puede descargarse desde http://maven.apache.org/download.cgi, para nuestro proyecto, utilizaremos la última versión estable disponible, 3.2.5, de la que descargaremos los binarios.

Descomprimimos el fichero descargado sobre la raíz de nuestra unidad, en nuestro caso C:.

Una vez descomprimido, crearemos dos variables de sistema:

- MAVEN_HOME: que apuntará a la carpeta de instalación de Maven.

Ilustración 44 - Creación de variable MAVEN_HOME

- En segundo lugar añadiremos en la variable PATH la ruta a la carpeta en la que se encuentran los ejecutables de Maven (%MAVEN_HOME%\bin) para hacerlas accesibles a nuestra herramienta.

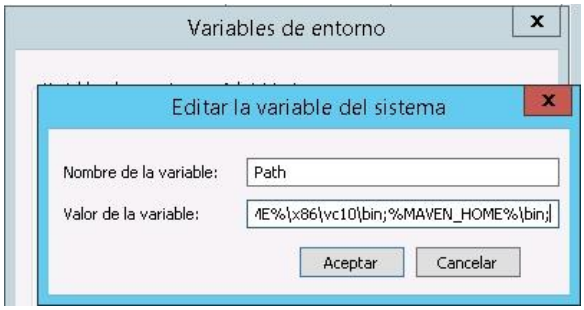

Ilustración 45 - Inclusión de librerías MAVEN en PATH de sistema

Aplicación “Where’s My Pet”

Descomprimiremos en el servidor el proyecto wheresmypet (versión servidor). Desde una ventana de comandos, iremos a la carpeta donde hemos descomprimido el proyecto y ejecutaremos el mismo con Maven mediante el comando:

mvn clean install -Dplatform.dependencies -Dmaven.tomcat.port=8080 tomcat:run

Veremos cómo Maven actualice todas las dependencias y una vez finalizada la ejecución, tendremos el servidor en funcionamiento.

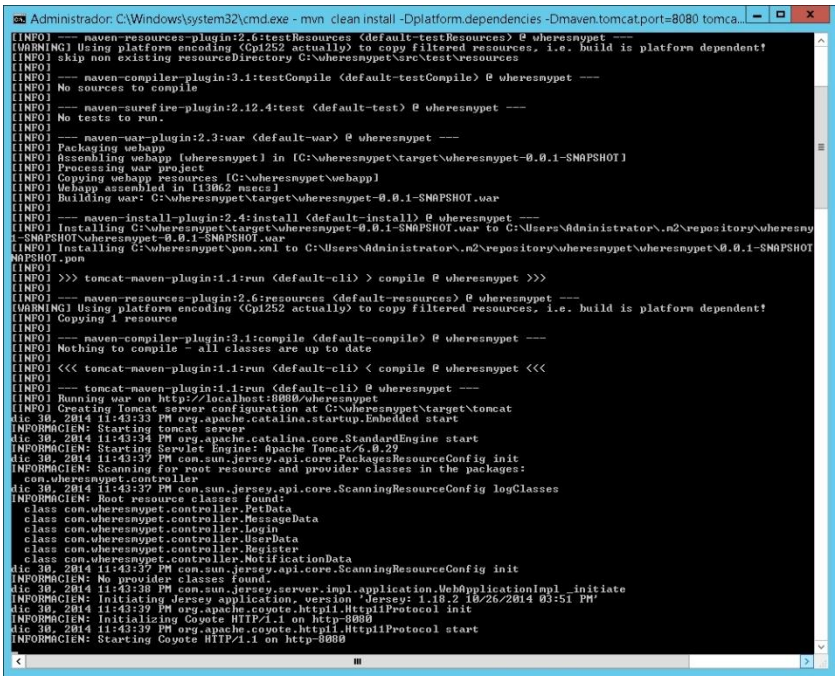

Ilustración 46 - Ejecución de la aplicación servidor mediante Maven

Anexo III. Manual del usuario

Este manual pretende ofrecer una visión rápida sobre cómo utilizar la aplicación Where's my Pet. El uso de la aplicación no obliga al usuario a estar registrado, sin embargo, el registro proporcionará al usuario un mayor número de utilidades dentro de la aplicación como la gestión de mascotas, notificaciones, etc.

Un usuario no registrado, tan solo podrá registrar avisos de mascotas localizadas.

