TRABAJO FINAL DE CARRERA

TÍTOL DEL TFC: Estudio de la plataforma Android

TITULACIÓ: Ingeniería Técnica de Telecomunicaciones, especialidad Telemática

AUTOR: Judit Balaguero Peña

DIRECTOR: Antoni Oller Arcas

DATA: 24 de juliol de 2008

Título: : Estudio de la plataforma Android

Autor: Judit Balaguero Peña

Director: Antoni Oller Arcas

Fecha: 24 de Julio de 2008

Resumen

Este proyecto estudia y realiza una prueba de concepto sobre la nueva plataforma para dispositivos móviles creada por Google y actualmente en fase de desarrollo, Android.

Se estudian los conceptos más novedosos de Android, se aprende como desarrollar aplicaciones para él y se crea una aplicación que permita evaluar algunas de las prestaciones del sistema.

La aplicación es un principio para la creación de un sistema que utilicé el servició de localización, uno de los aspectos más interesantes de Android.

Title: Study about Android platform

Author: Judit Balaguero Peña

Director: Antoni Oller Arcas

Date: 24th July 2008

Overview

This Project looks and makes one concept test about the new platform mobiles devices created by Google and currently in development phase, Android.

It explores the most novel concepts of Android, learn how to develop applications for it and created one application that allow evaluate some of system ´s benefits.

The application is the principle for one designed system that will use de location service, one of the more interesting aspects of Android.

AGRADECIMIENTOS

Me gustaría dar las gracias a los compañeros del I2CAT, sobretodo a Xavi Calvo y Dani Rodríguez que han estado siempre para cualquier duda, debate o café.

Muchísimas gracias.

ÍNDICE

INTRODUCCIÓN

La tecnología móvil ha avanzado de una manera espectacular esta última década. Todo el mundo recuerda los primeros móviles de los 90, sus grandes dimensiones, con su antena exterior y el antiguo juego de la serpiente. Parecía imposible pensar que algún día la gente podría conectarse a Internet, escuchar música y mucho menos utilizar un móvil como cámara de fotos. Pues bien, hoy en día ya es algo común y al alcance de todo el mundo. Pasadas las novedades de tener un móvil que reprodujera mp3 o echara fotografías en VGA, ahora comprase un terminal implica pensar en los Gigabytes que tiene o los megapíxeles de la cámara de fotos, asumiendo ya lo que antes parecía futurista.

Entonces, la duda es: ¿Está ya todo inventado? Al menos las empresas se esfuerzan por hacer ver que no es así. Apple con su IPhone ha sorprendido al público con un teléfono con pantalla multitactil, que reproduce audio, video e incluso los actuales podcast. Nokia está pensando en móviles basados en la nanotecnología. Symbian que está preparando su primera versión "libre", la s60 y ahora Google, que no contento con ser el dueño del mercado de la información que circula por la red mundial, ha decidido conquistar también el mundo de la tecnología móvil con el primer sistema operativo libre y de código abierto.

Google, que todo lo hace a lo grande, ha reunido a empresas del sector móvil de todas partes para preparar la salida de Android al mercado a finales de año, después de múltiples especulaciones, rumores y dudas, éste ha lanzado su primera plataforma de desarrollo mostrando un sistema operativo nuevo, y ha lanzado un concurso con un premio de 10 millones de dólares para el diseñador de la mejor aplicación.

¿Qué puede tener Android que nos sorprenda? Además de ser de código abierto a lo que ya hemos hecho referencia, nos habla de servicios basados en la localización geográfica, es decir, programas que puedan decirnos donde se encuentran otras personas, o avisarnos si estas se encuentran cerca de nosotros.

El trabajo se ha estructurado de la siguiente manera: un aprendizaje, un diseño y una implementación. Esta memoria está dividida en cinco capítulos: el primer capítulo donde se habla propiamente de la plataforma Android mostrando los aspectos más destacables de ésta. El segundo capítulo donde se realiza un diseño de una aplicación basándose en las posibilidades estudiadas en el primer capítulo, un tercer capítulo que detalla la implementación y un cuarto y quinto que describen la planificación del proyecto, las conclusiones y los trabajos futuros.

El objetivo del proyecto es realizar un aprendizaje de Android, empezando por estudiar sus características y ver las novedades que ofrece para los usuarios y

los programadores. Y finalmente realizar una prueba de concepto que permita evaluar las posibilidades futuras de Android.

CAPÍTULO 1. PRESENTACIÓN DE ANDROID

En este capítulo se presenta Android, explicando que es, como se creo y destacando las novedades que ofrece. Se hecha un vistazo al interior de la plataforma detallando su arquitectura y explicando conceptos propios como la activity que resultan necesarios para poder desarrollar aplicaciones.

1.1. ¿Qué es Android?

Android es una plataforma de software de código abierto que incluye un sistema operativo para dispositivos móviles basado en Linux. Desarrollado por Google y por la Open Handset Alliance.

1.1.1. Breve historia.

La historia de Android es bastante corta y reciente, todavía no ha salido al mercado y tan solo disponemos de emuladores para poder hacernos una idea de cómo funcionará. Por lo que podemos decir que todavía es un proyecto en una fase de diseño muy avanzada.

El 5 de noviembre del 2007 Google presentaba oficialmente el proyecto Android y también la Open Handset Alliance, acabando así con semanas de múltiples rumores que corría por todos los medios. Android ya existía antes de que Google le echara el ojo. Antes de que ésta fuera comprada en el 2005, era una pequeña empresa de apenas 7 personas que se acaba de crear, con grandes prototipos e ideas, una de ellas el actual Android.

7 días más tarde Google lanzaba un entorno de desarrollo con una primera versión del emulador para que cualquier persona pudiera ver cómo iba a ser Android y lo más importante, que los programadores empezasen a desarrollar aplicaciones para Android.

En la última edición de la Mobile Word Congress (Febrero 2008, en Barcelona), Rich Miner, responsable de plataformas móviles en Google, aseguraba que habrán terminales desde 100 euros (68,5 euros) y que al menos, los primeros terminales no tendrán publicidad. Google informo que todavía están pensando en cómo introducir la publicidad a los usuarios en teléfonos móviles y que, además ésta sea útil para ellos.

Todavía no se tiene una fecha exacta para su lanzamiento, Se rumorea que los primeros teléfonos con Android serán saldrán al mercado a finales del 2008 y las primeras marcas serán LG, Motorola i HTC.

1.1.2 Handset Alliance

Creada y fundada por Google, la Open Handset Alliance nació para el desarrollo e implementación de Android en terminales móviles. Actualmente está formada por más de 30 compañías o empresas del sector móvil. Sus objetivos son acelerar la innovación en las comunicaciones móviles y ofrecer a los consumidores una experiencia más rica, menos cara y mejor.

Con estos propósitos, dentro de las empresas que la forman hay fabricantes de terminales móviles (Samsung, LG, Htc y Motorola), fabricantes de componentes (Texas Instruments, Intel, Nvidia etc.), de software (PV, EBay, Esmertec etc.), operadores de todo el mundo (T-Mobile, Italia Telecom, China Mobile etc.) y también empresas de comercialización, podemos ver que la representación española se encuentra en Telefónica Móviles. El hecho de que hayan importantes multinacionales dentro de la alianza nos hace ver que hay un gran equipo detrás de este proyecto que ya empieza a hacerse notar incluso antes de su entrada en el mercado.

1.1.3. Filosofía Open Source

Android ha nacido con una filosofía de código abierto, esperando que programadores de todo el mundo contribuyan de manera libre al constante desarrollo del sistema operativo. Además Google también dejará libertad a los fabricantes para que desarrollen aplicaciones para sus teléfonos y, que la licencia sea de código abierto, no descarta la opción de hayan empresas que cobren por los programas que desarrollen. De hecho cualquier operadora puede vender terminales con Android, siempre que ésta cumpla lo siguiente:

- Precio bajo
- Acceso abierto a Internet
- Libertad del usuario para descargar cualquier aplicación

Para animar a los programadores a desarrollar para Android, Google ha ofrecido 10 millones de dólares a las mejores aplicaciones creadas. Esto nos muestra la estrategia de Google: recopilar un montón de aplicaciones nuevas y atractivas para impresionar al mercado y aumentar la demanda de terminales.

Para que los programadores puedan desarrollar, Google dispone de una página Web para que los usuarios puedan descargarse el SDK, bajarse manuales, ver ejemplos y leer instrucciones para que estos puedan familiarizarse con la plataforma. Algo realmente lógico si quieren que los programadores se predispongan a programar de manera voluntaria.

1.1.4. Para todo tipo de terminales

Aunque Android se podrá instalar en cualquier terminal parece ser que irá orientado para los SmartPhone, donde el uso de Internet es más grande que en los teléfonos móviles ya que todas las aplicaciones de Google son accesibles a través de Internet. De todas maneras se quiere optimizar al máximo el uso de recursos de un móvil para que Android pueda funcionar en terminales poco potentes, eso nos permitiría tener un teléfono con sistema operativo Android a un precio mucho más bajo que el de un SmartPhone.

1.1.5. La competencia de Android

Los principales rivales de Android son Windows Mobile, Palm OS y el conocidísimo Symbian de los Nokia. Navegando por Internet podemos ver que con quien más se le compara es con el IPhone y en segundo lugar (y más lógico) con el sistema operativo Symbian. Fabricantes como Apple, Microsoft, Nokia, que apuesta fuerte por Symbian, o Palm y que no forman parte de esta alianza tampoco descartan formar parte de ella en un futuro. Seguramente son conscientes de la amenaza que puede suponer el lanzamiento de Android.

Después del lanzamiento del IPhone de Apple se esperaba que Google contraatacase con modelo de teléfono con un software específico (como el IPhone). Pero han dejado claro que ellos se van a centrar en la parte del software i dejan la fabricación de terminales para las empresas de la Open Handset Alliance.

1.1.6. Resumen de las características técnicas de Android

Lo primero a lo que se ha de hacer referencia en Android es que está basado en Linux, es decir, todos los servicios base (gestión de drivers, memoria, seguridad) están basados en el sistema operativo de código abierto.

Dalvik es el nombre de la maquina virtual donde se ejecutan las aplicaciones. Ésta está optimizada para requerir poca memoria y poder usar varias instancias simultáneamente sin que el dispositivo se ralentice. Los ejecutables pasan a tener la extensión .dex, una versión optimizada de los .class y el lenguaje en el que se programa es puramente Java. Esta conversión podría ser una estrategia de Google para evitar conflictos con Sun por la licencia de la máquina virtual, pudiendo así poder modificarla y manteniendo igualmente el lenguaje java, que ya resulta conocido por los programadores.

El motor de navegación es el Webkit, el mismo que utilizan los Mac o los IPhone. Éste es de código abierto y actúa como base para varias aplicaciones que hay actualmente en el mercado, la más famosa el navegador Safari de Apple que hoy en día podemos encontrar también para Windows).

Android utiliza SQLite para el almacenamiento estructurado de datos. SQLite ya viene incluido en el SDK y se puede acceder plenamente a sus clases. También será posible la utilización de otras bases de datos como Perst o incluso utilizar las clases de almacenamiento de datos de la API de Android sin tener que hacer uso de SQLite.

Otras características interesantes será el soporte a los formatos más comunes de archivos multimedia, un framework que permite la reutilización de componentes y gráficos optimizados, provenientes de librerías 2D y 3D. Además de aquellos recursos que dependen del terminal como el Bluetooth, 3G, Wifi, cámara y GPS entre otros.

1.2. Arquitectura de Android

La arquitectura de Android está formada por capas de software donde cada una puede utilizar los servicios de la capa inferior. Empezando por la capa inferior tenemos el conjunto de drivers basados en Linux, esta parte no es pública. Un nivel más arriba tenemos un conjunto de librerías que no son accesibles directamente si no a través del nivel superior a ésta, el Framework de aplicaciones, que junto a la capa de aplicaciones son totalmente públicas y los usuarios pueden acceder libremente.

Fig. 1.1 **Representación de la arquitectura de Android.**

Applications. Un conjunto de aplicaciones base de Android, entre las que encontramos un navegador Web, un cliente de email, un calendario etc. Todas hechas en lenguaje Java.

