

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN  
UNIVERSIDAD POLITÉCNICA DE CARTAGENA



Proyecto Fin de Carrera

Sistema para la geolocalización y recuperación de puntos de interés por  
medio de terminales Android  
(Interest points geolocation and query system based on android)



AUTOR: CAROLINA SÁNCHEZ GARCÍA  
DIRECTOR: JAVIER VALES ALONSO  
Junio / 2013



<b>Autor</b>	Carolina Sánchez García
<b>E-mail del Autor</b>	<a href="mailto:carolina.sanchez.garcia@gmail.com">carolina.sanchez.garcia@gmail.com</a>
<b>Director</b>	Javier Vales Alonso
<b>E-mail del Director</b>	Javier.vales@upct.es
<b>Título del PFC</b>	Sistema para la geolocalización y recuperación de puntos de interés por medio de terminales Android
<b>Descriptor</b>	Smartphone, Sistema operativo móvil, Android, Java, GPS, SQLite
<p><b>Resumen</b></p> <p>En algunas ocasiones, sobre todo cuando nos encontramos de viaje, se ha dado la situación en que nos hemos perdido y no logramos volver a un punto de nuestro interés. Debido a esta problemática, surge la idea de una aplicación capaz de guardar todas las ubicaciones que el usuario desee, además de la posibilidad de mostrar la ruta para su posterior retorno.</p> <p>Para la creación de dicha aplicación, se propone la utilización de smartphones con sistema operativo Android, ya que gracias a su pequeño tamaño son fáciles de transportar y, además, incorporan los sensores necesarios para recoger los datos de ubicación del usuario.</p>	
<b>Titulación</b>	Ingeniería Técnica de Telecomunicación, especialidad Telemática
<b>Intensificación</b>	-
<b>Departamento</b>	Tecnologías de la Información y las Comunicaciones
<b>Fecha de Presentación</b>	06-2013

<b>Capítulo 1. Introducción .....</b>	<b>2</b>
1.1. Necesidad del proyecto.....	2
1.2. Objetivos.....	2
1.3. Orígenes de Android .....	2
1.4. Organización de la memoria .....	3
<b>Capítulo 2. Tecnologías asociadas.....</b>	<b>4</b>
2.1. Programación orientada a objetos: JAVA.....	4
2.2. Entorno de programación ECLIPSE.....	4
2.3. Plataforma Android.....	5
2.3.1. Introducción a la Arquitectura de Android.....	5
2.3.2. Arquitectura por capas Android.....	5
2.3.3. Máquina virtual Dalvik.....	7
2.3.4. Ciclo de vida de las aplicaciones Android.....	7
2.3.5 Seguridad en Android .....	9
2.3.6. Componentes de las aplicaciones Android.....	10
<b>Capítulo 3. Creación del entorno .....</b>	<b>11</b>
3.1. Descargar Eclipse .....	11
3.2. Descarga e instalación del SDK de Android .....	12
3.3. Vincular Eclipse y SDK.....	13
<b>Capítulo 4. Uso de la aplicación. Explicación de la implementación .....</b>	<b>18</b>
4.1. Pantalla principal.....	18
4.2. ¿Dónde estoy? .....	20
4.3. ¡Quiero volver! .....	24
4.4. Opciones .....	26
4.4.1. Borrar todo.....	26
4.4.2. Borrar.....	27
4.4.3. Editar.....	28
4.5. Otras implementaciones de interés .....	29
4.5.1. Pantallas de control.....	29
4.5.2. Interfaz gráfica .....	30
4.5.3. Android Manifest.....	31
<b>Capítulo 5. Conclusiones y trabajo futuro.....</b>	<b>32</b>
5.1. Posibles mejoras y trabajo futuro.....	32
<b>ANEXO A. Código de la aplicación .....</b>	<b>33</b>
A.1. Adaptador Lista.....	33
A.2. AndroidMapas .....	34
A.3. Borrar.....	38
A.4. Editar .....	39
A.5. Elegir.....	41
A.6. MapOverlay .....	42
A.7. MyDBAdapter .....	43
A.8. MyDBCursorAdapter .....	46
A.9. MyLocation .....	47
A.10. MyLocationListener .....	50
A.11. MyOverlay.....	52
A.12. Opciones .....	55
A.13. ViewMaps.....	58
A.14. ViewReturn.....	62
A.14. AndroidManifest .....	65
<b>Anexo B. Obtención de la API Key de Google .....</b>	<b>67</b>
<b>Anexo C. Bibliografía y Referencias .....</b>	<b>69</b>

## Capítulo 1. Introducción

En las siguientes líneas se hace una breve introducción al presente proyecto, exponiendo cuál es su motivación y qué objetivos son los que persiguen.

### 1.1. Necesidad del proyecto

El presente documento recoge toda la información del desarrollo llevado a cabo para el proyecto final de carrera. Este proyecto surge a partir de la motivación de desarrollar una aplicación para una plataforma móvil que permita al usuario guardar ubicaciones que sean de su interés y, además, volver a dicha ubicación en el momento que se quiera, a través de las rutas que la propia aplicación devuelve.

Con este objetivo global, se ha utilizado para el desarrollo la plataforma Android al ser una de las plataformas con mayor ratio de crecimiento del mercado y por permitir mediante interfaces, crear todas las funcionalidades que eran necesarias para el desarrollo de esta aplicación como se podrá ver a lo largo del documento.

### 1.2. Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación para un dispositivo móvil que facilite al usuario la geolocalización y la posibilidad de guardar y obtener puntos de interés geográficos a través de un mapa, mediante una interfaz simple y sencilla.

Los dispositivos móviles utilizados son smartphones con sistema operativo Android, este sistema operativo ha sido seleccionado para el presente proyecto por estos motivos:

- Sistema que cuenta con el respaldo de Google, una de las empresas más grandes a nivel mundial.
- Numerosos distribuidores y operadores se han unido a la plataforma de patrocinadores de Android, y fabricantes de gran renombre han anunciado la producción inminente de nuevos modelos con Android como sistema nativo, siendo algunos de ellos: HTC, Samsung, Sony Ericson o Motorola.
- Crecimiento de ventas de dispositivos con este sistema operativo, llegando incluso a ser líder en algunos países.
- Es posible la realización de aplicaciones en ordenadores con sistema operativo Windows, ampliamente extendido.
- Existencia de plugin que permite el desarrollo de la aplicación en el entorno de programación Eclipse, con todas las facilidades que ello conlleva.
- El lenguaje de programación utilizado es Java, uno de los lenguajes mas conocidos por los programadores.

### 1.3. Orígenes de Android

Android fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio. Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestres de 2010, con una cuota de mercado de 43,6% en el tercer trimestre

Android tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. A finales de 2012 ya existían más de 700.000 aplicaciones disponibles para Android. Android Market (GooglePlay) es la tienda de aplicaciones oficial en línea administrada por Google, aunque existe la posibilidad de obtener software externamente, por ejemplo: Samsung tiene su propia tienda (Samsung App).

Google liberó la mayoría del código de Android bajo la licencia Apache, una licencia libre y de código abierto. Actualmente Android posee el 32,9% de cuota de mercado a escala mundial de los teléfonos inteligentes, por delante de Symbian OS que tiene 30,6%. En tercer lugar se sitúa Apple con cuota de mercado del 16%. 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.

Hoy en día, se encuentra en uso la versión de Android 4.2.2, Jelly Bean. El nombre de las versiones de Android reciben el nombre de postres en inglés, en la que el postre elegido empieza por una letra distinta, siguiendo el orden alfabético:

- A: Apple Pie (v1.0)
- B: Banana Bread (v1.1),
- C: Cupcake (v1.5)
- D: Donut (v1.6)
- E: Éclair (v2.0/v2.1)
- F: Froyo (v2.2), (Abreviatura de «frozen yogurt»)
- G: Gingerbread (v2.3)
- H: Honeycomb (v3.0/v3.1/v3.2)
- I: Ice Cream Sandwich (v4.0)
- J: Jelly Bean (v4.1/v4.2)

#### **1.4. Organización de la memoria**

La organización de los capítulos de la memoria es la siguiente:

- Capítulo 2: En este capítulo se exponen las diferentes tecnologías que han permitido el desarrollo de la aplicación
- Capítulo 3: Aquí se explica como se ha preparado el entorno para hacer posible la implementación y el desarrollo.
- Capítulo 4: En este capítulo se explica con detalle el uso de la aplicación, además de como se han implementado ciertos puntos que quizá resulten más interesantes de comentar.
- Capítulo 5: Se ofrecen las conclusiones después de la finalización del proyecto, además de las mejoras que se introducirían en un futuro.
- Anexo A: Se encuentra toda la implementación de la aplicación
- Anexo B: Bibliografía y referencias

## Capítulo 2. Tecnologías asociadas

Se presentan a continuación las principales tecnologías que han permitido el desarrollo del sistema de geolocalización y posterior retorno a punto de interés basado en sistemas Android.

El lenguaje de programación Java es utilizado para la creación de la aplicación Android. El entorno de programación utilizado es Eclipse.

Al utilizar el sistema operativo Android, se precisan conocimientos sobre esta plataforma.

La geolocalización se realiza a través del GPS integrado en el smartphone. Además, es necesario el uso de 3G o Wifi, para la carga de mapas.

### 2.1. Programación orientada a objetos: JAVA.

El lenguaje de programación Java es el utilizado para la creación de aplicaciones para los dispositivos Android. Es un lenguaje orientado a objetos desarrollado por Sun Microsystems a principios de los años 90 y algunas de sus características principales son:

- **Sencillo.**

El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

- **Robusto.**

Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. La recolección de basura elimina la necesidad de liberación explícita de memoria.

- **Multihilo.**

Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, pudiendo así una aplicación ejecutar múltiples acciones a la vez.

- **Permite realizar aplicaciones distribuidas.**

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

- **Indiferente a la arquitectura: El problema de los dispositivos móviles.**

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos.