Pantalla principal de la aplicación

La pantalla principal de la aplicación muestra al usuario las diferentes opciones disponibles.

Ilustración 47 - Pantalla principal

Las opciones que se le muestran al usuario son las siguientes:

- Register: permite el registro de un nuevo usuario.
- Login: Permite el login de usuario en la aplicación y acceder a las opciones propias del usuario.
- Notify Pet Found: permite registrar una mascota localizada.

Registro de usuario

La pantalla de registro de usuario muestra los campos básicos necesarios para el registro de usuario.

Ilustración 48 - Pantalla Registro de usuario

Una vez introducidos los datos del usuario, se debe pulsar el botón "Register" para almacenar los datos del usuario. Una vez registrado el usuario, este es redirigido a la pantalla de login.

Login de usuario

La pantalla de login de usuario muestra los campos necesarios para acceder a la aplicación. El usuario debe introducir sus datos correctamente y pulsar el botón del "login" para acceder a su menú personal.

Ilustración 49 - Pantalla de Login de usuario

Registro de mascota localizada

Cuando un usuario de la aplicación localiza a una mascota perdida, este, puede crear un aviso en la aplicación. La aplicación comparará la fotografía enviada con las fotografías de las mascotas perdidas y en caso de existir coincidencia, generará notificaciones al usuario o usuarios dueños de la mascota localizada.

Para el registro de una posible mascota localizada, el usuario debe introducir los diferentes datos del formulario que puedan servir posteriormente al posible dueño para ponerse en contacto, estos datos, serán únicamente mostrados en el caso de que el dueño de la mascota perdida confirme que se trata de su mascota, no siendo por tanto visibles de manera inicial.

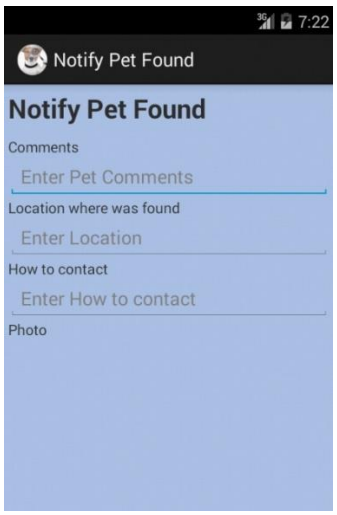

Ilustración 50 - Pantalla de Registro de mascota localizada

MENÚ DE USUARIO

Una vez confirmado el acceso del usuario, a este se le presenta el menú de usuario, como puede verse en la siguiente ilustración:

Ilustración 51 – Pantalla Menú de usuario

Las opciones que se le muestran al usuario registrado son las siguientes:

- Modify User: permite modificar los datos propios al usuario (dirección, teléfono, etc).
- Pet Manager: permite gestionar las diferentes mascotas del usuario.
- Notifications: permite registrar las diferentes notificaciones generadas por la aplicación al localizar una mascota perdida.

- Logout: permite al usuario salir de la parte privada de la aplicación.

Modificación de datos de usuario

La pantalla de modificación de datos de usuario muestra diferentes datos referentes al usuario. La aplicación permite registrar la ficha de usuario con los datos que el usuario crea necesarios. Una vez introducidos los datos, el usuario debe pulsar el botón “Save” para que estos queden registrados.

Ilustración 52 - Pantalla Modificación de datos de usuario

Gestión de mascotas

La pantalla de gestión de mascotas permite al usuario crear y gestionar las diferentes mascotas dadas de alta, pudiendo entre otras cosas, marcarlas como perdidas.

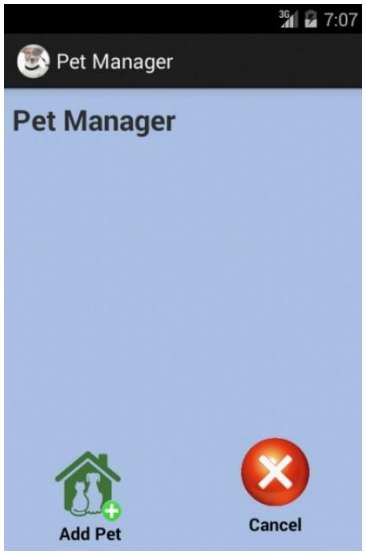

Ilustración 53 - Pantalla de gestión de mascotas sin mascotas creadas previamente

Gestión de notificaciones

Las posibles coincidencias que el sistema detecta para las mascotas perdidas y potencialmente localizadas se muestran mediante notificaciones al propietario de las mascotas. Desde el menú de usuario se dispone de un botón “Notifications” sobre el cual se indicará el número de notificaciones que tiene un usuario. Una vez accedido a la gestión de notificaciones, el usuario podrá actualizar el estado de las mismas.