El **Application Framework** es una base para las aplicaciones donde los desarrolladores tienen acceso completo. Pensado para la reutilización de componentes, es decir, una aplicación puede coger funcionalidades de otra creada anteriormente para su desarrollo. Éste incluye:

- **Telephony** manager: gestor de hardware del teléfono.
- **View system** Conjunto de vistas para poder desarrollar una aplicación Buscadores, cajas de texto, botones etc. Algunos ejemplos son la "vista de mapas" y la "vista de navegadores". Estas las podemos usar en nuestra aplicación.
- **Content providers**. Para datos que son compartidos entre varias aplicaciones, como por ejemplo la agenda de teléfono.
- **Resource Manager.** Administrador de recursos que permite acceder a recursos como Strings, gráficos, archivos de layout…
- **Notification Manager**. Administrador de notificaciones para mostrar alertas. Las aplicaciones pueden añadir eventos en una barra de notificaciones.
- **Activity Manager.** Administrador de actividades, éste maneja el ciclo de vida de las aplicaciones y la navegación entre ellas.
- **Location Manager.** Servicio de localización. Permite al móvil recibir avisos, notificaciones, eventos etc., de un lugar específico o por nuestra localización actual. Una posibilidad de las muchísimas que se podrían dar con esta API sería que cada vez que nos acercásemos a un cine, recibamos las novedades de éste en nuestro teléfono.
- **Servicio XMPP.** Envío de mensajes para aplicaciones entre terminales Android. Se podría utilizar en juegos multiusuario por ejemplo.

La capa **Libraries** está formada por un conjunto de librerías escritas en C/C++. Todas expuestas a los desarrolladores a través del Framework de aplicaciones. Entre las más importantes encontramos SQLite disponible para todas las aplicaciones o las Media Libraries que nos permiten poder reproducir los archivos de audio, video y fotografía más populares como el mpeg4, el JPEG, etc.

El **Android Runtime** está compuesto por el núcleo de librerías y la máquina virtual Dalvik. Aquí disponemos de un conjunto de librerías que incluyen las funcionalidades que solemos encontrar en las librerías básicas de Java. Como

ya se ha explicado anteriormente, esto no quiere decir que la máquina virtual sea Java, pero si su lenguaje usado para programar.

En la parte más inferior tenemos la **Kernel de Linux**. Basado en el núcleo de Linux 2.6 y base de la pila de software del sistema, se encarga de las funciones más básicas: gestión de drivers, seguridad, gestión de memoria, administración de procesos etc.

1.3. Bloques de construcción

A la hora de desarrollar aplicaciones, estas se construyen basándonos en bloques de construcción básicos, entre los cuales se muestran los más importantes. No todas las aplicaciones tienen que necesitarlos todos y estos pueden relacionarse entre sí. Los principales son:

1.3.1. Activity

Éste es, sin duda, el bloque de construcción más utilizado. Una definición simple de ésta sería la de una tarea que se lleva a cabo en la aplicación y tienen una interacción con el usuario. Estas se implementan extendiendo de la clase “activity” y cada una de ellas tiene su proceso de vida propio. Típicamente tendremos una activity como punto de entrada a nuestra aplicación

La mayoría de las aplicaciones tienen pantallas múltiples y cada vez que se accede a una nueva la anterior es retenida y guardada en una pila. Gracias a esta pila el usuario puede navegar “hacía atrás” por las activities que ya ha tenido activas. Si Android lo considera oportuno puede priorizar y quitarlas de la pila por cuestiones de memoria. Tomando por ejemplo la mensajería instantánea, cada pantalla seria una activity diferente y una calculadora sería un programa con una activity solamente.

1.3.2. Intent y IntentReceiver.

Intent es una solicitud para realizar una acción. Éste sería como “hacer un intento de algo” o decir “quiero hacer tal cosa”. Es un evento de la aplicación genérico creada por ella. Puede ser un evento predefinido ya por Android como “quiero llamar”,” quiero abrir el navegador” o “quiero enviar un mail”. Al producirse el Intent Android trata de buscar la aplicación más apropiada para éste. Éste es necesario para movernos por las pantallas de una aplicación.

Un IntentReceiver hace que ejecute nuestra aplicación al producirse un evento. A diferencia del “Intent” anterior, nuestra aplicación no tiene que estar corriendo

para que ejecute sus IntentReceiver, es decir, el usuario no tiene porque estar interactuando con el programa en ese momento.

1.3.3. Service

El equivalente a un Daemon o un Servicio de Windows. Se ejecuta en background sin necesidad de interactuar con el usuario ni de desplegar una pantalla y posiblemente por largos periodos de tiempo.

1.3.4. Content Provider

Un Content Provider se usa cuando los datos de una aplicación se compartirán con otras aplicaciones. Esta clase implementa un conjunto de métodos estándar para las aplicaciones que comparten datos almacenen y extraigan la información que contiene el Content provider.

1.3. Ciclo de vida de las aplicaciones

En Android es el sistema el que determina el ciclo de vida de una aplicación y no esta última. Es decir, éste determina su duración en base a las partes de la aplicación que están ejecutando en ese momento, la importancia de estas para el usuario y la memoria disponible. Ésta es una característica importante y poco usual.

Android clasifica los procesos de manera jerárquica según la importancia que tengan en un momento determinado para el sistema:

- **Foreground process**: éste es un proceso en primer plano con el que el usuario estaría interactuando, éste hospeda una activity y el sistema lo eliminaría como última opción.

- **Visible process:** también es un proceso que hospeda una activity pero que no está en primer plano, aunque si visible. Un ejemplo sería una pantalla que lanza un cuadro de dialogo, interactuamos con el cuadro pero seguimos viendo la pantalla detrás de éste. Este tipo de proceso solo será eliminado en estados de memoria muy críticos igual que el Foreground process.

- **Background process**: proceso que hospeda una activity pero no está visible en pantalla, es decir que si lo eliminamos no tiene una repercusión directa con el usuario.

- **Empty process**: un proceso que no hospeda ningún componente visualmente activo. El sistema los eliminará con frecuencia y puede mantenerlos vivos si hay memoria suficiente para mejorar el tiempo de activación de otro componente de esa aplicación.

Las activities, los Intent Receiver o los Services impactan de una manera importante en el tiempo de ejecución de las aplicaciones, posibles errores en el uso de estos pueden hacer que procesos importantes para el usuario finalicen antes de tiempo.

1.4. Ciclo de vida de las activities y tratamiento.

Muy parecido al ciclo de vida de las aplicaciones, aunque solo tratando de los componentes activities. A diferencia de las aplicaciones, para hablar del ciclo de vida de una activity tenemos que hacer referencia al “stack de activities” del sistema que apila exclusivamente activities que están ejecutándose.

Cuando se cambia de una activity a otra, la nueva activity que es creada se colocada en la parte superior de la pila y pasa a ser la activity que se ejecuta en ese momento, mientras que la activity anterior queda por debajo de ésta en la pila, quedando así en background hasta que la principal no sea eliminada de la fila.

Las activities tienen cuatro estados según su interacción con el usuario:

- **Activa o en ejecución**. Este estado se da cuando la activity está en la parte superior de la pila, es la última activity creada por la aplicación y será la última que el sistema intente eliminar.

- **En pausa**. Este estado se da cuando una activity deja de ser la principal pero todavía queda visible en pantalla, esto es posible en activities transparentes que dejan ver la inferior. Ésta continúa viva manteniendo todo la información del usuario.

- **Parada**. A diferencia del el estado “pausa” la activity en este estado no es visible en pantalla. Sigue manteniendo la información del usuario pero frecuentemente será eliminada por el sistema para liberar memoria.

- **Eliminada,** puede ser porque el sistema necesite memoria o simplemente porque la aplicación lo ha ordenado.

Podemos ver que el ciclo de vida de una aplicación está relacionando con el ciclo de vida de una activity, dando prioridad a los procesos que contienen una. Como conclusión Android considera como imprescindibles aquellos procesos con los que el usuario está interaccionando y hace todo lo posible para que no le afecten estados críticos de memoria que se puedan dar.

Para manejar el estado de las activities tenemos encontramos los siguientes métodos incluidos en la clase activity: OnCreate(), OnDestroy(), OnPause(), OnStop(), OnFrezze(), OnResume() , OnRestart() y Finish().

Cada vez que se lanza una aplicación o pasamos de una activity a otra, ésta es creada por el sistema, en cambio si se navega hacia atrás ésta puede ser restablecida (recuperando toda la información del usuario) o volviendo a ser creada de nuevo, también se pueden poner en pausa, en ambos casos ésta pasa a estar en segundo plano.

Una activity puede estar finalizada por la aplicación, o por el sistema en caso de que éste necesite memoria.

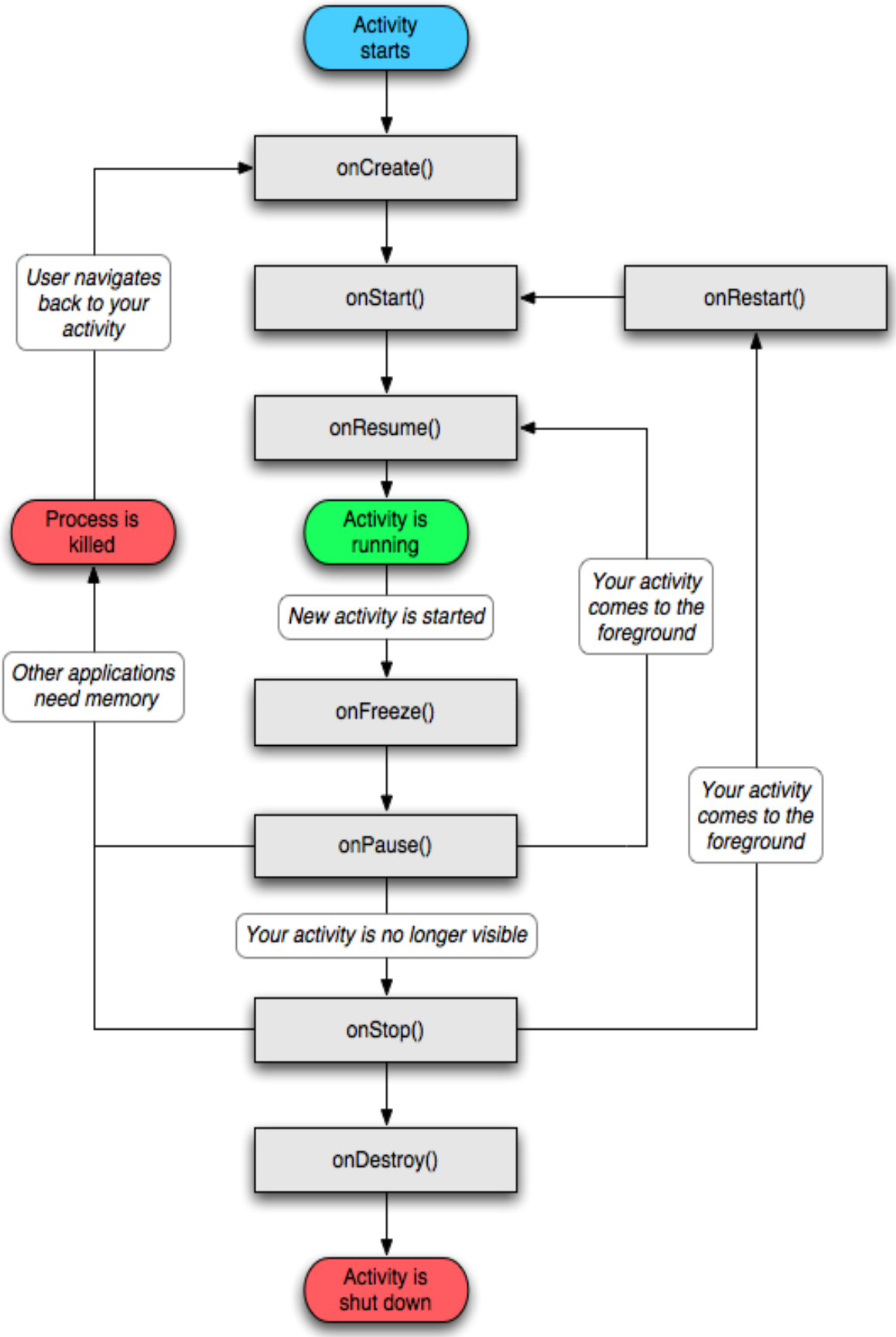


Fig. 1.2 Esquema del ciclo de vida de una Acitivity.