### 2.2. Entorno de programación ECLIPSE.

Comenzó como un proyecto de IBM. En la actualidad es desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto, es decir, código distribuido y desarrollado libremente.

Consiste en un Entorno de Desarrollo Integrado (IDE, Integrated Development Environment). Un IDE es un programa compuesto por un conjunto de herramientas útiles para un desarrollador de software. Como elementos básicos, cuenta con un editor de código, un compilador/intérprete y un depurador.

Eclipse es utilizado mayoritariamente como IDE Java, llamado éste Java Development Toolkit (JDT). Aunque también da soporte a otros lenguajes de programación, como son C/C++, Cobol, Fortran, PHP o Python.

A la plataforma base de Eclipse se le pueden añadir complementos (plugins) para extender la funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no.

Así pues, para integrar Android en Eclipse, se utiliza el plugin ADT (Android Development Tools), que permite el uso de las herramientas y métodos imprescindibles para la realización de aplicaciones, que están recogidas en el SDK (software developer kit) de Android. Existe además disponible una amplia documentación de respaldo para este SDK.

Eclipse y todos los complementos necesarios para el desarrollo de aplicaciones Android son utilizados en este proyecto para la creación de la aplicación final

## **2.3. Plataforma Android.**

Android es una solución completa de software de código libre para dispositivos móviles, la cual se encuentra instalada en el smartphone donde se han ejecutado las pruebas de este proyecto.

Se distribuye bajo una licencia Apache, versión 2, lo que implica que, al tratarse de software libre, cualquier desarrollador tiene acceso completo al SDK del sistema, incluidas todas sus API, documentación y emulador para pruebas, pudiendo distribuirlo y modificarlo. Además, esta licencia permite a los desarrolladores tanto publicar sus creaciones, como distribuirlas ocultando el código fuente.

### **2.3.1. Introducción a la Arquitectura de Android.**

Android proporciona un paquete completo de software a todos los niveles:

- Un kernel Linux que sirve como base de la pila de software y se encarga de las funciones más básicas del sistema: gestión de drivers, seguridad, comunicaciones, etc.
- Una capa de librerías de bajo nivel en C y C++.
- Un framework para el desarrollo de aplicaciones.
- Una suite de aplicaciones (navegador, agenda, gestión del teléfono).

Las aplicaciones Android están programadas en lenguaje Java, y corren sobre Dalvik, una máquina virtual Java desarrollada por Google.

Con Android se busca reunir en una misma plataforma todos los elementos necesarios que permitan al desarrollador controlar y aprovechar al máximo cualquier funcionalidad ofrecida por un dispositivo móvil (llamadas, mensajes de texto, cámara, agenda de contactos, conexión Wi-Fi, Bluetooth, aplicaciones ofimáticas, videojuegos, etc.), así como poder crear aplicaciones.

### **2.3.2. Arquitectura por capas Android.**

En las siguientes líneas se dará una visión global por capas de cuál es la arquitectura empleada en Android. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores.

#### **Capa Linux Kernel.**

La capa más inmediata es la corresponde al núcleo de Android. Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que

cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes.

Siempre que un fabricante incluya un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro de este kernel de Linux embebido en el propio Android.

La elección de Linux 2.6 se ha debido principalmente a dos razones: la primera, su naturaleza de código abierto y libre se ajusta al tipo de distribución que se buscaba para Android, la segunda es que este kernel de Linux incluye de por sí numerosos drivers, además de contemplar la gestión de memoria, gestión de procesos, módulos de seguridad, comunicación en red y otras muchas responsabilidades propias de un sistema operativo.

### **Capa Librerías y Android Runtime.**

Las librerías utilizadas por Android están escritas en C/C++. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android.

Entre las librerías más importantes de este nivel, se pueden mencionar las siguientes:

- La librería libc incluye todas las cabeceras y funciones según el estándar del lenguaje C.

- La librería Surface Manager es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.

- OpenGL/SL y SGL representan las librerías gráficas. Android permite gráficos tanto en 2D como en 3D (este último si está disponible en el propio dispositivo móvil), e incluso, una combinación de ambos.

- La librería Media Libraries proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.)

- FreeType, permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.

- La librería SSL posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.

- A través de la librería SQLite, Android ofrece la creación y gestión de bases de datos relacionales, pudiendo transformar estructuras de datos en objetos fáciles de manejar por las aplicaciones.

- La librería WebKit proporciona un motor para las aplicaciones de tipo navegador, y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

Al mismo nivel que las librerías de Android se sitúa el entorno de ejecución. Éste lo constituyen las Core Libraries, que son librerías con multitud de clases de Java, y la máquina virtual Dalvik, de la cual se habla en el siguiente apartado de este capítulo.

Los dos últimos niveles de la arquitectura de Android están escritos enteramente en Java y son los siguientes:



## Capa Framework de aplicaciones

El framework de aplicaciones representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo framework, representado por este nivel.

## Capa Aplicaciones

El último nivel del diseño arquitectónico de Android son las aplicaciones. Éste nivel incluye tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo.

Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.

### 2.3.3. Máquina virtual Dalvik.

En Android, todas las aplicaciones se programan en el lenguaje Java y se ejecutan mediante una máquina virtual de nombre Dalvik, específicamente diseñada para Android. Esta máquina virtual ha sido optimizada y adaptada a las peculiaridades propias de los dispositivos móviles (menor capacidad de proceso, baja memoria, alimentación por batería, etc.) y trabaja con ficheros de extensión .dex (Dalvik Executables). Dalvik no trabaja directamente con el bytecode de Java, sino que lo transforma en un código más eficiente que el original, pensado para procesadores pequeños.

Gracias a la herramienta “dx”, esta transformación es posible: los ficheros .class de Java se compilan en ficheros .dex, de forma que cada fichero .dex puede contener varias clases. Después, este resultado se comprime en un único archivo de extensión .apk (Android Package), que es el que se distribuirá en el dispositivo móvil.

Dalvik permite varias instancias simultáneas de la máquina virtual, y a diferencia de otras máquinas virtuales, está basada en registros y no en pila, lo que implica que las instrucciones son más reducidas y el número de accesos a memoria es menor.

### 2.3.4. Ciclo de vida de las aplicaciones Android.

Cada aplicación Android es ejecutada en su propio proceso y es el propio sistema operativo el encargado de lanzar y parar estos procesos, gestionar su ejecución y decidir qué hacer en función de los recursos disponibles y de las órdenes dadas por el usuario.

El número de procesos posibles serán tantos como permitan los recursos del dispositivo.

El usuario desconoce este comportamiento. Solo es consciente de que mediante un simple clic pasa de una a otra aplicación y puede volver a cualquiera de ellas en el momento que lo desee. No debe preocuparse sobre cuál es la aplicación que realmente está activa, cuánta memoria está consumiendo, ni si existen o no recursos suficientes para abrir una aplicación adicional.

Cada aplicación, está formada por una o varias pantallas, llamadas también componentes Activity o actividades, debido a que estas son implementadas mediante una extensión de la clase Activity. Esta clase permite formar las



**\_ onCreate(), onDestroy():** Cada uno de estos métodos representan el principio y el fin de la actividad.

**\_ onStart(), onStop():** representan la parte visible del ciclo de vida. Desde onStart () hasta onStop (), la actividad será visible para el usuario, aunque es posible que no tenga el foco de acción por existir otras actividades superpuestas con las que el usuario está interactuando. Pueden ser llamados múltiples veces.

**\_ onResume(), onPause():** delimitan la parte útil del ciclo de vida. Desde onResume() hasta onPause(), la actividad no sólo es visible, sino que además tiene el foco de la acción y el usuario puede interactuar con ella.

Tal y como se ve en el diagrama, el proceso que mantiene a esta Activity puede ser eliminado cuando se encuentra en onPause() o en onStop(), es decir, cuando no tiene el foco de la aplicación. Android nunca elimina procesos con los que el usuario está interactuando en ese momento.

Cuando el usuario quiere navegar de una actividad a otra el sistema duerme la actividad actual y realiza una copia de su estado para poder recuperarlo más tarde. Es entonces cuando crea, o despierta si ya existe nueva actividad, asumiendo que existan recursos para ello.

### 2.3.5 Seguridad en Android

En Android cada aplicación se ejecuta en su propio proceso. La mayoría de las medidas de seguridad entre el sistema y las aplicaciones deriva de los estándares de Linux 2.6, cuyo kernel, recuérdese, constituye el núcleo principal de Android.

Cada proceso en Android constituye lo que se llama un cajón de arena o sandbox, que proporciona un entorno seguro de ejecución. Por defecto, ninguna aplicación tiene permiso para realizar ninguna operación o comportamiento que pueda impactar negativamente en la ejecución de otras aplicaciones o del sistema mismo. Por ejemplo, acciones como leer o escribir ficheros privados del usuario (contactos, teléfonos, etc.), leer o escribir ficheros de otras aplicaciones, acceso de red, habilitación de algún recurso hardware del dispositivo, etc., no están permitidas.

La única forma de poder saltar estas restricciones impuestas por Android, es mediante la declaración explícita de un permiso que autorice a llevar a cabo una determinada acción habitualmente prohibida.

Además, en Android es obligatorio que cada aplicación esté firmada digitalmente mediante un certificado, cuya clave privada sea la del desarrollador de dicha aplicación.

No es necesario vincular a una autoridad de certificado, el único cometido del certificado es crear una relación de confianza entre las aplicaciones. Mediante la firma, la aplicación lleva adjunta su autoría.

Para establecer un permiso para una aplicación, es necesario que en el archivo XML llamado AndroidManifest, el cual todas las aplicaciones poseen, se declare uno o más elementos <uses-permission> donde se indique el tipo de permiso que se desea habilitar.

En la aplicación Android del presente proyecto se han habilitado los siguientes permisos:

**android.permission.ACCESS\_COARSE\_LOCATION**

Permite la obtención de la ubicación a través de redes de datos y WiFi

**android.permission.ACCESS\_FINE\_LOCATION**

Permite la localización precisa a través del uso del GPS del teléfono.