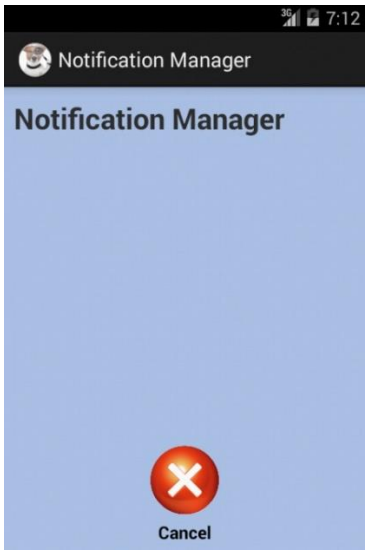

Ilustración 54 - Pantalla Gestión de notificaciones sin notificaciones de usuario

Logout

Para salir de la parte privada de la aplicación, es necesario realizar un logout de la misma. Una vez pulsado el botón “Logout”, se solicitará al usuario confirmación para salir de la aplicación.

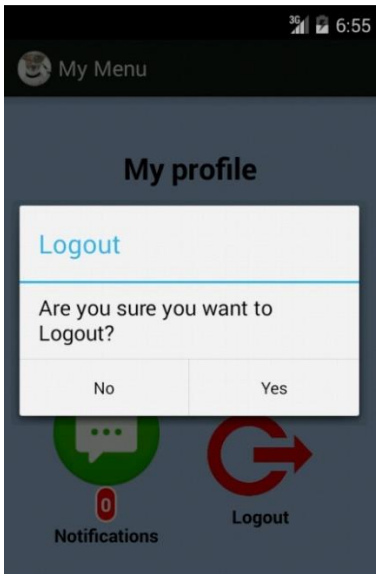

Ilustración 55 -Confirmación de Logout de la aplicación

GESTIÓN DE MASCOTAS

Los estados de las mascotas serán visualmente identificables mediante códigos de colores. La lista de mascotas mostrará las mascotas sobre fondo de diferente color en función del estado de la notificación:

- Amarillo: mascota marcada como perdida.
- Blanco: mascota no perdida.

Ilustración 56 - Pantalla Gestión de mascotas con identificación por color

Añadir nueva mascota

A la pantalla de registro de mascotas se accede desde la gestión de mascotas pulsando el botón “Add Pet”. La pantalla resultante muestra los campos básicos necesarios para el registro de mascotas.

La aplicación permite el guardado de imágenes de mascotas tanto de la cámara como de la tarjeta de memoria del dispositivo. Para adjuntar una fotografía de la cámara será necesario pulsar el botón “Take Photo” y realizar una fotografía. Para registrar una fotografía desde la galería del dispositivo, será necesario pulsar “Select Photo”.

El usuario podrá eliminar la foto de una mascota pulsando el botón “Delete Photo”. Una vez introducidos los datos de la mascota, se debe pulsar el botón “Save” para almacenarlos en el sistema. El sistema en ese momento mostrará la pantalla de gestión de mascotas mostrando la nueva mascota introducida anteriormente.

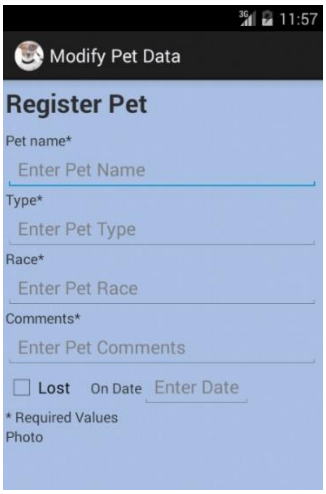

Ilustración 57 - Pantalla de Creación de mascota

Modificar datos de mascota

El sistema permite modificar los datos una vez registrada la mascota. Para ello, desde la pantalla de gestión de mascotas, pulsando sobre la mascota que se desee modificar, se abre la ficha de datos de mascota, donde el usuario podrá modificar los datos del formulario para posteriormente registrarlos pulsando el botón “Save”. La mascota quedará actualizada con los datos introducidos por el usuario. Tanto para la modificación como para la creación de mascota existen unos campos requeridos los cuales deberán estar completos para poder registrar correctamente la mascota.