1.4.1. Cambiar de Activity

Seguramente la aplicación que tendrá crear tendrá varias pantallas y intuitivamente se piensa que desde una clase se podrá pasar de una pantalla a otra solo poniendo las instrucciones adecuadas, lo más lógico sería pensar en SetContentView, una instrucción que indica que XML o View mostrar. Pues bien, de esta manera el programa en Android no funcionaría. En Android cada pantalla que tenga la aplicación es una activity diferente y para poder pasar de una a otra se utilizan los Intents. Como ya se ha dicho anteriormente un Intent utilizando para abrir una activity es como decir “quiero inicializar esta pantalla” y Android ya tiene este método predefinido para hacerlo de la mejor manera. Igualmente todas las activities tendrán que estar declaradas en el AndroidManifiest. Cada vez que pasemos a una activity nueva la anterior será retenida y ésta quedará en Background pero a diferencia de un proceso ésta no seguirá activa. Hay que comprender muy bien el ciclo de vida de estas para que la aplicación no abuse de memoria.

1.4.2. Paso de datos entre activities.

De una manera muy parecida al cambio de una activity con Intents también se realiza el paso de información entre estas. Antes de llamar al Intent se crean bundles donde se introducen los datos a traspasar y, en la activity donde se reciben los datos, se crea otro Bundle que los recoja. Resulta muy sencillo ya que son dos líneas de código.

También puede darse el caso que se quiera devolver un valor al finalizar una activity. Esto también es posible usando los métodos startSubActivity(Intent, int) y onActivityResult(int, int, String), El primero creara la activity y los valores que ésta devuelva serán recogidos en el segundo método.

1.5. Versión m5 de el emulador.

La última versión lanzada para Android es la m5-RC14, de un total de cuatro versiones disponibles. Esta última todavía continua teniendo limitaciones importantes a tener en cuenta para el desarrollo de aplicaciones, quizás la más importante es que no soporta Bluetooth.

Además de emular los programas creados por los usuarios, también se puede acceder a otras aplicaciones, navegar por Internet, escuchar música, ver videos, almacenar datos en la agenda de contactos etc. También contiene un numeroso paquete de elementos de desarrollo para que el programador pueda observar su funcionamiento y coger ideas y un programa que transforma los archivos .class en .dex que es el usado por la máquina Dalvik,

El emulador viene incorporado con el entorno de desarrollo cuando nos lo descargamos de Internet. Puede ser ejecutarlo directamente o, lanzado cuando compilamos nuestras aplicaciones para Android con el Eclipse. Puede que incluso sea necesario meterse dentro del sistema para emular señales de GPS o restablecer los parámetros del usuario.

Entre las limitaciones más importantes como la del bluetooth ya nombrada, el sistema permite simular llamadas entrantes, la utilización de una tarjeta de memoria externa o incluso la interrupción de un programa si se ha recibido un SMS.

1.6. Plug-in de Android para Eclipse

Para crear aplicaciones para Android, Google ha creado un Plug-in perfectamente adaptado para Eclipse que se puede descargar gratuitamente desde la página oficial de Android. Al instalarse el plug-in y crear un proyecto nuevo vemos que éste está estructurado de la siguiente manera:

- Un directorio (src/) donde estará el package con la clase principal que hayamos creado (tendrá el código genérico que crea Android) y el archivo R.java,

  R.java es una clase que se va modificando sola. Cada vez que creamos una variable ésta aparece reflejada en el R.java. Se podría decir que es un índice a todos los recursos declarados en el proyecto y el motivo de éste es que sea más rápido localizar las referencias de una aplicación que se estén buscando.

- **AndroidManifiest.XML.** Donde se declaran las activities y se indica que Intents realizarán. Éste es un archivo con extensión XML que dice al sistema que es lo que hay que hacer, con qué capacidad y qué componentes utiliza. Cuando se genera un nuevo proyecto, este archivo aparecerá automáticamente y medida que se vayan creando activities, estas se tendrán que ir declarando en el Manifiest.

  Si se observa la primera activity que se crea junto con un proyecto nuevo aparece en el Manifiest como la pantalla principal a la aplicación (Figura 2.1).

  Los intentos son descritos también en este archivo, estos nos dicen cuando y como una activity es creada.

- Un directorio de carpetas (res/) que contiene las carpetas drawable, layout y values. Donde drawable contendrá las imágenes (en formato .PNG) que utilice nuestra aplicación, values contendrá un archivo llamado strings.XML que definirá cadenas de caracteres constantes y el subdirectorio layout, también en formato .XML donde definiremos los

elementos que forman la interfaz gráfica de nuestra aplicación. Es posible crear nuevas carpetas dentro del directorio para poner otro tipo de contenidos que necesitamos.

- El directorio Android Library como ya dice su nombre se encuentran las librerías especificas de Android.

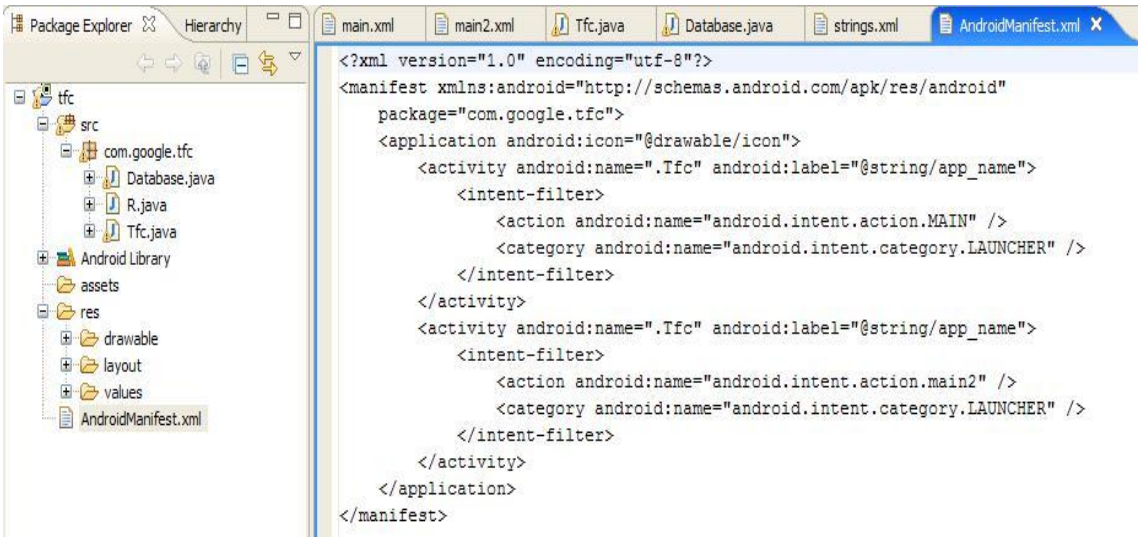


Fig. 1.3 Ejemplo de un proyecto en el Eclipse.

1.7. XML-based layout

Android plantea un modelo alternativo para la construcción de interfaces gráficas, llamado “XML-based Layout”. En Android las UIs pueden ser construidas de dos maneras, directamente invocando operaciones a las API desde las clases del programa o a través de relacionar estas con archivos XML.

De la primera manera, puede resultar complicado realizar una modificación de una UI, además la conexión de cada elemento con las Views de manera incorrecta puede ocasionar errores en la creación de las activities y demasiada pérdida de tiempo debugando problemas de creación de interfaz.

Por ello Android separa la creación de UI en archivos XML donde cada etiqueta del árbol es un elemento del tipo View representado dentro de la pantalla del programa, con sus atributos correspondientes que lo personalizan. De esta manera resulta mucho más rápido y sencillo crear una UI. Este modelo está basado en el diseño de páginas Web, donde se separan el diseño y las manipulación de los datos.

1.8 Servicio de localización geográfica.

Es una de los aspectos más atractivos del sistema operativo y que dará de hablar en cuanto salga al mercado. El sistema de localización geográfica usa la propia red de telefonía para localizar un terminal, haciendo que haya una permanente comunicación con las antenas que dan cobertura a éste. Hasta ahora existía este servicio pero resulta demasiado caro para particulares.

1.8.1. Conseguir coordenadas de una ubicación

El servicio de localización lo primero que intenta ofrecer son coordenadas GPS para móviles que lo soporten, por lo contrario el móvil intenta aproximar coordenadas utilizando antenas celulares y en el caso que éste tampoco estuviera disponible finalmente el terminal buscaría la conexión a una red Wi-fi.

CAPÍTULO 2. DISEÑO DE LA APLICACIÓN

El capítulo dos presenta el diseño de la aplicación, mostrando un sistema completo basándose en la localización geográfica y detallando sus funcionalidades internas y externas.

2.1. Presentación de la aplicación.

Se ha creado una aplicación para Android con la finalidad de poder evaluar algunas de las posibilidades que éste nos ofrece.

Esta aplicación ofrece al usuario consultar a través de su teléfono móvil si tiene algún punto de interés próximo a él (cines, centros comerciales, gasolineras, hoteles etc.).

Por ejemplo, una persona está cerca de un cine pero no encuentra la posición exacta, a través de su teléfono puede saber los cines que hay en un radio de un par de kilómetros, comprobar que está la película que quiere ver, incluso poder ver el tráiler de ésta si está disponible y que el teléfono móvil le muestre camino para llegar hasta él.

También podría darse el caso, de que esta misma persona quisiera planear un fin de semana lejos de la ciudad donde vive, podría buscar los hoteles cerca del destino, consultar la disponibilidad, comprobar que restaurantes hay cerca y reservar mesa, todo eso desde el teléfono móvil, sin necesidad de tener un ordenador delante.

Dando un paso más, podría darse el caso de que tuviéramos configurado nuestro teléfono para que nos avisase cuando estemos cerca de un restaurante de comida tailandesa o cuando tengamos cerca una gasolinera.

Con todo esto, se ha hecho un estudio de las posibilidades que nos ofrece este sistema operativo y se empezado a construir una aplicación que pueda hacer posible todos los casos descritos anteriormente, empezando por el principio más básico: aprender a programar para Android.

2.2. Funcionalidades del terminal

El programa permite al usuario:

- Permite al usuario introducir una posición y obtener los puntos de interés (Cines, hoteles, gasolineras, centros comerciales, etc.) que están dentro de un radio de distancia escogido también por el usuario. Por ejemplo, el usuario solo quiere ver aquellos sitios que estén como mucho a unos dos kilómetros de distancia.

- Almacenar la información de los puntos de interés dentro de una base de datos en el teléfono móvil. El usuario puede repetir consultas en poco tiempo y poder extraer esta información directamente de su móvil para no tener que volver a hacer una petición al servidor.

- Puede darse el caso, que un usuario solo quiera buscar un punto de interés concreto, puede realizar la búsqueda por su nombre (por ejemplo: la maquinista), por su posición geográfica o por el tipo de establecimiento.

- El terminal permite mostrar la ubicación de un punto de interés en un mapa y desplazarse a través de él.

- Cuando un punto de interés tiene un contenido multimedia, el usuario es avisado, permitiéndole reproducirlo o no.

2.3. Funcionalidades externas.

Todas estas funcionalidades hacen referencia al usuario portador del teléfono móvil. Aunque el sistema para que funcione necesita de una parte externa que mantenga y actualice los contenidos que el usuario querrá pedir. En la siguiente figura, a la derecha observamos la parte del sistema que está fuera del terminal, el servidor. El servidor contiene la base de datos que almacena todos los puntos de interés. Esto requiere que haya alguien que se encargue de crear los puntos de interés y introducirlos en la base de datos, que mire que la información que contiene ésta no se quede obsoleta e introducir los contenidos multimedia a los que hacen referencia los puntos de interés, siempre que estos no sean enlaces a otras páginas Web.