**android.permission.INTERNET**

Permite a la aplicación abrir los sockets de red.

### 2.3.6. Componentes de las aplicaciones Android

**Actividades (Activities):** Cada pantalla de una aplicación. Utilizan Vistas (Views) como componentes que muestran información y responden a las acciones del usuario

**Servicios (Services):** Componentes de la aplicación que se ejecutan de forma invisible, actualizando los datos y las Actividades, y disparando notificaciones. Realizan el procesamiento normal de la aplicación que debe continuar incluso cuando las Actividades de la aplicación no están visibles.

**Proveedores de Contenidos (Content Providers):** Almacenes de datos compartidos. Gestionan las Bases de Datos para las aplicaciones.

**Intenciones (Intents):** Mecanismo que permite el paso de mensajes destinados a ciertas Actividades o Servicios, o a todo el sistema (Intenciones de broadcast). Exponen la intención de que se haga algo. El sistema determinara el destinatario que lo efectuara.

**Receptores de Broadcast (Broadcast Receivers):** Los crean las aplicaciones como consumidores de las Intenciones de broadcast que cumplan ciertos criterios.

**Notificaciones (Notifications):** Mecanismo que permite a las aplicaciones señalar algo a los usuarios sin interrumpir la actividad en primer plano.

## Capítulo 3. Creación del entorno

En este apartado se exponen los pasos necesarios para empezar a desarrollar y a entender aplicaciones para Android.

El entorno normal para desarrollar en la plataforma Android es Eclipse, que es el que se usa para generar esta aplicación. Para usarlo, debemos instalar varios complementos que darán la funcionalidad necesaria para trabajar con esta plataforma.

Las instrucciones de instalación aquí descritas se basan en el sistema operativo Windows 7, y en el entorno de desarrollo Eclipse version 3.6.0, conocida como Helios, ya que es software libre multiplataforma, con soporte para numerosos lenguajes de programación, entre ellos Android. Aunque en esta memoria no se contempla, Eclipse y el SDK de android puede correr en otros sistemas operativos, como Mac OS o Linux.

### 3.1. Descargar Eclipse

Desde la página de eclipse, descargaremos la versión Classic a nuestro ordenador. Finalizada la descarga, no se realiza ningún proceso de instalación; simplemente, se deben descomprimir los ficheros y pulsar el ejecutable para iniciar la aplicación. La primera vez que se ejecuta Eclipse, se pide al usuario una localización para crear el workspace, donde se ubicarán por defecto todos los proyectos desarrollados. Tras seleccionar la carpeta en la que guardaremos nuestros trabajos, nos encontramos ante la interfaz de Eclipse:

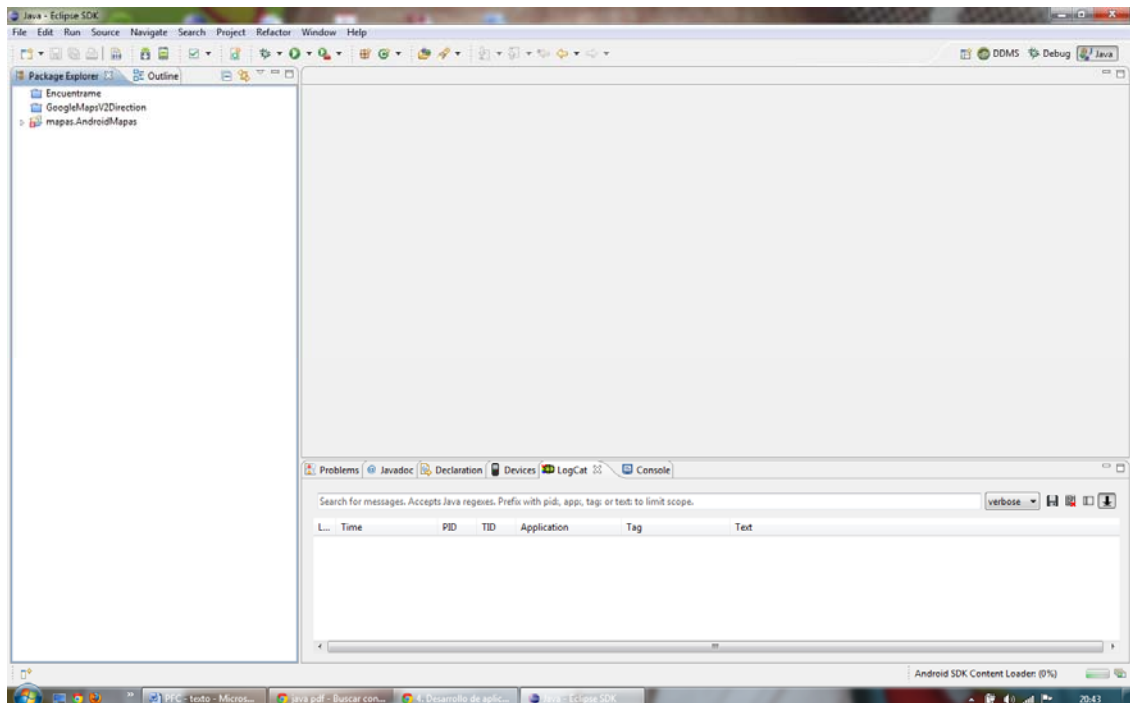


Figura 2. Pantalla principal de Eclipse

## 3.2. Descarga e instalación del SDK de Android

Entramos en la web de desarrolladores de Android, y descargamos el SDK:

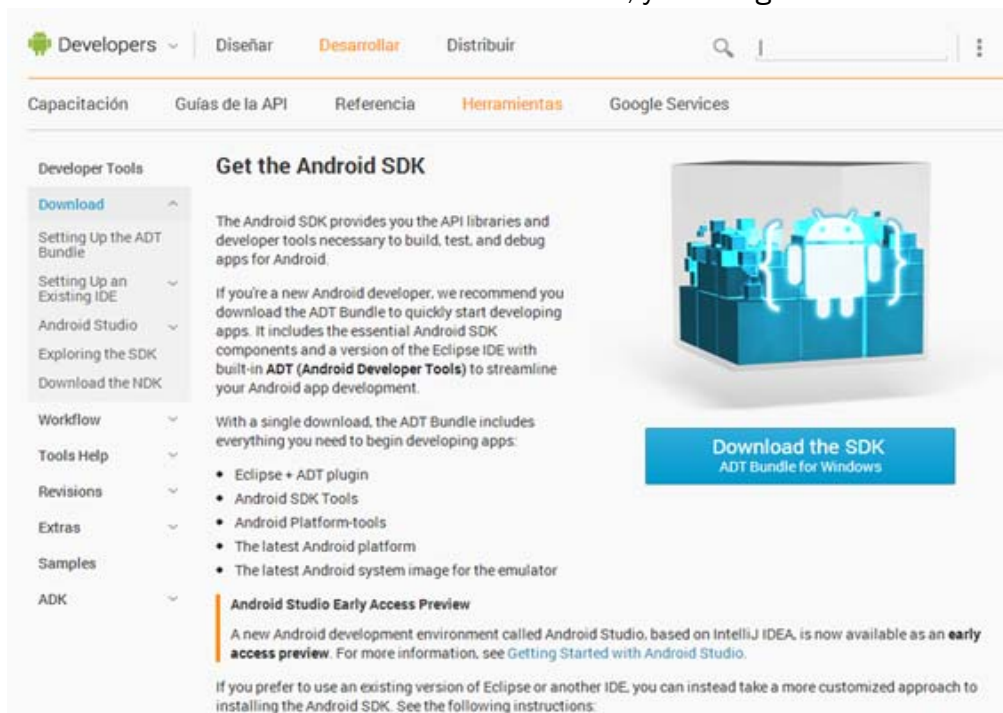


Figura 3. Vista de la web de developers, para bajar el SDK

Se inicia el instalador y se va dando al botón Next cuando corresponda. Debemos tener instalado en el ordenador el JDK de Java, para que el instalador lo detecte. Si no, entrar en la página de Java, y bajar la última versión disponible. Seguimos adelante con la instalación, hasta que aparezca la selección de ruta donde se va a instalar. Se recomienda instalar directamente en la raíz de la unidad de arranque del sistema operativo, para evitar problemas futuros.

Cuando finaliza la instalación, se abre la ventana de "Android SDK Manager". En ella debemos seleccionar las versiones que queremos instalar. Además, por defecto, vienen algunas opciones ya marcadas para instalar, como los drivers de USB para poder probar las aplicaciones en algún dispositivo físico, además de en el emulador del ordenador.