Borrar mascota

En el caso de querer borrar una de las mascotas, desde su ficha, se pulsará el botón “Delete”. La aplicación solicitará confirmación del borrado de la mascota. Una vez confirmado, la mascota se eliminará del sistema y dejará de estar presente en la gestión de mascotas.

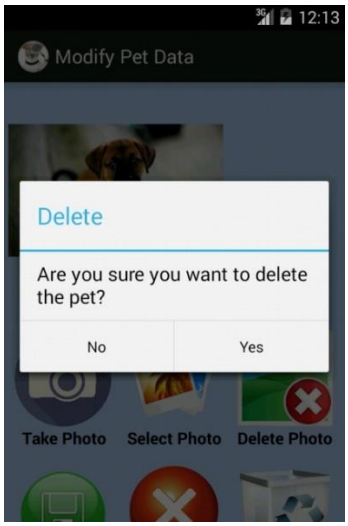

Ilustración 58 - Confirmación de borrado de mascota

Marcar mascota como perdida

El sistema realiza búsquedas sobre mascotas perdidas, en caso de haber perdido a nuestra mascota, querremos marcarla como perdida para que el sistema la tenga en cuenta en caso de una posible localización por parte de otro usuario. En este caso, sobre la ficha de la mascota, se marcará el recuadro "Lost" así como la fecha en la que la mascota fue perdida. Es muy importante tener en cuenta que el sistema necesitará de una imagen de la mascota para poder realizar posteriores búsquedas, por lo que deberá ser incluida dicha imagen.

El usuario podrá comprobar en la lista de gestión de mascotas cómo la mascota queda marcada como perdida.

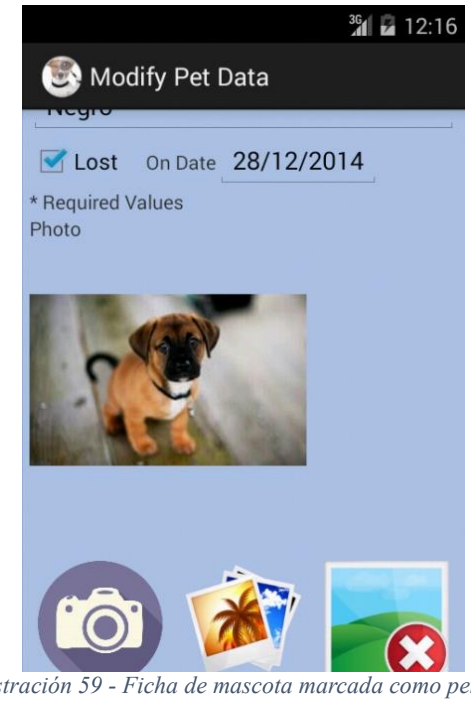

Ilustración 59 - Ficha de mascota marcada como perdida

Desmarcar mascota como perdida

En el caso de querer desmarcar una de las mascotas previamente registradas como perdidas, desde la ficha de la mascota se desmarcará el recuadro "Lost" y se grabará posteriormente, quedando la mascota desmarcada.

El usuario podrá comprobar en la lista de gestión de mascotas cómo la mascota ya no aparece marcada como perdida.

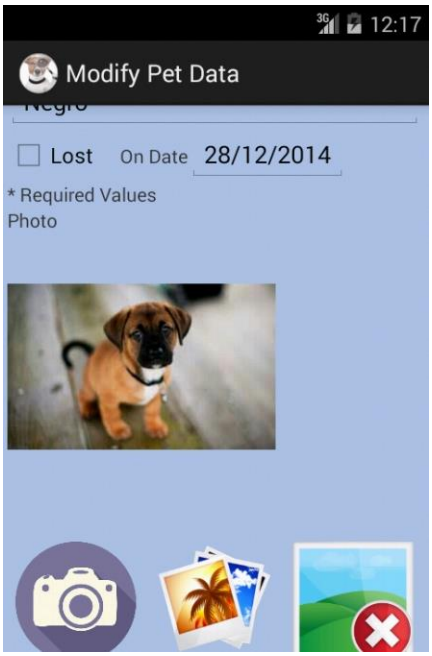

Ilustración 60 - Mascota desmarcada como perdida

GESTIÓN DE NOTIFICACIONES

Las notificaciones serán visualmente identificables mediante códigos de colores. La lista de notificaciones mostrará l alista de mascotas sobre fondo de diferente color en función del estado de la notificación:

- Verde: la notificación ha sido confirmada como válida.
- Rojo: la notificación ha sido rechazada.
- Blanco: la notificación está pendiente de validar.

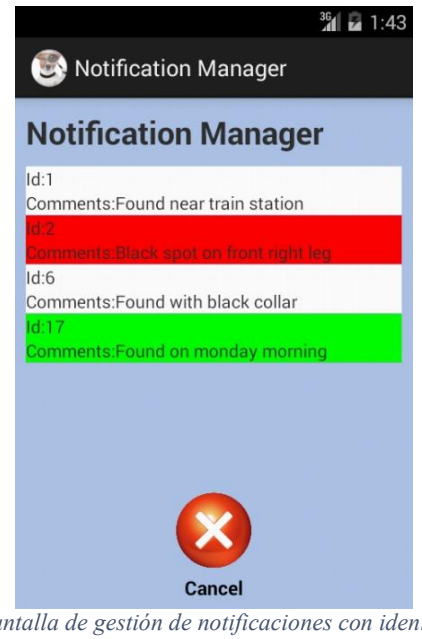

Ilustración 61 - Pantalla de gestión de notificaciones con identificación por color

Seleccionar notificación

Desde la pantalla de gestión de notificaciones, seleccionando una notificación de la lista, se accede a los datos de la misma.

Confirmar notificación

Desde la notificación seleccionada, se confirmará una notificación pulsando sobre la opción “Confirm”, guardando posteriormente la notificación. En este caso, se podrá ver la notificación como confirmada en la lista de notificaciones y la mascota relacionada dejará de estar en estado perdido.

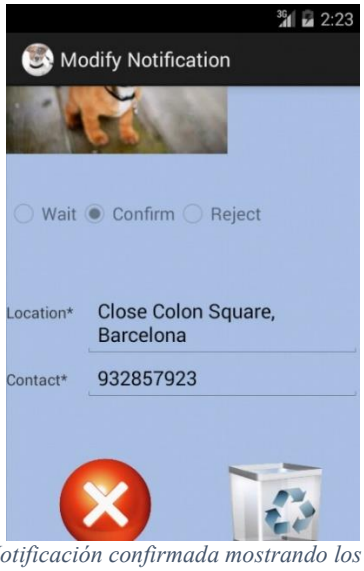

Ilustración 62 - Notificación confirmada mostrando los datos de contacto

Cancelar notificación

Desde la notificación seleccionada, se cancelará una notificación pulsando sobre la opción “Reject”, guardando posteriormente la notificación. En este caso, se podrá ver la notificación como cancelada en la lista de notificaciones.

Ilustración 63 - Notificación rechazada

Borrar notificación

Desde la notificación seleccionada, se borrará una notificación pulsando sobre el botón "Delete". La aplicación solicitará confirmación al usuario para borrar la notificación, la cual, dejará de estar presente en la lista de notificaciones del usuario.

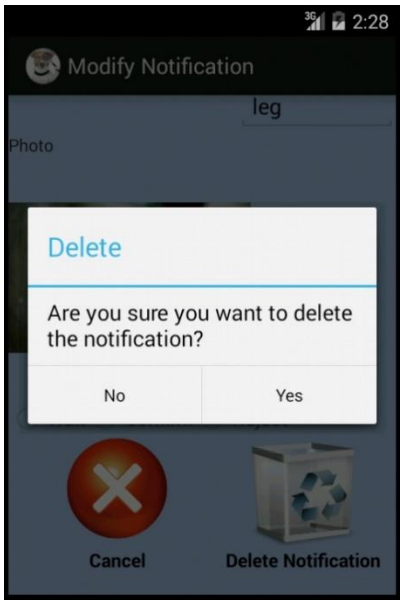

Ilustración 64 - Confirmación de borrado de notificación