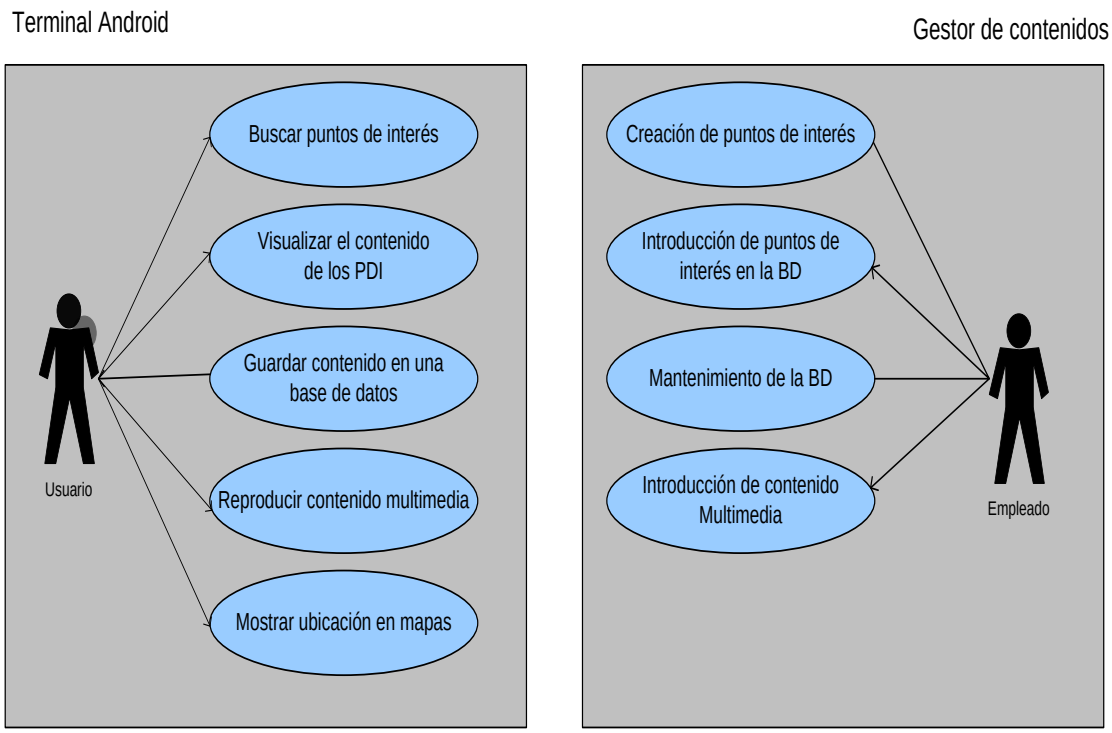


Fig. 2.1 Funcionalidades internas y externas.

CAPÍTULO 3. IMPLEMENTACIÓN DE LA APLICACIÓN

En este capítulo, se explica como se ha creado la aplicación, las tecnologías que se han utilizado , la estructura interna y los procesos que se siguen para llevar a cabo la prueba de concepto.

3.1. Objetivos de la implementación.

Estos son los objetivos, hablando ya más específicamente, que permiten realizar las funcionalidades descritas en el capítulo anterior.

- Realizar la descarga de contenidos: hacer una petición Http a un servidor externo y descargarnos un XML, en nuestro caso una vez hecha la descarga manejaremos los datos con un SaxParser.

- Almacenar los datos en el teléfono: poder manipular la base de datos SQLite, insertar, mostrar todo el contenido y realizar consultas por parámetros.

- Limitar la descarga de contenido: a la hora de hacer una petición Http, solo nos bajaremos parte de la información total que haya en el servidor, es decir, nos descargaremos todos los PDI que estén dentro de nuestro cuadrante calculado al introducir nuestra posición en la aplicación. Esto se hará a través de peticiones a diferentes Url.

- Notificación de archivos multimedia: Dentro de las etiquetas XML de los PDI, crear un campo que contenga una URL a un archivo multimedia, si el PDI mostrado contiene información en esta etiqueta, mostrar algún tipo de notificación por pantalla (lanzar un pop-up por ejemplo).

- Reproducción de archivos multimedia: se implementará una clase especial de las librerías de Android para reproducir por pantalla audio o video.

- Utilización de mapas. El usuario pasará unos parámetros (coordenadas x y) a una clase especial llamada MapActivity para desplegar una adaptación de los mapas de GoogleMaps.

3.2. Partes específicas de Android tratadas.

Estas son los conceptos concretos de Android que se han tenido que asimilar para poder realizar el programa.

- Creación de una Interfaz de usuario con los elementos que proporciona la plataforma.
- Creación, manipulación de bloques de construcción tan importantes como las activities y los Intents. También como se relacionan entre ellos y su ciclo de vida.
- Paso de datos entre activities.
- Uso de la base de datos SQLite, aunque ésta no sea una base de Datos exclusiva de Android.
- Utilización de las clases Map activities.
- Compresión de la estructura y los componentes que forman un proyecto en Android tales como AndroidManifiest.XML, R.java, directorio de layouts, etc.

3.3. Arquitectura de servicio.

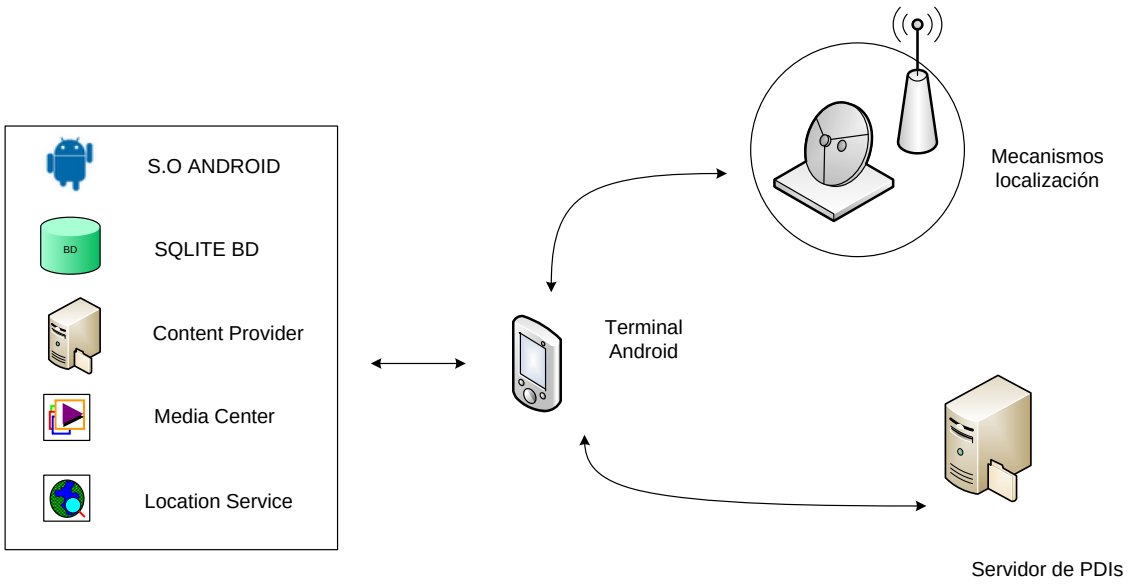


Fig. 3.1 Sistema completo.

Emulador:

La versión utilizada del emulador es la m5, que presenta bastantes diferencias respecto su antecesora.

El terminal estará dotado de:

- Sistema operativo Android. Disponemos de la versión todavía incompleta de éste.
- Base de datos SQLite. La base de datos seleccionada es SQLite, versión reducida de MySQL que utiliza este mismo lenguaje. Ésta ya viene incorporada en el sistema operativo.
- Proveedor de Contenidos. Los PDI serán gestionados dentro del terminal a través de una clase de java.
- Media Center. Conjunto de librerías especificas de Android para la reproducción de archivos multimedia. Actualmente bastante limitado por la versión del emulador m5.
- Servicio de localización. Aunque Android proporciona un servicio de localización, el utilizado para la aplicación es una clase más sencilla para determinar la actualización de la base de datos.

Servidor externo:

El servidor externo no dispondrá de ningún tipo de inteligencia para esta prueba de concepto, tan solo almacenará archivos XML, pero se tiene en cuenta los componentes que tendría en el sistema completo.

Otras tecnologías utilizadas:

- XML: formato de los puntos de interés.
- SAX: tratamiento de los datos descargados por el usuario.
- HTTP: protocolo de transferencia Web.

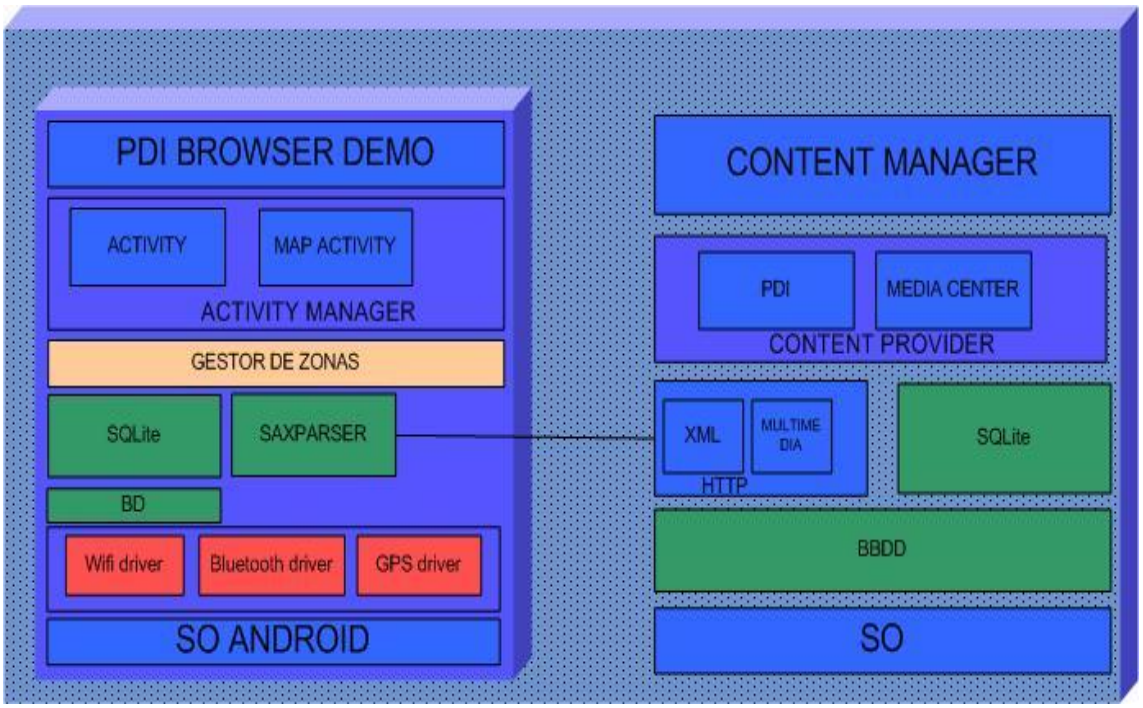


Fig. 3.2 Bloques internos del sistema

3.4. Estructura del programa

El programa está estructurado en tres conjuntos de clases y una clase especial que las relaciona. El primer conjunto agrupa todas las clases con las que el usuario interacciona y tienen una interfaz en la pantalla del teléfono. Estas clases son las activities y las Map activities. El segundo grupo corresponde a las clases que realizan las peticiones http y manejan los datos que obtienen de los XML descargados. Finalmente tenemos las clases que manejan la base de datos SQLite. Estos tres conjuntos se relacionan a través de clases auxiliares para obtener, buscar e insertar la información que se utilice en cada ejecución del programa.

La clase especial gestor de zonas, es la que controla la actualización de la base de datos y pide la descargas de los ficheros necesarios.