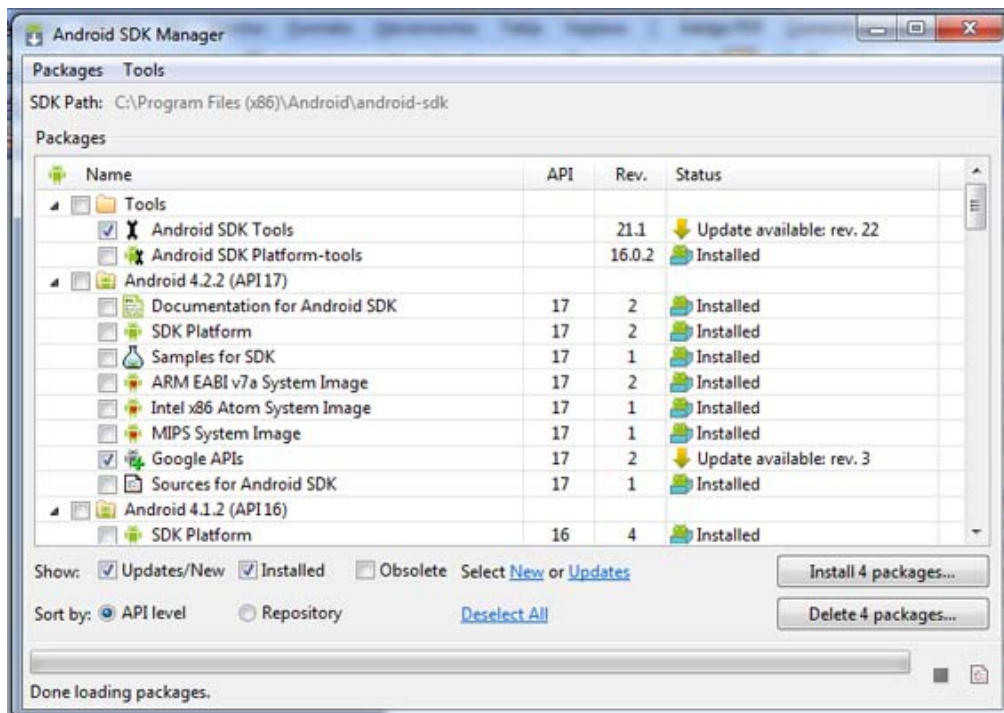


Figura 4. Vista del SDK Manager de Android

Una vez finalice la instalación, tendremos Eclipse y el SDK de Android instalados, aunque estos programas no están vinculados entre sí:

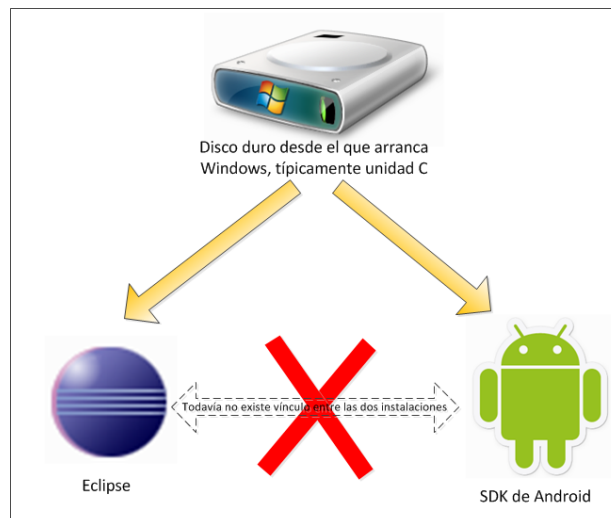


Figura 5. Figura explicativa conexión Eclipse - Android

### 3.3. Vincular Eclipse y SDK

Ahora trabajaremos con Eclipse, hay que vincular el SDK a nuestro entorno de desarrollo. Esto se realiza mediante la instalación de un plugin para Eclipse denominado ADT. Tiene lógica que el plugin sean las herramientas de desarrollo de Android, ya que Eclipse como entorno de trabajo requiere de estas herramientas que no tiene para programar en Android. En el menú vamos a "Help", ahí elegimos "Install New Software..."



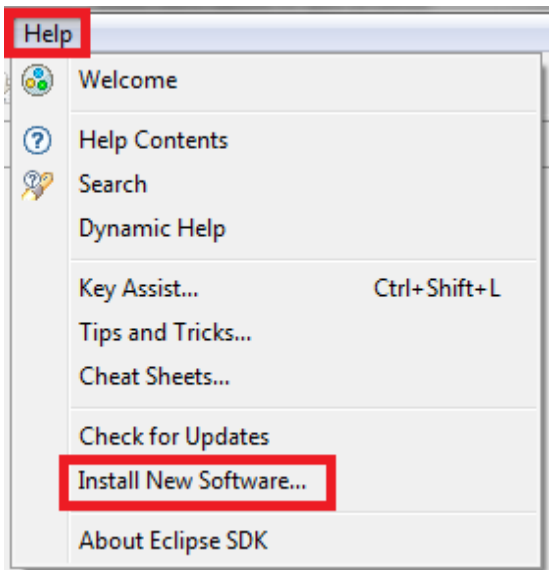


Figura 6. Vista de menú eclipse

Se nos abrirá una nueva ventana. Pulsamos en “Add...”, y en la ventana emergente que nos sale le ponemos el nombre que queremos que tenga el Plugin, por ejemplo: “ADT para vincular con el SDK de Android”. En el campo URL ponemos la dirección del ADT que aparece en la web de desarrolladores de android. Cuando acabemos aceptamos la ventana emergente.

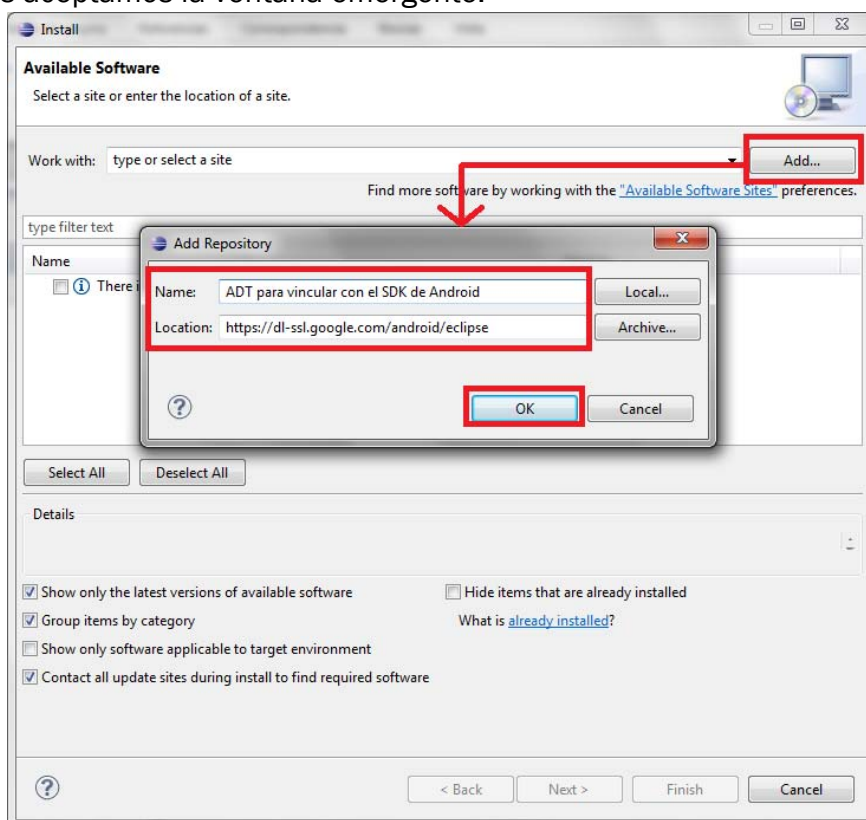


Figura 7. Vista explicativa para la vinculación

Ahora marcamos la casilla de “Developer Tools” para que se marquen todas y pulsamos en siguiente.



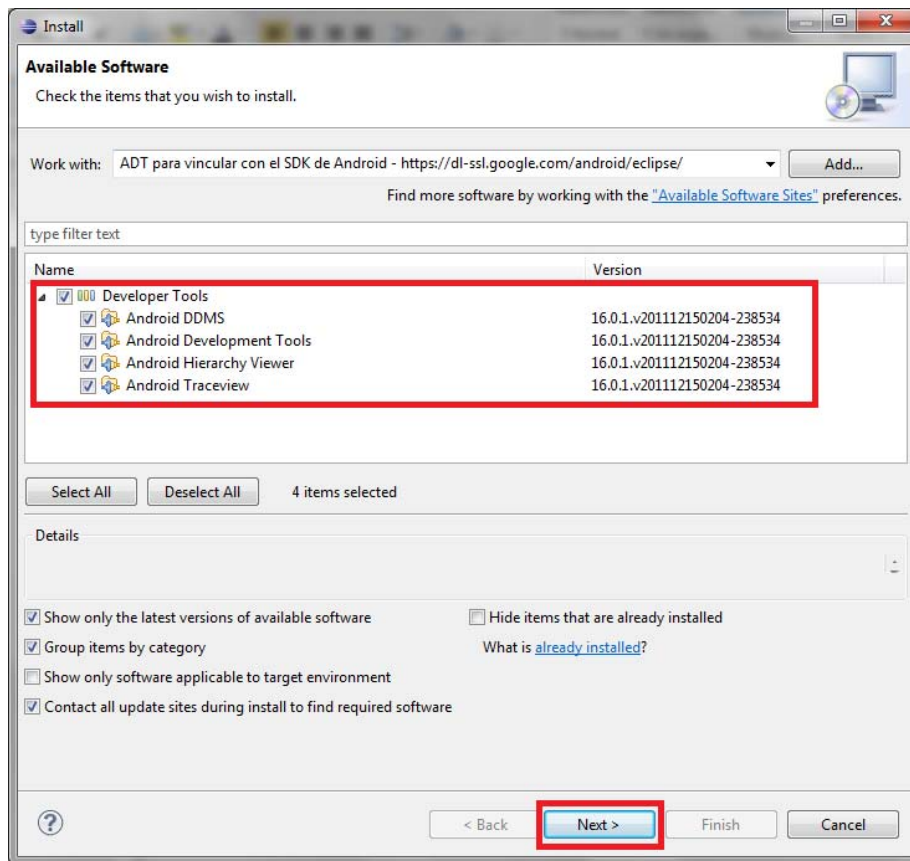


Figura 8. Selección de Developers tolos

En el siguiente paso pulsamos directamente a siguiente.

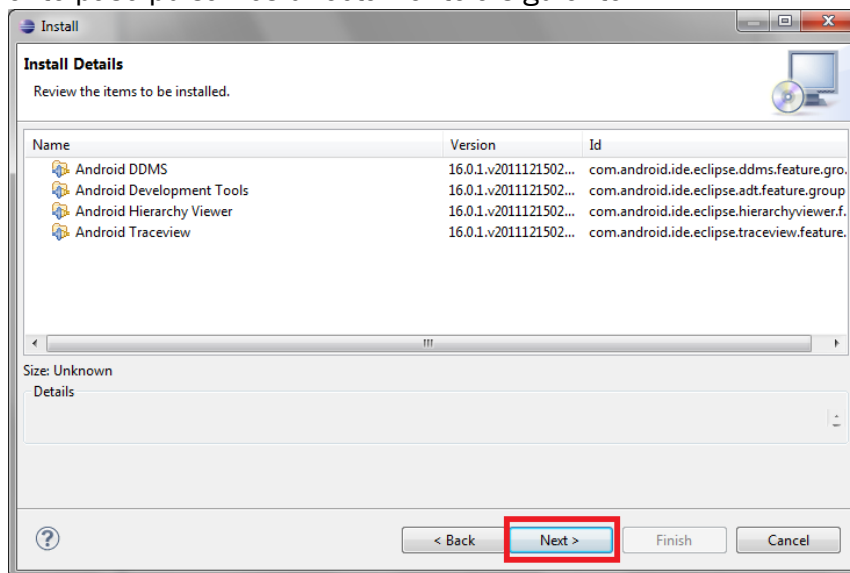


Figura 9. Explicación de instalación

Llegaremos a otro paso que tenemos que aceptar la licencia y pulsamos terminar.

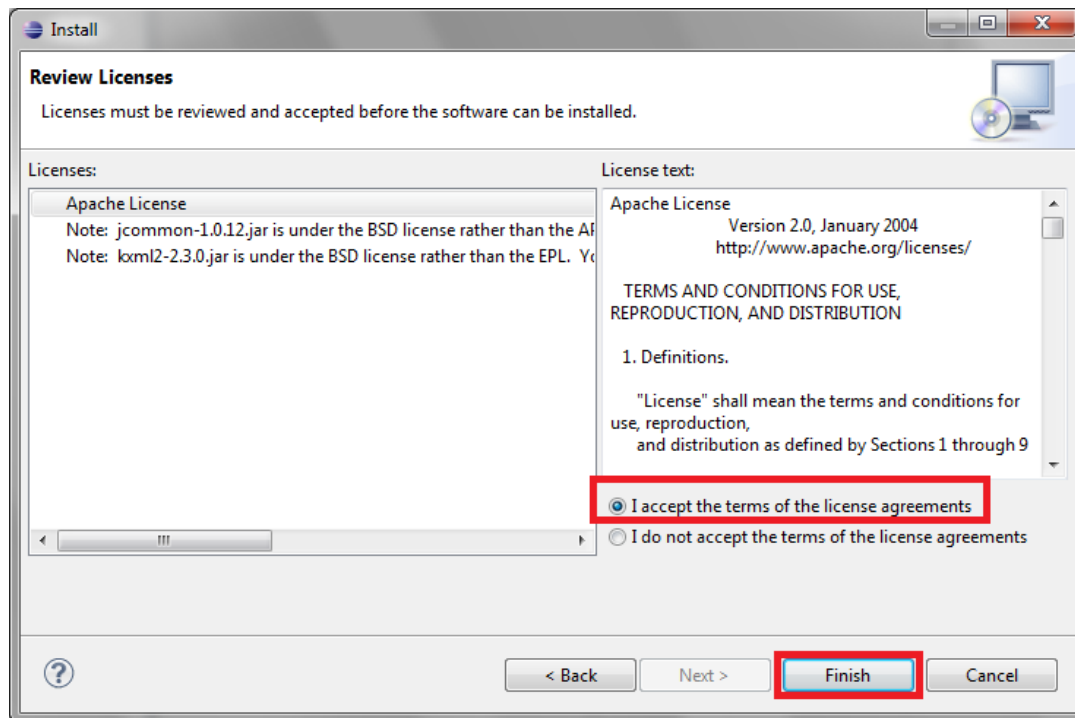


Figura 10. Aceptación de los términos de licencia

Entonces empezará la instalación. Seguramente a mitad de la instalación nos salga una ventana de advertencia avisándonos que existe contenido no firmado y que no puede validarlo, simplemente pulsamos en “OK” y esperamos a que termine.

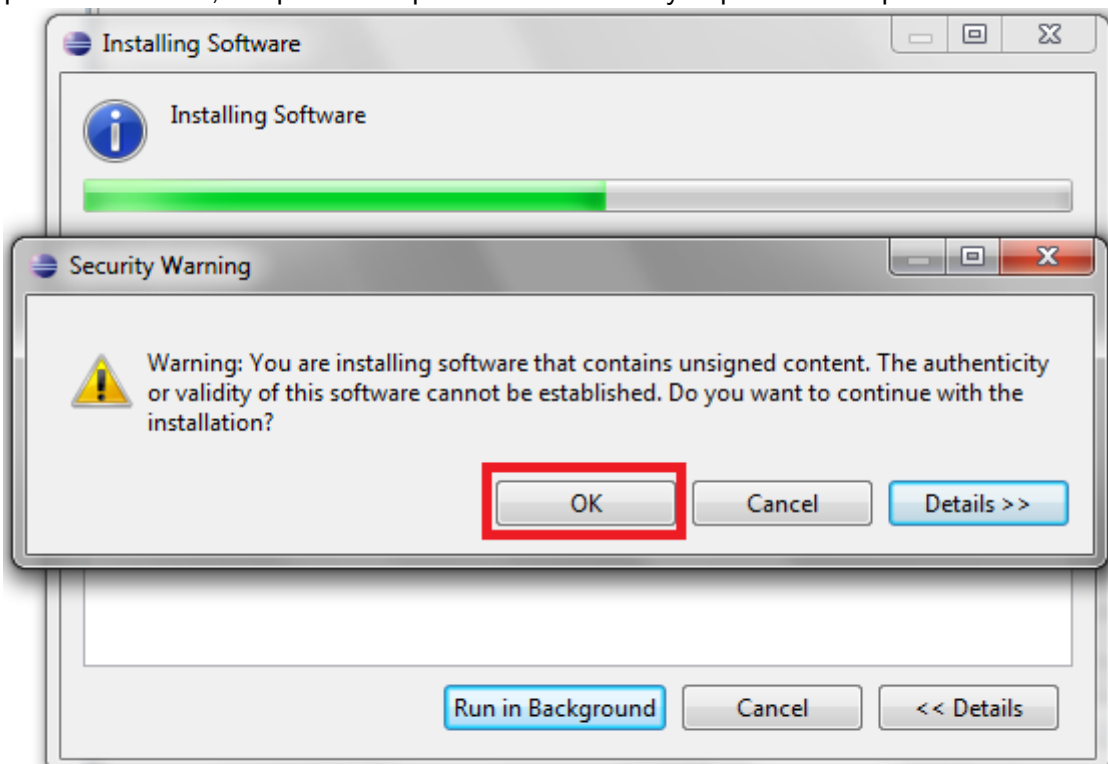


Figura 11. Instalación de contenido no firmado

Al terminar nos pedirá reiniciar Eclipse. Le decimos que reinicie ahora.

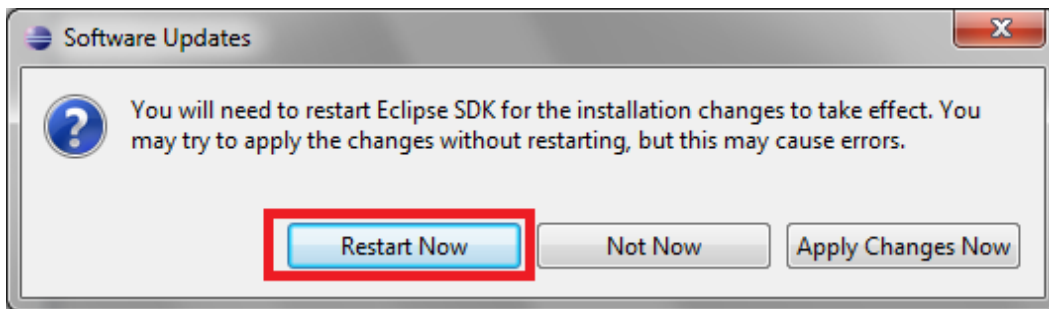


Figura 12. Confirmación de reinicio de eclipse

Al reiniciarse Eclipse nos saldrá una ventana que nos dirá que si queremos instalar el SDK. El SDK ya está instalado, con lo que pulsamos en “Use existing SDKs” y elegimos la carpeta donde instalamos anteriormente el SDK.