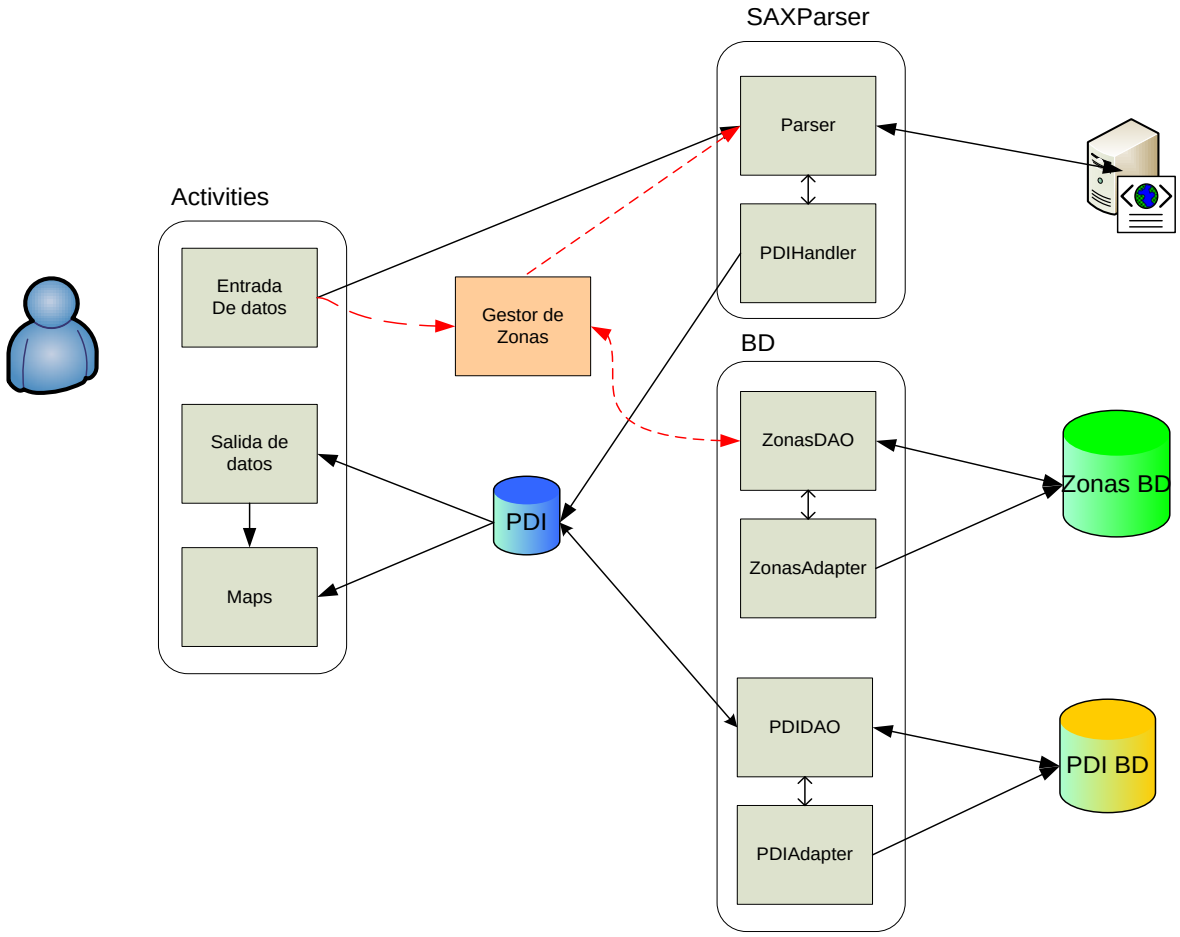


Fig. 3.3 Representación de la estructura del programa y su relación.

3.4.1. Funcionamiento general

El usuario accederá a la aplicación desde el terminal, y a través del menú principal podrá escoger una de las opciones disponibles.

Cuando el usuario quiera ver si hay puntos de interés cerca de él, escogerá la opción “Buscar PDI cercanos” e introducirá sus coordenadas en las activities de entrada de datos..

El programa contiene una clase especial que es el Gestor de Zonas, que se encarga de comprobar la base de datos y determinar que peticiones son necesarias realizar. Para ello la aplicación contará con dos base de datos, una que guarde los puntos de interés y otra la identidad de los archivos XML descargados. Cada XML corresponde a una región geográfica diferente definida para esta versión demo. Cada base de datos tiene sus propias clases con sus métodos.

Una vez se realiza una descarga de un fichero XML del servidor, éste es tratado con un SAXParser y su información es guardada en una base de datos

SQLite. Para las posteriores consultas siempre se comprobará la actualización de la base de datos, evitando así tener que volver a hacer peticiones http cada vez que quiera hacer una consulta con los mismos resultados. De esta manera permite al usuario reducir el número de peticiones a Internet.

El usuario podrá ver los puntos de interés próximos a su zona y mostrar la localización de estos usando una adaptación de GoogleMaps para Android y si además éste contiene algún tipo de contenido multimedia poder reproducirlo en el emulador.

3.5 Búsqueda de puntos de interés

En el siguiente diagrama de flujo se muestra el proceso que sigue una petición de los puntos de interés.

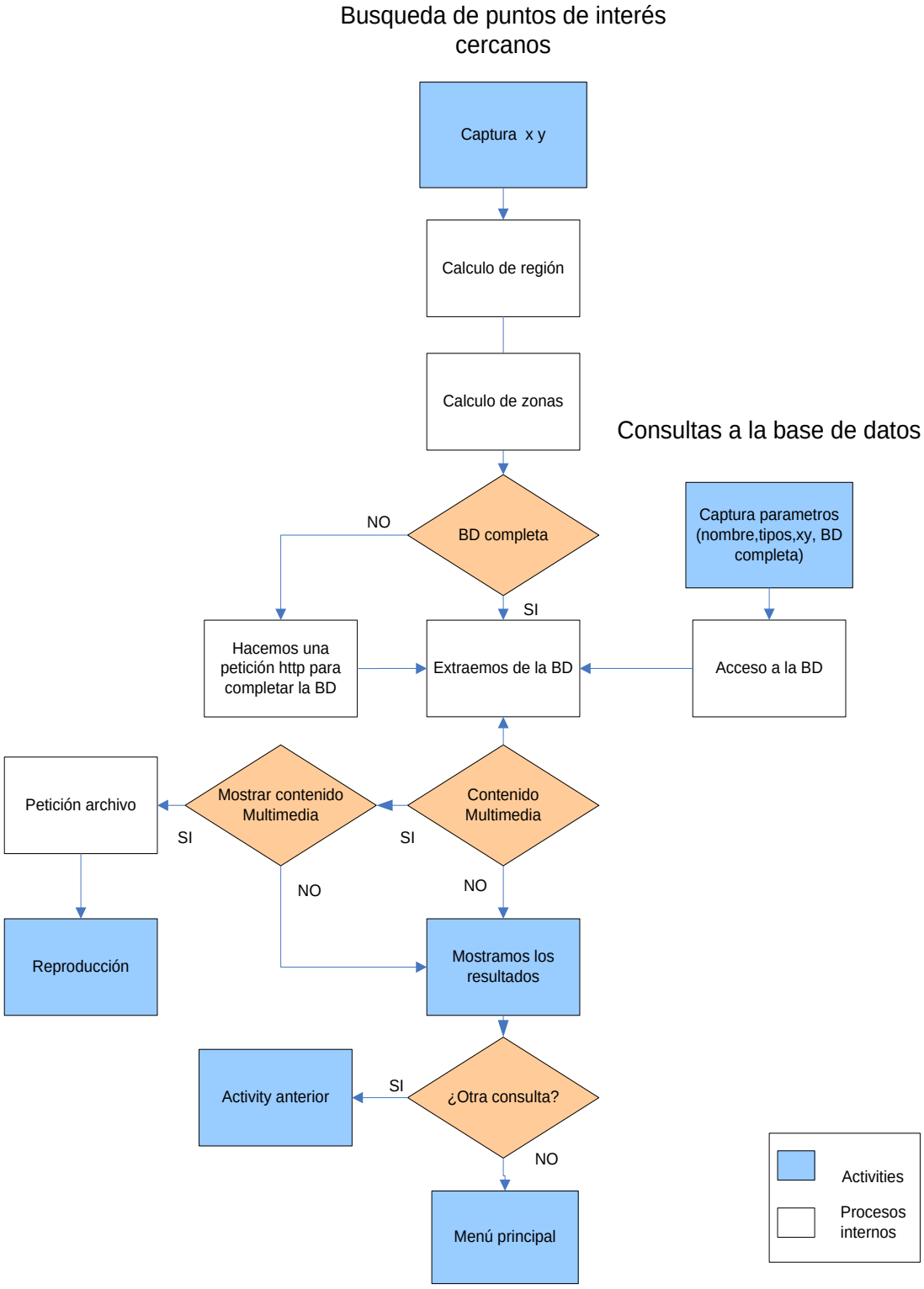


Fig. 3.4 Proceso en la búsqueda de un punto de interés.

3.5.1. Calculo de la Región para mostrar puntos de interés.

A la hora de querer realizar una petición de PDI el programa lo primero que consultará será nuestra posición geográfica y si se tiene la información correspondiente en la base de datos. Si la información ya se encuentra en la base de datos simplemente la mostrará por pantalla al usuario. Si ésta en cambio requiere pedir o actualizar su contenido realizará peticiones HTTP para poder completar la base de datos con la información que le falta y mostrarla posteriormente.

En el caso que la información de del dispositivo móvil estuviera incompleta, tendríamos que determinar qué zonas son las que hay descargar, ya que la información está dividida en varios archivos XML subidos al servidor. Esto es para no llenar la memoria del móvil con información innecesaria de otras zonas y hacer más ágil y precisa la consulta para el usuario.

Para comprobar si nuestra base de datos está actualizada, calculamos el área donde nos interesa mostrar puntos de interés, es decir, los que estarán cercanos a nosotros, y a que zonas dentro de nuestra región pertenece esta área.

Se ha definido una región de cuatro zonas con el eje central en las coordenadas x= 4137175 e y= 2055988, que corresponden a la cuidad de Sant Joan Despí. Se ha escogido esta localidad porque divide con su eje cuatro zonas urbanas y donde podría ser más útil solicitar este tipo de servicios.

Las coordenadas las podremos se pueden introducir manualmente, esto permite buscar PDIs fuera de localización actual, por ejemplo para planear un viaje. Una vez introducida la posición, el área de interés del usuario se calcula obteniendo las cuatro esquinas de un cuadrado calculado a través su posición x,y. Cuando obtenemos estas, se consulta una a una si están dentro de las cuatro zonas creadas para esta versión demo. De esta manera el programa sabe que información tiene que consultar en su base de datos.

El área de interés se calcula a través de un margen, que es el radio de distancia en el que se quiere que aparezcan puntos de interés. En el programa este margen se puede escoger desentendiendo. El usuario puede escoger un margen de 20.000 que corresponde a unos 4 Km de radio aproximadamente, o 10.000 si queremos un radio de unos 2 Km. Estos valores están fijados por el programa y el usuario puede escoger entre las dos opciones.

Las variables que utiliza el programa son las mismas que utiliza Google Maps, éste convierte la longitud y la latitud de una posición geográfica a números enteros. La clase Map activity trata las coordenadas como enteros para definir una posición y al pasarle los valores x e y tienen que tener este formato para que coincidan con la representación de Google Maps.

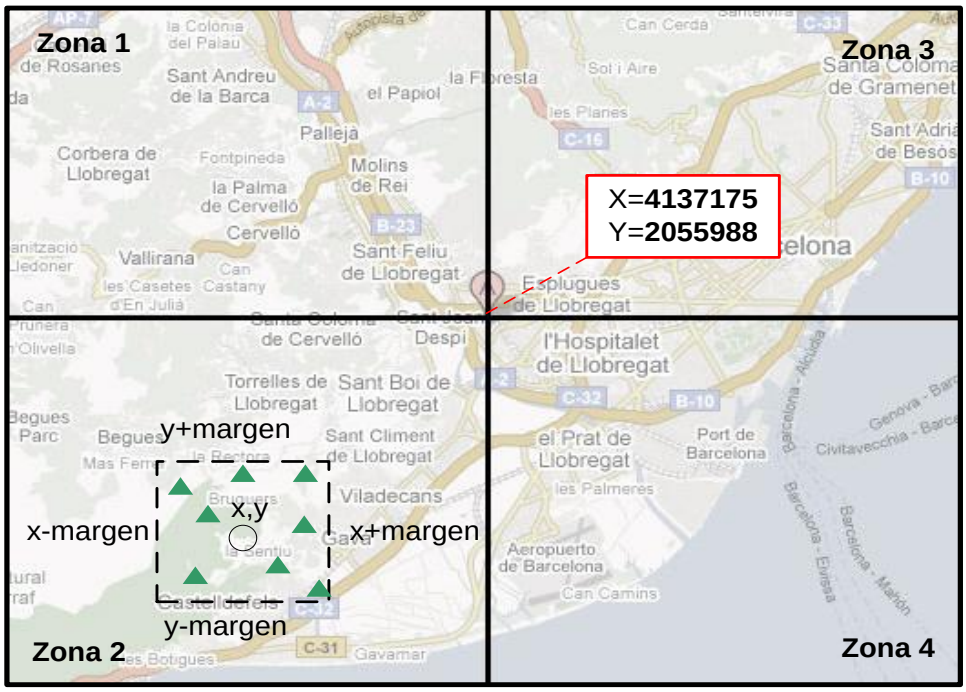


Fig. 3.5 Ejemplo de una región que solo necesita la descarga de la Zona 2.

3.5.2 Actualización de la base de datos.

Cuando el programa sabe a qué zonas pertenece la región del usuario, el siguiente paso que realiza es la comprobación de que su base de datos está completa. Para ello en cada descarga anterior se ha guardado un identificador de zona en una base de datos que exclusivamente guarda esta información. De esta manera se consulta directamente que zonas se han descargado anteriormente en la base de datos.

Para ello se extrae toda la información de la base de datos de zonas y se guarda en una hashtable, para que las posteriores consultas se hagan directamente sobre ella, evitando así un número elevado de consultas a la base de datos que hace que la aplicación sea más lenta. Se podría haber escogido una lista de zonas en vez de una hashtable, pero la búsqueda sobre tablas indexadas resulta más eficiente. Cuando el programa sabe que zonas necesita para completar su base de datos pasa a la clase que hace las peticiones las zonas que ha de pedir. Finalmente, se harán las peticiones al servidor correspondientes y se insertará en la base de datos de zonas el Id de la zona descargada para las posteriores consultas.

3.5.3 Peticiones al servidor y tratamiento de los datos.

En esta versión demo, el servidor lo único que dispondrá serán los diferentes archivos XML pertenecientes a cada zona. Es el terminal y no el servidor el que determina los datos que necesita el usuario. También se puede trabajar en

local y guardar los archivos dentro del proyecto, esto resulta cómodo para realizar las pruebas del programa.

Una vez hecha la descarga, el programa utiliza un Saxparser para el tratamiento de los datos y los inserta en la base de datos. Pudiendo elegir entre Sax y otra opción como DOM, la razón de elegir la opción SAX es que consume menos memoria y es muy sencilla de aplicar.

3.6. Notificación de archivos multimedia.

Cuando se muestre un punto de interés que contenga algún tipo de contenido multimedia, el programa lanzará un pop-up avisando al usuario de esté, permitiendo acceder al contenido multimedia o no. En caso de respuesta positiva se pasará a una activity diferente para reproducir el archivo.

Fig. 3.6 Notificación de dos archivos multimedia

3.7. Consultas a la base de datos por un PDI concreto.

El programa permite además de mostrar todos los PDIs de las zonas, realizar consultas a la base de datos por un PDI concreto. El usuario puede realizar las consultas a través de la activity destinada a la búsqueda de PDI y puede escoger entre varias opciones de búsqueda: a través del nombre concreto del PDI, el tipo de PDI que busca o sus coordenadas geográficas. La clase recoge los datos introducidos por el usuario en el widget y llama al método correspondiente de la clase PDIDao. Una vez hecha la consulta el programa crea una activity nueva e inserta todo el resultado de la búsqueda en un TextView.

Fig. 3.7 Pantalla de consulta de datos

3.8. Estructura de la base de datos

Ambas tienen el objetivo de evitar conexiones a Internet cada vez que se quiera hacer una consulta de PDI.

- **Base de datos de PDI.** Esta base de datos, guarda como su nombre indica todos los PDI descargados. La estructura de la tabla es la siguiente, coincidiendo con la estructura de las etiquetas de los XML para cada PDI:

Titulo	Pos X	Pos Y	Tipo	Precio	Mult.

- **Base de datos de Zonas.** La base de datos de Zonas se ha creado con la finalidad de guardar durante un cierto tiempo la identidad de las zonas que la aplicación ha tenido que descargar, incluso si el terminal móvil es apagado. En un principio se planteo el uso de una clase singleton para guardar estos valores, ésta evitaba tener que realizar consultas a la base de datos, por lo que resultaba una solución ágil, pero el problema aparece cuando un usuario apaga el terminal y al poco tiempo vuelve a hacer una petición. El móvil ya no guarda las zonas descargas y tiene que volver a realizar otra conexión. Para esta versión demo puede parecer una medida exagerada, ya que la información a descargar es muy pequeña, pero pensando en una aplicación real, donde la base de datos puede contener un número mucho más grande de información es una opción correcta.

Además para reducir el número de consultas a la base de datos de zonas, ésta es cargada en una hashtable. Por lo que solo haríamos una primera conexión a la base de datos y luego trataríamos la tabla.

La estructura de la tabla de zonas es la siguiente:

ID Zona	Descripción

Donde ID Zona es un numero entero que identifica la zona descargada y descripción será un String con alguna referencia al ID de Zona.

3.9 Reproducción de archivos Multimedia

Aunque se sabe que Android soportará los siguientes archivos multimedia: MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF, para esta versión del emulador la API de Android recomienda utilizar un archivo de video mp4 o 3gp. De archivos de imagen y audio, no se observa ningún tipo de limitación para la versión m5.

Para la reproducción de archivos multimedia la API de Android proporciona la clase Mediaplayer que permite reproducir archivos que estén guardados en el sistema, en un directorio Raw, o pasándole una URL. En la aplicación se ha probado con video 3gp y mp4, pero solo se ha conseguido reproducir el sonido de estos. Con la prueba de archivos mp3 no ha habido ningún tipo de problema. Esto es debido a que el emulador actualmente solo reproduce video de flujos de streaming que se descarguen progresivamente.

En la aplicación, cuando hay una notificación que avisa al usuario de algún contenido multimedia se puede acceder a la activity que lo reproduce. También se puede acceder desde el menú principal de la aplicación.

3.10. Map activities

Android incorpora una clase especial de activity llamada Map activity que adapta los conocidos Google Maps para su edición y utilización en las aplicaciones Android diseñadas por los usuarios.

Esta clase incorpora un conjunto de métodos que permite mostrar mapas según una posición geográfica introducida, dibujar elementos en él y incluso simular una señal de GPS sobre él entre sus posibilidades. En esta aplicación no se ha entrado en el detalle de todas las posibilidades que esta clase ofrece, se ha limitado a hacer una prueba sencilla que muestre al usuario un mapa por pantalla con la posición deseada.

En la aplicación se usa esta clase Map activity de dos maneras:

-La primera que recibe los parámetros X e Y de otra activity anterior y despliega una vista del Google Maps en el programa.

-la segunda que directamente ya tiene unas coordenadas fijas, para observar puntos de interés favoritos guardados por el usuario. Cada vez que el usuario llama a esta clase, no tiene que volver a introducir los parámetros X, Y.

Además se le ha incorporado algún control para el usuario, como la modificación del zoom de la pantalla, la vista por satélite y la vista del tráfico aunque esta última solamente está disponible para América.

3.11. Diseño del programa

El acceso a los contenidos de la aplicación se ha hecho de manera separada, para poder probar y visualizar más fácilmente los diferentes bloques que contiene la aplicación. Los componentes del programa son reutilizados varias veces accediendo a ellos a través de diferentes activities.

Cambiar el programa a un desarrollo secuencial no implicaría mucha dificultad dado que resulta muy fácil modificar la interfaz gráfica a través de los archivos XML y los Intent para pasar de una activity a otra.

Fig. 3.8 Pantalla del menú principal de la aplicación

3.11.1 Creación de la interfaz gráfica

Todas las pantallas del programa se han creado utilizando el modelo "XML-based-layout".

Para agilizar la creación de las UI se ha utilizado una aplicación Web hecha en Java llamada DroidDraw 0 Esta aplicación permite diseñar la interfaz a base de pegar widgets de Android en una simulación de una pantalla del emulador y generar el código automáticamente. Además permite modificar los atributos de los componentes de una manera mucho más visible y rápida.

3.11.2 Conjuntos de vistas

A la hora de crear una UI, tenemos diferentes "conjuntos de vistas" con los que trabajar. Un conjunto de vistas nos distribuye de una manera o de otra los objetos que se van colocando en la pantalla. Entre los cuales encontramos:

- **Frame Layout**: el más sencillo de todos, pensado para colocar solamente un objeto en un espacio en blanco (una foto, por ejemplo).

- **Linear Layout**: ubica los objetos unos al lado del otro según la dirección. Solo habrá uno por fila o columna. Seguramente el más apropiado para crear formularios.

- **Table Layout:** éste posiciona los elementos dentro de filas y de columnas formando como su nombre indica una tabla.

- **Absolute Layout:** con el podemos colocar los objetos en cualquier posición de la pantalla, estos utilizan las coordenadas "x" y "y". En el podemos encontrar que los objetos se solapen unos con otros. Será el utilizado en la aplicación.

- **Relative Layout:** para establecer posiciones relativas entre objetos.

Aunque el absolute layout es el menos recomendado, es el que se ha escogido puesto que se puede indicar exactamente la posición de un elemento en pantalla. A pesar de las recomendaciones de la API no ha dado ningún tipo de problema a la hora de implementarlo. La orientación de la pantalla es vertical.

CAPÍTULO 4. PLANIFICACIÓN DEL PROYECTO

El proyecto se ha dividido en tres partes: la primera que ha consistido en realizar un estudio de la plataforma, la segunda en el diseño de la aplicación a base de pruebas que se han ido probando con el Eclipse y finalmente el desarrollo de la aplicación. En este capítulo se enseña la trayectoria que se ha seguido y las limitaciones que se han encontrado.

4.1. Planificación del proyecto:

En el siguiente cuadro se muestran el número de horas aproximadamente dedicadas al proyecto:

Asunto	**Horas**
Introducción a Android, documentarse sobre la plataforma, posibilidades y situación actual de ésta.	40 h
Aprendizaje de la programación para Android: descarga de programas, estructura de un proyecto en Eclipse, funcionamiento de las activities, del cambio de datos entre ellas y los Intent, diseñar una interfaz gráfica con XML. Prueba con ejemplos sencillos, con SQLite y refuerzo del lenguaje de programación.	90 h
Diseño de la aplicación Elección de tecnologías utilizadas, comprobación de las limitaciones emulador: soporte de bluetooth, wi-fi, audio, video etc.	60 h
Pruebas con base de datos Perst	35 h.
Cambios del diseño, creación de la primera base de datos con SQLite, peticiones Http y tratamiento de los datos descargados del XML.	100 h.
Creación de la segunda base de datos, tratamiento de los Map activities, finalización de las partes básicas del programa	80 h.
Escritura de la memoria y revisión del programa.	40 h.
TOTAL DE HORAS DEDICADAS	445 h

4.2. Limitaciones o dificultades encontradas.

Todas las limitaciones que han aparecido en el desarrollo de la aplicación están relacionadas con la versión actual del SDK de Android, la m5. La última versión del emulador presenta bastantes incompatibilidades con la versión anterior m3.

4.2.1. Versión m5 del emulador

Ha resultado problemático a la hora de implementar algunos métodos de las clases de SQLite que han variado para la versión m5. Para ello en la API oficial hay publicado un documento con los cambios dados en las versiones.

4.2.2. Base de datos Perst

En el diseño principal de la aplicación se pensó en utilizar la base de datos orientada a objetos Perst, por su facilidad de implementación y por su modelo de búsqueda espacial. La API de Perst no estaba actualizada a la versión m5 en el momento de construcción de la aplicación y se tuvo que utilizar SQLite.

4.2.3. Bluetooth

En la API de Android hay publicada la clase de Bluetooth, pero éste todavía no puede ser emulado ya que la clase publicada está actualmente incompleta. Se pensó en la posibilidad de utilizar bluetooth para notificaciones entre dos dispositivos que estuvieran cerca, permitiéndole añadir alguna funcionalidad más al programa.

4.2.4. Reproducción de video.

Como ya se ha explicado en un apartado anterior, solamente se ha conseguido reproducir el audio de los ficheros 3gp y mp4.