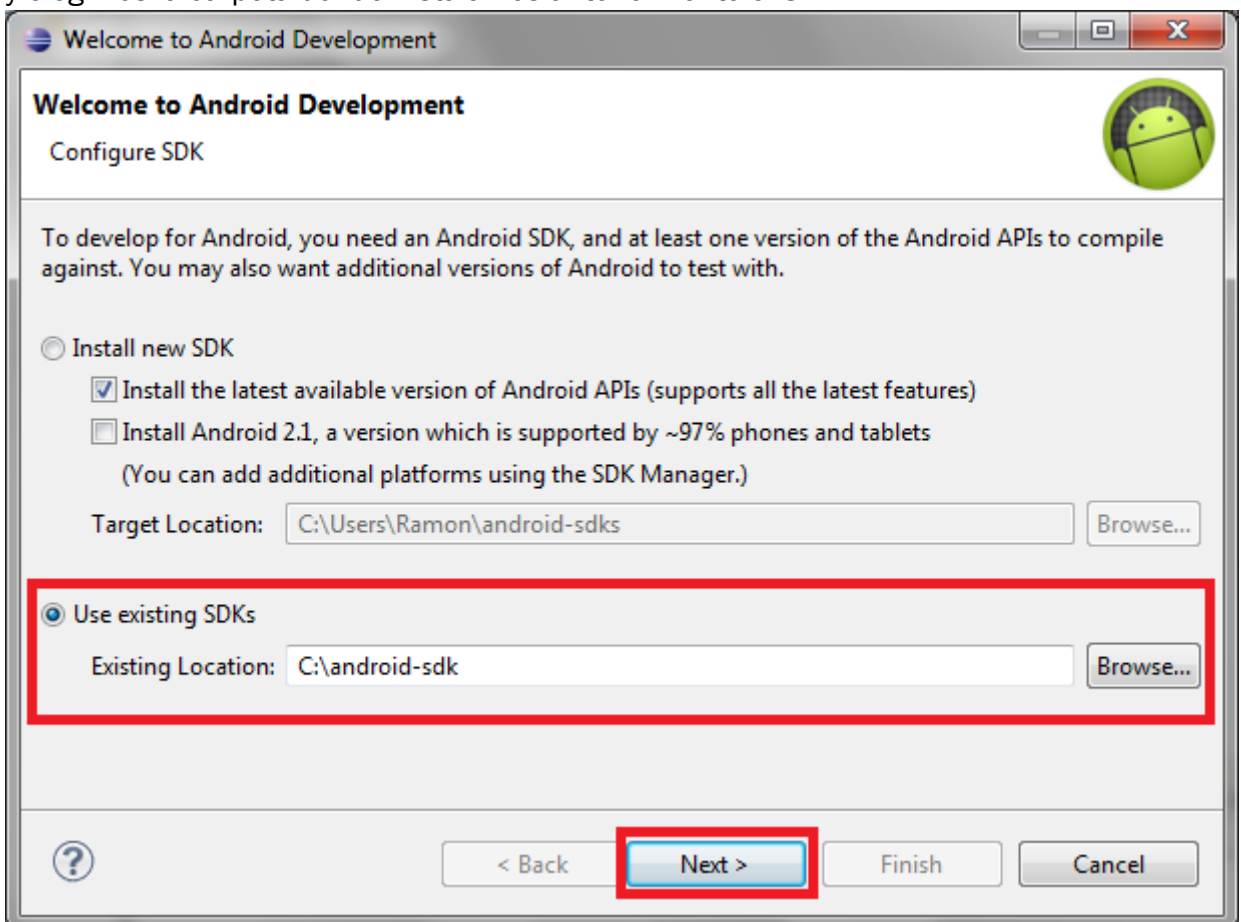


Figura 13. Asociación android-sdk con eclipse

En la siguiente ventana nos preguntará si queremos enviar estadísticas de uso a Google. Si queremos colaborar con Google le decimos que sí, sino que no y pulsamos en terminar.

## Capítulo 4. Uso de la aplicación. Explicación de la implementación

En este capítulo se explica con todo detalle las funcionalidades de la aplicación para la localización y recuperación de puntos de interés mediante terminales android.

### 4.1. Pantalla principal



Figura 14. Vista principal de aplicación

Lo primero que encontramos al abrir la aplicación, es la pantalla que aparece a la derecha, que consta de tres botones:

- **¿Dónde estoy?** El cual, como veremos más adelante, nos ubica mediante GPS y nos da la posibilidad de guardar nuestra aplicación.
- **¡Quiero volver!** En la cual elegimos una ubicación ya guardada y nos muestra la ruta de retorno.
- **Opciones.** Dentro de este menú, nos ofrece otra pantalla con distintos ajustes que detallaremos con posterioridad.

Para el desarrollo de esta pantalla, se ha realizado mediante XML desde eclipse. Además, en la clase principal java, se han creado los botones y, mediante el uso de "Items", hemos saltado de una Activity a otra, según la pulsación usada.

Creación de los botones en java:

```
Button donde;  
Button vuelta;  
Button opciones;  
private MyDBAdapter mdb;
```

Identificar los botones con el id usado en el XML:

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.principal);  
  
    Typeface font = Typeface.createFromAsset(getAssets(), "DroidSans.ttf");  
  
    donde = (Button) findViewById(R.id.button1);  
    vuelta = (Button) findViewById(R.id.button2);  
    opciones = (Button) findViewById(R.id.button4);  
  
    donde.setTypeface(font);  
    vuelta.setTypeface(font);  
    opciones.setTypeface(font);  
}
```

Método `setOnClickListener`, que detecta cuando un botón ha sido pulsado, y nos lleva a otra Activity:

```
donde.setOnClickListener(new View.OnClickListener() {  
  
    @Override  
    public void onClick(View v) {  
        Intent intent = new Intent (AndroidMapas.this, ViewMaps.class);  
        startActivity(intent);  
    }  
});
```

XML, donde se crean los botones de la pantalla principal, con los identificadores vistos anteriormente en el código java.

```
1 <?xml version="1.0" encoding="utf-8"?>  
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
3     android:layout_width="fill_parent"  
4     android:layout_height="fill_parent"  
5     android:orientation="vertical"  
6     android:gravity="center_vertical|center_horizontal"  
7     android:background="@drawable/android">  
8     <Button  
9         android:id="@+id/button1"  
10        style="?android:attr/buttonStyleSmall"  
11        android:layout_width="fill_parent"  
12        android:layout_height="wrap_content"  
13        android:text="@string/donde"  
14        android:layout_margin="20dp"  
15        android:textSize="20sp"  
16        android:textStyle="bold"  
17        android:textColor="@color/azul2"  
18        android:background="@drawable/shape_creditos"/>  
19     <Button  
20        android:id="@+id/button2"  
21        style="?android:attr/buttonStyleSmall"  
22        android:layout_width="fill_parent"  
23        android:layout_height="wrap_content"  
24        android:layout_margin="20dp"  
25        android:text="@string/vuelta"  
26        android:textSize="20sp"  
27        android:textStyle="bold"  
28        android:textColor="@color/azul2"  
29        android:background="@drawable/shape_creditos"/>  
30     <Button  
31        android:id="@+id/button4"  
32        style="?android:attr/buttonStyleSmall"  
33        android:layout_width="fill_parent"  
34        android:layout_height="wrap_content"  
35        android:text="@string/opciones"  
36        android:layout_margin="20dp"  
37        android:textSize="20sp"  
38        android:textStyle="bold"  
39        android:textColor="@color/azul2"  
40        android:background="@drawable/shape_creditos"/>  
41 </LinearLayout>
```

## 4.2. ¿Dónde estoy?

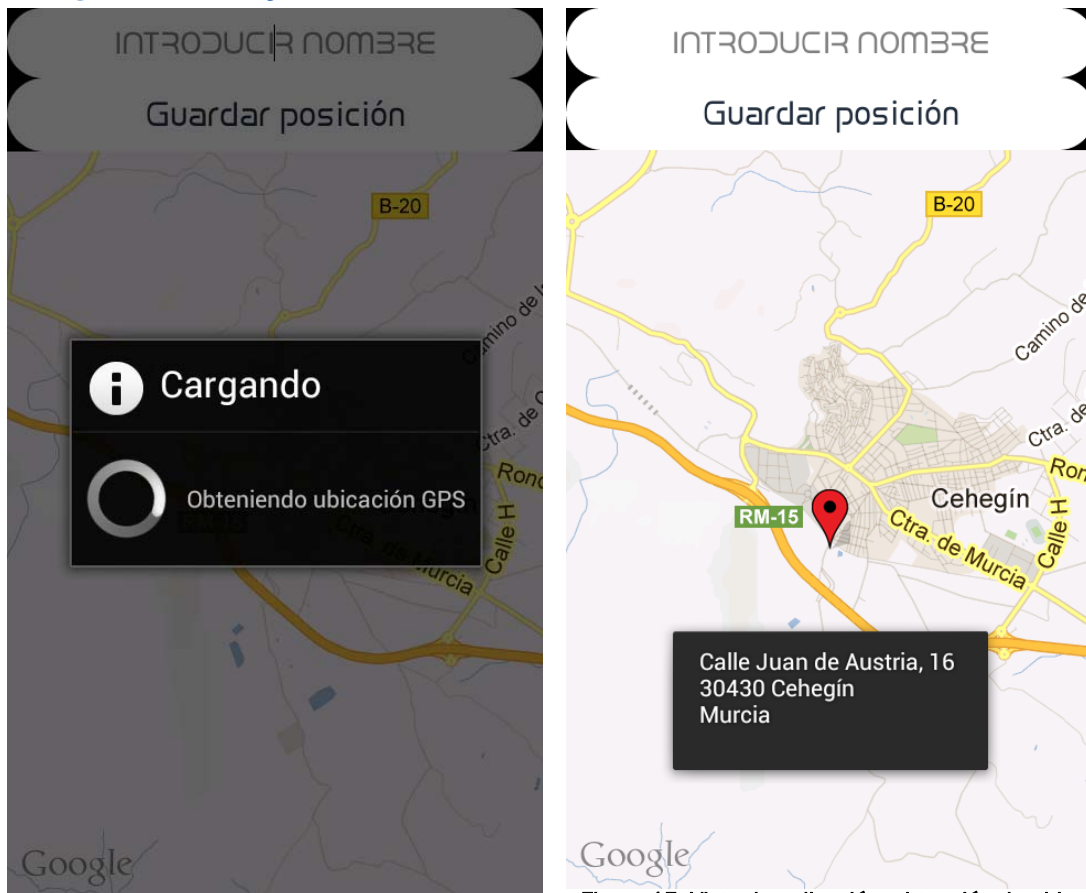


Figura 15. Vista de aplicación, obtención de ubicación

Cuando pulsamos el botón “¿Dónde estoy?”, automáticamente, la aplicación realiza los pasos que vemos arriba. Aunque, antes de todo esto, se ha implementado un método que revisa que la obtención de la ubicación GPS en el teléfono esté disponible y, sino lo está, abre automáticamente los ajustes en esta sección para habilitarlo.