CAPÍTULO 5. CONCLUSIONES

En el último capítulo se enseñan las conclusiones en referencia a los objetivos marcados en el planteamiento del proyecto y se explican las mejoras y líneas futuras que se podrán hacer a la aplicación. Se tienen en cuenta las limitaciones del emulador, funcionalidades que se podrían añadir y funcionalidades que se pueden plantear para un futuro y que con la versión actual de momento no se pueden implementar.

5.1. Conclusión general.

Se ha podido aprender a desarrollar aplicaciones para Android en un periodo corto de tiempo gracias a la gran cantidad de información que hay disponible, como es lógico Google ha facilitado tutoriales y ha publicado una API para acelerar la creación de programas por parte de los usuarios. La instalación de los programas es rápida y al utilizar un lenguaje de programación tan conocido como Java facilita aun más las cosas. Además, el emulador dispone de un kit de demos con ejemplos de aplicaciones muy interesante que producen interés nada más instalarlo. Así que se puede afirmar que hay una gran predisposición para poner las cosas fáciles al programador y motivar a éste con los elementos que puede usar y la libertad de la que dispone para desarrollar, abriéndose infinitas posibilidades de aplicaciones.

Que Android sea tan accesible, resultará cómodo y atractivo para las empresas que tendrán más predisposición para apostar por la plataforma y para el usuario con nociones de programación, que sin llegar a ser un experto también puede crear y adaptar programas para sus necesidades.

Google solo se encarga del software del teléfono, dejando a los fabricantes la creación del hardware, esto es bueno, ya que compañías con ya experiencia en el sector pueden crear terminales que permitan aprovechar al máximo la tecnología de la plataforma. Esto además atraerá a público que ya confía en marcas como HTC añadiendo el atractivo de un sistema operativo creado por Google, desde luego despertará interés.

La versión actual del emulador presenta algunas limitaciones que han impedido realizar algunas funcionalidades para el programa, aun así se ha podido conseguir con éxito la propuesta principal que era realizar una prueba de concepto y una primera toma de contacto con la plataforma. Se ha podido realizar una aplicación que puede ser el punto de partida para un sistema completo que utilice el servicio de localización geográfica de Android. El desarrollo de la aplicación ha sido el correcto, encontrando travas leves por ser un sistema todavía incompleto y por ser la primera vez que se realiza una aplicación para esta plataforma. No se ha tenido ninguna limitación importante que no se pueda sustituir por otra tecnología o que no se pueda implementar en un futuro, por lo que la valoración final de la implementación resulta muy optima.

Estudiando tan solo una parte de la plataforma, se toma consciencia que Android tiene mucho más que ofrecer despertando las ganas de ampliar y explorar otros elementos como los gestores de notificaciones o los controladores de mapas por ejemplo.

5.1.1. El futuro de Android

Estudiadas las posibilidades que ofrece actualmente y las futuras aplicaciones que se podrán desarrollar, Android tiene todas las cualidades para ser un producto rentable. Tiene propuestas innovadoras, una comunidad de programadores que estarán dispuestos a apoyar un teléfono de código abierto, un gran nombre de empresas implicadas para la creación de éste y una empresa como Google que dispone de enormes recursos para llevar el éxito a Android. Aun así, Google tendrá que luchar para hacerse un hueco en el mundo de la telefonía móvil, tiene grandes competidores que ya tienen mucha experiencia al sector con clientes bastante fieles, pero que creemos que lo acabará consiguiendo.

5.2. Líneas futuras

Como se ha dicho de la aplicación, ésta es un punto de partida para crear un sistema completo con funcionalidades que utilicen el sistema de localización geográfica. Quedan muchas mejoras que se pueden hacer sobre el a medida que la versión del emulador vaya siendo más completa. Algunas de las que se muestran en este apartado ya se pueden realizar como el montaje del servidor que por prioridad de objetivos en el proyecto no se ha hecho y que darían más inteligencia a la aplicación y la acercarían más a ser un producto que se pueda comercializar.

5.2.1. Base de datos Perst

Una mejora de la aplicación, sería utilizar una base de datos Perst, más sencilla que SQLite a la hora de implementar

5.2.2. Lógica del servidor.

Con el montaje de un servidor externo se podría reducir la complejidad del código en el terminal y traspasar parte de la lógica del programa al servidor. De esta manera se podría dejar que fuera el servidor el que calculará la región en la que está el usuario y la información que necesita descargar con el

intercambio de mensajes HTTP El usuario realizaría una conexión con el servidor enviando sus coordenadas geográficas, y el servidor devolvería una página HTML con los resultados obtenidos.

El servidor tendría una base de datos que almacenase todos los PDIs y los buscase por su posición. En este caso no haría falta definir zonas, ya que se guardarían por igual en la base de datos y las búsquedas obtendrían los PDIs que estén dentro de los márgenes establecidos. También se podría crear una base de datos más pequeña para que el usuario guardase aquellos puntos de interés como “favoritos”. Las consultas por puntos de interés concretos se realizarían de la misma manera a través de formularios http.

Esta alternativa implicaría un programa más sencillo y ágil en el terminal, pero un número de conexiones más elevado, por lo que al usuario le saldría más caro, además que no trata tanto SQLite en el terminal, que era uno de los objetivos.

5.2.3. Utilización del servicio de notificación y Bluetooth.

Con el uso de bluetooth, que seguramente se podrá emular en una próxima versión del emulador, se podría añadir una funcionalidad que utilizase el gestor de notificaciones que ofrece el sistema operativo. El terminal podría crear una alerta al aproximaros a un punto que estuviese difundiendo mensajes con contenido multimedia a través de esta tecnología y que dejase al usuario escoger entre la opción de descargarlo o descartarlo.

5.2.4. Actualización de la base de datos

La tabla de zonas podría contener una columna con un campo DATE por ejemplo para controlar la versión de la zona descargada que tenemos guardada en la base de datos y pedirla en caso que sea obsoleta.

5.2.5 Simulación de una señal GPS.

La simulación de GPS para Android está disponible para la versión m5. Se puede hacer una simulación guardando una ruta creada con la herramienta de Google Earth en la carpeta del GPS del emulador y creando una Map activity que la reproduzca.

5.2.6 Instalación del Sistema Operativo en un terminal real.

Para probar con más exactitud como soporta un terminal la aplicación diseñada, se podría realizar una prueba instalando el sistema operativo en un terminal. Esto ya es posible y se pueden encontrar artículos con la prueba de Android sobre diferentes terminales.

Bibliografía

[1] API oficial de Android: http://code.google.com/Android

[2] Open Handset Official Page: http://www.openhandsetalliance.com/

[3] Droiddraw: http://www.droiddraw.org/

[4] Ejemplos y tutoriales: http://www.anddev.org/

[5] Ejemplos y tutoriales: http://www.Android-spa.com/

[6] Información general: http://www.wikipedia.es

[7] Blog de Android en español: http://celutron.blogspot.com/

[8] Ejemplo de SQLite: http://developer.db4o.com/ProjectSpaces/view.aspx/Android_Password_Manager

[9] Grupos en Google de Android: http://groups.google.com/group/Android-developers/files

[10] Artículo en el Periódico sobre Android: http://www.elperiodico.com/default.asp?idpublicacio_PK=46&idioma=CAS&idnoticia_PK=482576&idseccio_PK=1009

ANNEXOS

TÍTOL DEL TFC/PFC

TITULACIÓ:

AUTOR: Judith Balaguero Peña

DIRECTOR: Toni Oller

DATA: 24 de julio de 2008

Anexo I) Empezar a programar con Android

Este guión no es una traducción de los pasos a seguir si no una explicación de los primeros conceptos que tenemos que tener en cuenta a la hora de empezar a programar y una valoración en la primera toma de contacto sobre lo sencillo o complicado que puede llegar a resultar. El primer ejemplo que se ha realizado para Android es el llamado HelloAndroid, una versión del mítico HelloWord de Java y obligado para todo aquel que vaya a tocar Android por primera vez.

Lo primero que tenemos que hacer es descargarnos el entorno de desarrollo de la página oficial de Android () y disponer del programa Eclipse. Ambos son gratuitos y hay múltiples portales en internet para acceder a ellos. Siguiendo los pasos que nos marcan en la página oficial (aunque también tenemos muchos tutoriales en castellano en otras direcciones de internet) descomprimimos el archivo que contiene el SDK de Android y instalamos el plug-in del eclipse, resulta bastante sencillo y está todo muy bien guiado
.

Ejemplo Hello Android.

Vemos que nuestra clase extiende de activity, esto quiere decir que activity ya es una clase hecha por Android (muy sencilla que equivale a una pantalla y poco más) y lo que nosotros hacemos es ampliar ésta y con el super.onCreate creamos un hijo que hereda las características de activity.

```
package com.Android.hello;
import Android.app.Activity;
import Android.os.Bundle;
import Android.widget.TextView;
public class HelloAndroid extends Activity {
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle icicle) {
     super.onCreate(icicle);
     TextView tv = new TextView(this);
     tv.setText("Hello, Android");
     setContentView(tv);
  }
}
```

En la última línea (setContentView(R.layout.main)) le estamos diciendo al programa que visualice en pantalla la interfaz que hay en R.layout.main o dicho de alguna manera, en el directorio res/layout/main.XML tendremos unos elementos definidos (botones, imágenes, iconos, fondos de pantalla etc.) con unas características especificas y colocados en ciertas partes de la pantalla y lo que le estamos diciendo al programa es que nos muestre la pantalla tal como está descrita en este main.XML. Este main.XML es nombrado como UI.
En este punto del programa podríamos compilar y automáticamente sería ejecutado el emulador, se nos mostraría una pantalla negra ya que nuestro

main esté vacío y nuestro programa no tiene ninguna instrucción aparte del código fuente.

Ahora modificaremos el código para que en vez de mostrar la UI, nos muestre un objeto creado directamente en la clase principal y sin hacer ninguna referencia al main.XML.

Creamos el objeto TextView e importamos la clase. El (this) a la hora de crearlo indica que no está predefinido en el main.XML (hay que ver el XML como si fuera un CSS de una página Web). Y con la instrucción setText lo que estamos haciendo es poner un mensaje para que aparezca en pantalla. Ahora al ejecutar la aplicación lo que queremos mostrar no es el main.XML ya que está vació si no TextView por eso ahora le decimos que lo que muestre sea tv y no el R.layout.main.

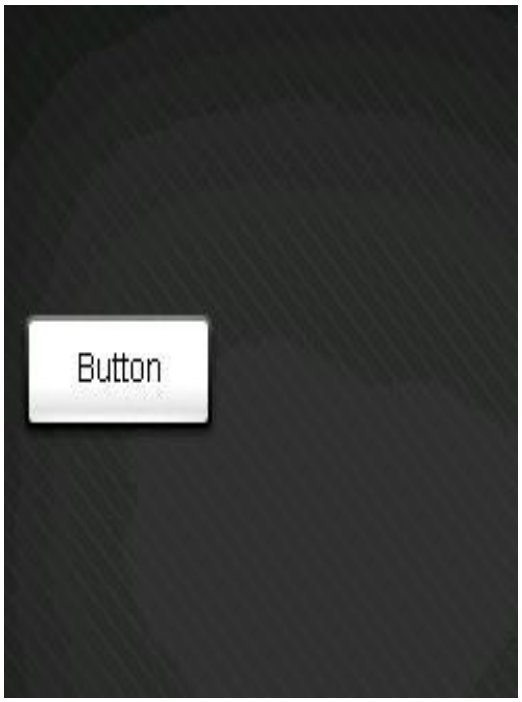
Este ejemplo, que es la primera toma de contacto con Android sirve para orientarnos sobre todo con las construcciones de las UI y la declaración de objetos en las clases. A partir de ahora los siguientes programas relacionarán los objetos creados con su definición en la UI a la hora de mostrarse por pantalla.

Desde el momento que se empieza a hacer una aplicación después del ejemplo de HelloAndroid se ha de tener a mano la API publicada en http://code.google.com/Android/ , ya que constantemente estaremos consultándola. Las explicaciones que vienen a continuación parten de una aplicación con el código fuente que se genera automáticamente y que va haciéndose más compleja.