```
//Para tener el GPS funcionando antes de que la aplicación inicie
@Override
public void onProviderDisabled(String provider) {
    Intent intent = new Intent( android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS);
    startActivity(intent);
}
```

Mientras no obtengamos la ubicación GPS, aparecen el diálogo que se ve en la primera de las pantallas, con el mensaje de “Obteniendo ubicación GPS”. Una vez obtengamos el punto, pasa a dibujarse la segunda de las pantallas, y el diálogo de carga desaparece:

```
dialog = new ProgressDialog(ViewMaps.this);
dialog.setTitle("Cargando");
dialog.setMessage("Obteniendo ubicación GPS");
dialog.show();
```

```

latitud=point.getLatitudeE6();
longitud=point.getLongitudeE6();

if (point != null){
    if (dialog != null){
        dialog.dismiss();
    }
}
}

```

Para la obtención de la ubicación, hacemos uso de varias funciones:

```

//Actualizaciones de ubicación
protected void updateLocation(Location location){
    MapView mapView = (MapView) findViewById(R.id.mapa);
    MapController mapController = mapView.getController();

    //Localización!!
    GeoPoint point = new GeoPoint((int) (location.getLatitude() * 1E6), (int) (location.getLongitude() * 1E6));

    mapController.animateTo(point);
    mapController.setZoom(15);

    latitud=point.getLatitudeE6();
    longitud=point.getLongitudeE6();

    if (point != null){
        if (dialog != null){
            dialog.dismiss();
        }
    }

    Geocoder geoCoder = new Geocoder(this, Locale.getDefault());

    try {
        List<Address> addresses = geoCoder.getFromLocation(point.getLatitudeE6() / 1E6, point.getLongitudeE6()

        String address = "";
        if (addresses.size() > 0) {
            for (int i = 0; i < addresses.get(0).getMaxAddressLineIndex(); i++)
                address += addresses.get(0).getAddressLine(i) + "\n";
        }

        Toast.makeText(this, address, Toast.LENGTH_SHORT).show();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

Con un GeoPoint, obtenemos la ubicación GPS actual. Con ese punto, obtenemos la latitud y la longitud, que es lo que guardamos en base de datos.

Para dibujar el punto hacemos uso de la función draw, de la clase MyOverlay, a la que pasamos el punto.

```

class MyOverlay extends Overlay {
    GeoPoint point;

    //El constructor recibe el punto donde se dibujará el marker
    public MyOverlay(GeoPoint point) {
        super();
        this.point = point;
    }

    @Override
    public boolean draw(Canvas canvas, MapView mapView, boolean shadow, long when) {
        super.draw(canvas, mapView, shadow);

        //se traduce el punto geolocalizado a un punto en la pantalla */
        Point scrnPoint = new Point();
        mapView.getProjection().toPixels(this.point, scrnPoint);

        //se construye un bitmap a partir de la imagen
        Bitmap marker = BitmapFactory.decodeResource(getResources(), R.drawable.icon);

        // se dibuja la imagen del marker
        canvas.drawBitmap(marker, scrnPoint.x - marker.getWidth() / 2, scrnPoint.y - marker.getHeight() / 2, null);

        return true;
    }
}

```



Una vez estemos en el punto de guardar ubicación, pueden suceder varias cosas:

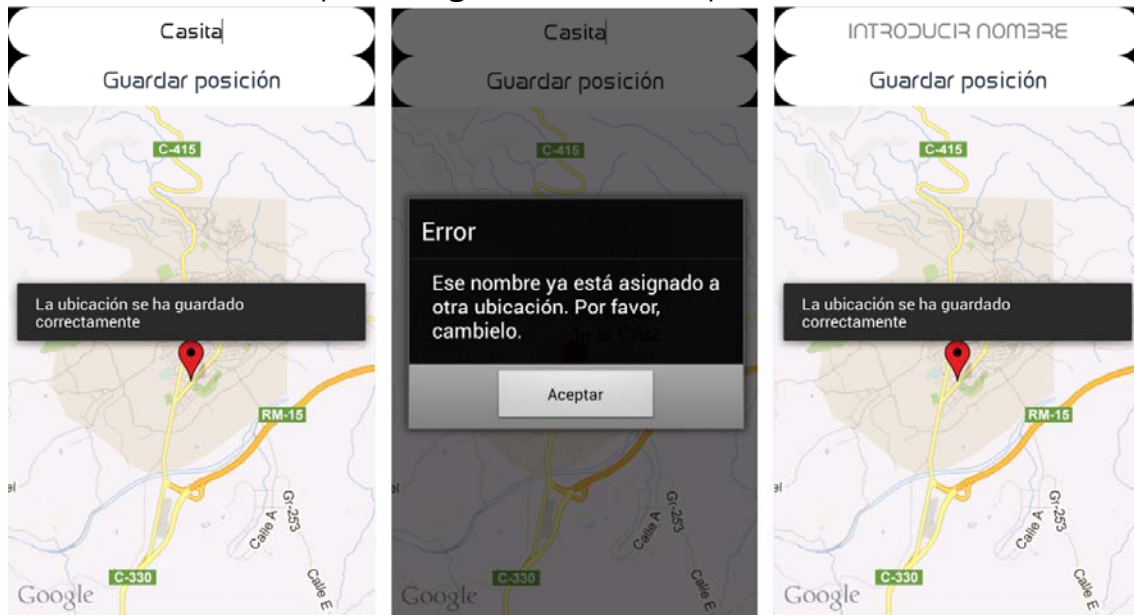


Figura 16. Distintos mensajes al guardar la ubicación

**Figura 1.** Insertamos el nombre de una ubicación, y esta se guarda correctamente en la base de datos.

**Figura 2.** El nombre introducido ya existe, por lo que debemos cambiarlo para poder guardarlo.

**Figura 3.** No introducimos ningún nombre para la ubicación. Ésta se guardará y, por defecto, tendrá como nombre la fecha y hora del momento en que se guardó.

En el XML se define un editText, que es el cuadro donde insertamos el texto, además de un MapView, donde introducimos la API key de google, para la obtención de mapas (Obtención de API key en anexo B)

```
<EditText android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/nombre"
    android:id="@+id/nombre"
    android:background="@drawable/shape_creditos"
    android:gravity="center"/>

<com.google.android.maps.MapView
    android:id="@+id/mapa"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:apiKey="0iSsIHgJAaBkNRnM8H69nyVD8JSdYCRYD3MN7vg"
    android:clickable="true" />
```





### 4.3. ¡Quiero volver!

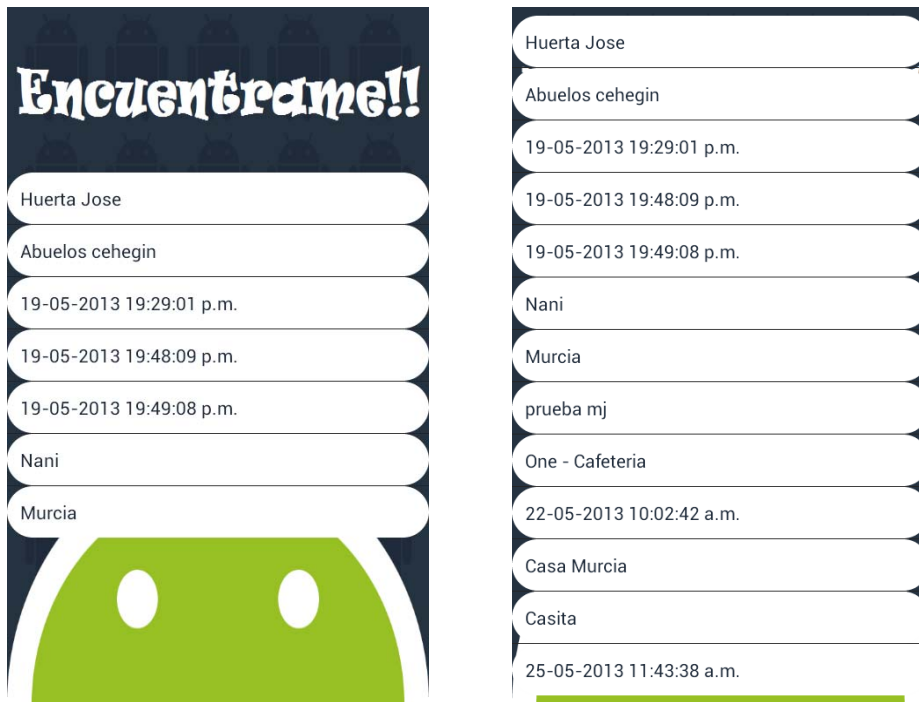


Figura 17. Vista dentro del menú “¡Quiero volver!”

Cuando entramos en esta parte de la aplicación, obtenemos un listado de todas las ubicaciones que tenemos (como se ve en los dos ejemplos de arriba). Podremos desplazarnos hacia arriba y abajo, según la cantidad de ubicaciones que tengamos, y pulsar sobre la deseada para obtener la ruta de vuelta a ella.