Construyendo una UI

Como ya se ha explicado anteriormente, una UI es el archivo dentro de nuestro proyecto que nos describe la interfaz gráfica que tendrá nuestra aplicación. Ésta estará localizada dentro del directorio /res/layout (sin hablar de páginas web dinámicas) y será un archivo XML. Para quien ya conoce XML es muy sencillo aplicar estilos, y para quien no lo ha utilizado nunca con mirar un par de ejemplos resulta suficiente. La creación de interfaces por XML resulta mucho más sencilla y amigable que no haciéndolo a través de código directamente.

```
<?XML version="1.0" encoding="utf-8"?>
<AbsoluteLayout
Android:id="@+id/widget42"
Android:layout_width="fill_parent"
Android:layout_height="fill_parent"
XMLns:Android="http://schemas.Android.com/apk/res/Android"
>
<Button
Android:id="@+id/widget69"
Android:layout_width="94px"
Android:layout_height="43px"
Android:text="Button"
Android:layout_x="10px"
Android:layout_y="112px"
>
</Button>
</AbsoluteLayout>
```



Un ejemplo sencillo de una UI:

En el estamos definiendo una pantalla simple de tipo lineal que ocupe todo el espacio de la pantalla. Dentro de la pantalla estamos declarando un botón, el identificador de éste será ok, ocupara el máximo espació posible dentro de la pantalla, estará situado a la derecha de éste, y dentro del botón aparecerá el texto “Button”. Como se puede ver es simplemente definir el objeto dándole atributos.

Si compilásemos nuestro programa ahora solo con el código fuente que genera la clase al crearse, nos aparecería en la pantalla un simple botón con las características que hemos descrito anteriormente.

El ejemplo mostrado es muy sencillo, la UI de Android se ha de ir creando manualmente aunque podemos encontrar en Internet una aplicación java en la que diseñamos la interfaz a base de pegar objetos en una pantalla y posteriormente ésta nos genera automáticamente el código. Ésta se encuentra en www.droiddraw.org.

Anexo II) Cambio de Activity

El siguiente código muestra el cambio de una pantalla del programa a otra, como vemos se realiza a través de un intent. Esta activity tiene que estar declarada en el Androidmanifiest. El cambio se produce al pulsar un botón.

```
mostrarPDI.setOnClickListener(new View.OnClickListener() {

                  public void onClick(View view) {
                        try {
                              Intent i = new Intent(Basemain.this,
ObtenerPDILista.class);


                           startActivity(i);

                        } catch (Exception e) { }


                  }

            });
```

Anexo III) Paso de datos entre activities.

Igual que el codigo anterior, hacemos un cambio de activity pero ahora añadiendo dos variables a un bundle que se enviará a otra activity. En este caso pasamos las coordeanadas x y y.

```
buscarPDIpos.setOnClickListener(new View.OnClickListener() {

            public void onClick(View view) {
                  try {

                        Intent i = new Intent(Basemain.this,
ObtenerPosicion.class);
                           Bundle b = new Bundle();
                         b.putString("posx",
x.getText().toString());
                        b.putString("posy", y.getText().toString());
                         i.putExtras(b);
                    startActivity(i);




      } catch (Exception e) { }
            }}

            );
```

Anexo IV) Recoger datos en una activity.

Se recogen los datos de una activity recogiendo el bundle que se pasa a la activity, y obteniendo las variables (con el mismo nombre que se creo ya que se pueden meter varios datos en un mismo bundle)

```
        Bundle d = this.getIntent().getExtras();

        final int posy =
Integer.parseInt(d.getString("posy").toString());
        final                int                posx                =
Integer.parseInt(d.getString("posx").toString());
```

Anexo V) Finalizar una Activity:

Una Acitvity puede tener diferentes ciclos, en nuestro caso no hace falta que las activities devuelvan valores o que guarden la información del usuario, si fuera así al cerrar la activity crearía un resultado que se enviaría a la que la lanzó.

```
salir.setOnClickListener(new View.OnClickListener() {

                public void onClick(View view) {
                      finish();

                }

            });
```

Alerta al finalizar la aplicación

Anexo VI) Consultas a la base de datos:

Con este método obtenemos todos los PDI que hay dentro de la base de datos.

```
public List<PDI>  getAlltitle(String comp) throws ParseException {
        //abrimos la base de datos
      mDb = myAdapter.open();

        Cursor cu = mDb.query(PDIDbAdapter.DATABASE_TABLE_PDI,
                  new String[] { "id",
"title","posy","posx","type","price","peliculas" },"title LIKE '%" +
comp+"%'",null , null, null, null);



        List<PDI> l = new ArrayList<PDI>();
        while (cu.next()) {
            PDI c = new PDI();

            c.setTitle(cu.getString(cu.getColumnIndex("title")));

            c.setPosy(cu.getInt(cu.getColumnIndex("posy")));
            c.setPosx(cu.getInt(cu.getColumnIndex("posx")));
            c.setType(cu.getString(cu.getColumnIndex("type")));
            c.setPrice(cu.getFloat(cu.getColumnIndex("price")));
```

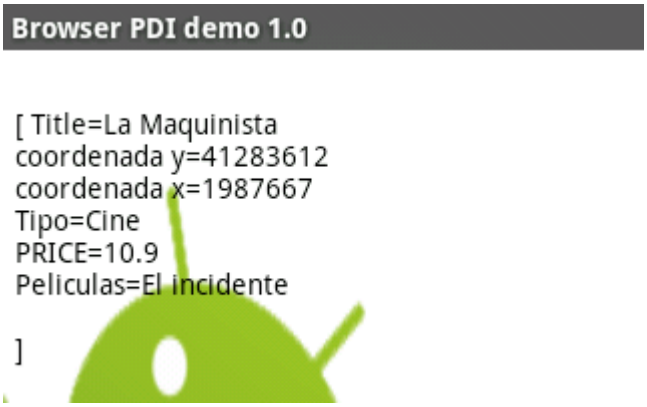
```
c.setPeliculas(cu.getString(cu.getColumnIndex("peliculas")));


            l.add(c);

        }

        mDb.close();

        return l;
    }
```

Muestra de todos los PDI que hay en la base de datos.

Ejemplo de una consulta que nos devuelve los PDI que contengan el titulo que introducimos (no es necesario introducirlo exacto):

```
   public List<PDI>  getAlltitle(String comp) throws ParseException {
        //abrimos la base de datos
      mDb = myAdapter.open();

        Cursor cu = mDb.query(PDIDbAdapter.DATABASE_TABLE_PDI,
                  new String[] { "id",
"title","posy","posx","type","price","peliculas" },"title LIKE '%" +
comp+"%'",null , null, null, null);
```

```
        List<PDI> l = new ArrayList<PDI>();
        while (cu.next()) {
            PDI c = new PDI();

            c.setTitle(cu.getString(cu.getColumnIndex("title")));
                 //metodo que damos el nombre de la columna y nos
devuelve el indice
            c.setPosy(cu.getInt(cu.getColumnIndex("posy")));
            c.setPosx(cu.getInt(cu.getColumnIndex("posx")));
            c.setType(cu.getString(cu.getColumnIndex("type")));
            c.setPrice(cu.getFloat(cu.getColumnIndex("price")));

c.setPeliculas(cu.getString(cu.getColumnIndex("peliculas")));


            l.add(c);

        }

        mDb.close();

        return l;
    }
```



Consulta y resultado obtenido

Anexo VII) Creación de un mapa en una MapActivity

Creamos un objeto MapView, una variable de tipo point y controlador de mapa para añadirle la posición y elegir el zoom al crear la activity. Al finalizar mostramos el mapa por pantalla.

```
MapView mMapView=new MapView(this);
```

```
Point p = new Point(41331785,2015988);
MapController mc = mMapView.getController();
mc.animateTo(p);
mc.zoomTo(9);
mc.centerMapTo(p, true);
```

```
setContentView(mMapView);
```

Vista de satélite con el centro en San Joan Despí.

Vista normal del mapa.

Anexo VIII) Android Manifiest:

En el Android manifiest se declaran todas las activities del programa, al crear un proyecto Nuevo para Android, la primera activity que se crea con el proyecto aparece en el manifiest como la activity principal como el caso de este ejemplo en la activity llamada androidtfc**.**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.activities">
<application android:icon="@drawable/icon">
<activity android:name=".androidtfc" android:label="@string/app_name">
            <intent-filter>
           <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>
<activity android:name=".Registro" android:label="@string/app_name" />
<activity android:name=".ParsingXML" android:label="@string/app_name"
/>
    </application>
</manifest>
```

Anexo IX) Arquitectura de Android.

Los Componentes mayores del sistema operativo de Android, cada sección se describe en detalle:

- Aplicaciones: Las aplicaciones base incluirán un cliente de email, programa de SMS, calendario, mapas, navegador, contactos, y otros. Todas las aplicaciones escritas en el lenguaje de programación Java.

- Framework de aplicaciones: Los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar el rehúso de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Éste mismo mecanismo permite que los componentes sean reemplazados por el usuario.

- Librerías: Android incluye un set de librerías C/C++ usadas por varios componentes del sistema Android. Éstas capacidades se exponen a los desarrolladores a través del framework de aplicaciones de Android. Algunas son: System C library (implementación librería C standard), librerías de medios, librerías de gráficos, 3d, SQLite, entre otras.

- Runtime de Android: Android incluye un set de librerías base que proveen la mayor parte de las funcionalidades disponibles en las librerías base del lenguaje de programación Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros, y corre clases compiladas por el compilador de Java que han sido transformadas al formato .dex por la herramienta incluida "dx".

- Núcleo - Linux: Android depende de un Linux versión 2.6 para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, stack de red, y modelo de drivers. El núcleo también actúa como una capa de abstracción entre el hardware y el resto del stack de software.

Anexo X) GLOSARIO

API (Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

HTTP (*HyperText Transfer Protocol*) Define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web (clientes, servidores, proxies) para comunicarse.

IPhone. Teléfono móvil multimedia, con capacidad para navegar en Internet y tecnología GSM, EDGE y 3G, que es desarrollado y comercializado por Apple. Tiene un pantalla multi-táctil con un botón y teclado virtual, una cámara integrada de 2.0 megapíxeles (sin flash y sin posibilidad de grabar video de manera nativa) e incorpora las funciones de un reproductor multimedia portátil ("iPod"). No es posible mandar mensajes multimedia hasta el momento, aunque sí permite el envío de correos de voz visuales; también incluye servicios de Internet como el correo electrónico, la navegación web y la conectividad Wi-Fi.

Framework: En el desarrollo de software, un ***framework*** es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un *framework* puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Los *Frameworks* son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

Map activity. Base class with code to manage the boring necessities of any activity that displays a MapView:

- activity lifecycle management
- Setup and teardown of services behind a MapView

A subclass should create its own MapView in onCreate(Bundle).

Podscast. El podcasting consiste en la creación de archivos de sonido (generalmente en formato mp3 o AAC, y en algunos casos ogg) y de video (llamados videocasts o vodcasts) y su distribución mediante un archivo RSS que permite suscribirse y usar un programa que lo descarga de Internet para que el usuario lo escuche en el momento que quiera, generalmente en un reproductor portátil.

SAX son las siglas de "Simple API for XML", originalmente, una API únicamente para el lenguaje de programación Java, que después se convirtió en la API estándar *de facto* para usar XML en JAVA. Existen versiones de SAX no sólo para JAVA, sino también para otros lenguajes de programación.

Symbian. Sistema operativo que fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran Nokia, Sony Ericsson, PSION, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic, Sharp, etc. Sus orígenes provienen de su antepasado EPOC32, utilizado en PDA's y Handhelds de PSION.

SQLite. Es un sistema de gestión de bases de datos relacional compatible con ACID, y que está contenida en una relativamente pequeña (~500KB) librería en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp.

La librería SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

View. The View class represents the basic UI building block. A view occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for *widgets*, used to create interactive graphical user interfaces.

WebKit es un framework para aplicaciones que funciona como base para el navegador web Safari y que facilita a los desarrolladores incluir gran parte de las funcionalidades de Safari en sus propias aplicaciones.

Widget The widget package contains (mostly visual) UI elements to use on your Application screen.

XML, (***Ex****tensible* ***M****arkup* ***L****anguage*) es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.