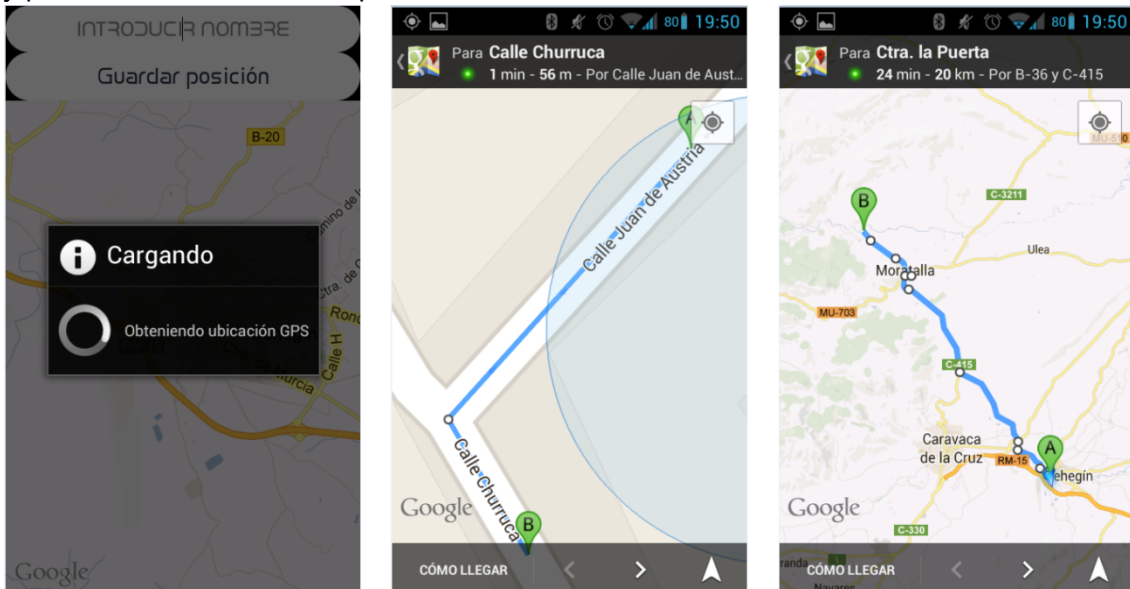


Figura 18. Vista de distintas fases de obtenciones de rutas

El dialogo de carga de la señal GPS funciona igual que en la pantalla anterior. Para la obtención de la ruta, buscamos en nuestra base de datos el punto seleccionado y después, a través de un “Intent”, la pasamos a la aplicación “Mapas” de nuestro teléfono, la cual nos devuelve la ruta dibujada tal y como vemos en los dos ejemplos de la derecha.

```

mapController.animateTo(point);
mapController.setZoom(15);
//PUNTO ACTUAL
latitud=point.getLatitudeE6()/1E6;
longitud=point.getLongitudeE6()/1E6;
//PUNTO RETORNO
latitud2=Double.parseDouble(ubicaciones[1])/1E6;
longitud2=Double.parseDouble(ubicaciones[2])/1E6;

GeoPoint point2 = new GeoPoint ((int)(latitud2*1E6), (int)(longitud2*1E6));

if (point != null){
    if (dialog != null){
        dialog.dismiss();
    }
}

List<Overlay> mapOverlays = mapView.getOverlays();
MyOverlay marker = new MyOverlay(point, point2);
mapOverlays.add(marker);
mapView.invalidate();

//Salto a google maps con las ubicaciones, para dibujar la ruta
Intent intent = new Intent(Intent.ACTION_VIEW,Uri.parse("http://maps.google.com/maps?" + "saddr="+
    latitud + "," + longitud + "saddr=" + latitud2 + "," + longitud2));
intent.setClassName("com.google.android.apps.maps","com.google.android.maps.MapsActivity");
startActivity(intent);
}

```

Para la obtención de las ubicaciones de la lista, usamos una clase intermedia, donde obtenemos todas las que tenemos guardadas en base de datos, y la enviamos a través de un "Intent" a la clase final, donde le damos el uso deseado.

```

intent=getIntent();
if (intent!=null){
    items=intent.getStringArrayListExtra("listadonombres");
    System.out.println(items.size());
    al = new AdaptadorLista(Elegir.this, R.layout.estilo,items);
    lista.setAdapter(al);
    lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {

        @Override
        public void onItemClick(AdapterView<?> arg0, View vista,
            final int position, long arg3) {
            // TODO Auto-generated method stub
            //IMPLEMENTAR BORRAR AQUI
            mdb.open();
            Cursor c1;
            String name, latitude, longitude;
            System.out.println(items.get(position).toString());
            c1=mdb.getAllEntries();
            c1.moveToPosition(position);
            name=c1.getString(c1.getColumnIndex("name"));
            latitude=c1.getString(c1.getColumnIndex("latitude"));
            longitude=c1.getString(c1.getColumnIndex("longitude"));
            System.out.println("name = "+name);
            System.out.println("latitude = "+latitude);
            System.out.println("longitude = "+longitude);
            String [] ubicacion = {name, latitude, longitude};

            Intent intent = new Intent (Elegir.this, ViewReturn.class);
            intent.putExtra("ubicaciones", ubicacion);
            startActivity(intent);
            mdb.close();
            c1.close();

        }

    });
}

```

## 4.4. Opciones



Figura 19. Menú Opciones

Esta parte de la ubicación tiene la funcionalidad de servir de ajustes para las ubicaciones guardadas. Existen tres opciones en este menú:

- **Borrar todo.** Borra todas las ubicaciones, y deja la aplicación como el primer uso.
- **Borrar.** Podemos elegir una ubicación y borrarla.
- **Editar.** Nos da la opción de cambiarle el nombre a una ubicación almacenada.

### 4.4.1. Borrar todo.

Se ha pensado que, después de volver de un viaje, todas las ubicaciones que tuviéramos guardadas ya no nos serían de ninguna utilidad. Por eso, se implementa la opción de borrar todas las ubicaciones y así, poder comenzar desde cero y que el listado actual no sea demasiado largo, de forma que se vuelva tediosa la búsqueda entre muchas.

```
borrartodo.setOnClickListener(new View.OnClickListener() {  
  
    @Override  
    public void onClick(View v) {  
        mdb = new MyDBAdapter(Opciones.this);  
        mdb.open();  
  
        final Cursor listado = mdb.getNombre();  
  
        if (listado.moveToFirst()) {  
            AlertDialog.Builder dialogol = new AlertDialog.Builder(Opciones.this);  
            dialogol.setTitle("Importante");  
            dialogol.setMessage("¿Realmente desea borrar todas las ubicaciones? Si le da a Aceptar, perderá to");  
            dialogol.setCancelable(false);  
            dialogol.setPositiveButton("Confirmar", new DialogInterface.OnClickListener() {  
                public void onClick(DialogInterface dialogol, int id) {  
                    do {  
                        mdb.removeEntry(listado.getString(listado.getColumnIndex("name")));  
                    } while (listado.moveToNext());  
  
                    mdb.close();  
                    String mensaje = "Se han borrado todas las ubicaciones";  
                    Toast.makeText(Opciones.this, mensaje, Toast.LENGTH_SHORT).show();  
                }  
            });  
            dialogol.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {  
                public void onClick(DialogInterface dialogol, int id) {  
                }  
            });  
            dialogol.show();  
        }  
    }  
});
```

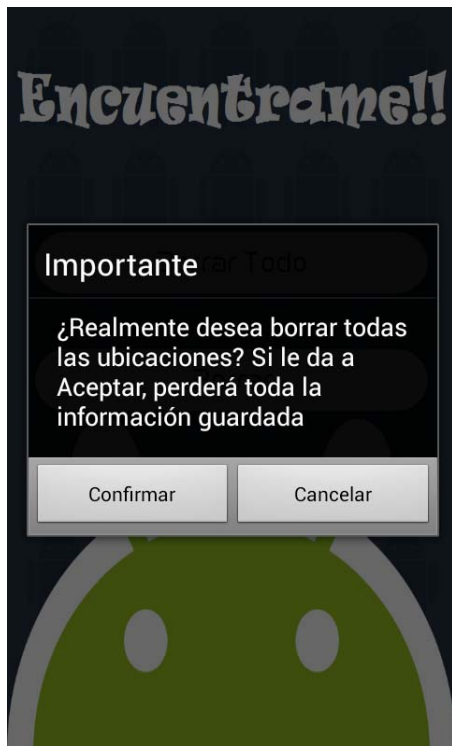


Figura 20. Solicitud de confirmación al borrar todas las ubicaciones

Se ha puesto un dialogo pidiendo confirmación, por si se realiza alguna pulsación por error, no perdamos toda la información guardada por un despiste.

Después, cuando ha finalizado el borrado, aparece un mensaje de tipo Toast, confirmando el borrado. La característica de este mensaje es que permanece de forma temporal en la pantalla, desapareciendo a los pocos segundos de mostrarse.

#### 4.4.2. Borrar

En esta parte de la implementación, debemos elegir una ubicación de la lista, ya sea que la hayamos guardado por error o, simplemente, que ya no nos es necesaria, y después de pedir confirmación, se borrará.

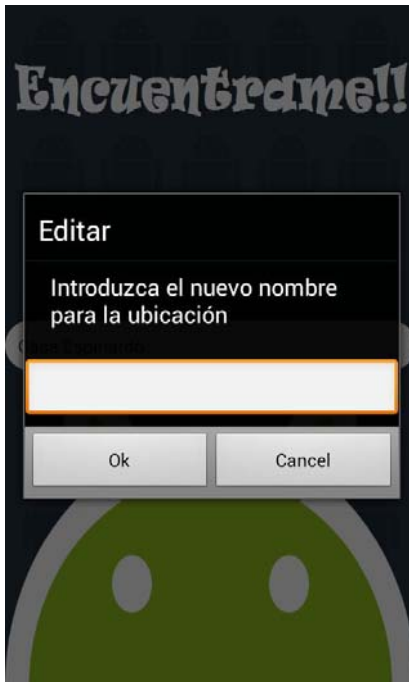
```

intent=getIntent();
if (intent!=null){
    items=intent.getStringArrayListExtra("listadonombres");
    System.out.println(items.size());
    al = new AdaptadorLista(Borrar.this, R.layout.estilo,items);
    lista.setAdapter(al);
    lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {

        @Override
        public void onItemClick(AdapterView<?> arg0, View vista,
            final int position, long arg3) {
            // TODO Auto-generated method stub
            //IMPLEMENTAR BORRAR AQUI
            System.out.println(items.get(position).toString());
            AlertDialog.Builder dialogol = new AlertDialog.Builder(Borrar.this);
            dialogol.setTitle("Importante");
            dialogol.setMessage("¿Realmente desea borrar la ubicación: "+items.get(position).toString()+"?");
            dialogol.setCancelable(false);
            dialogol.setPositiveButton("Confirmar", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialogol, int id) {
                    mdb.open();
                    if(mdb.removeEntry(items.get(position).toString())){
                        System.out.println(items.get(position).toString());
                        items.remove(position);
                        al.notifyDataSetChanged();
                        if(items.size()==0){
                            finish();
                        }
                    }
                    mdb.close();
                }
            });
            dialogol.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialogol, int id) {
            }
        });
    });
}

```

### 4.4.3. Editar



En este apartado, podemos modificar el nombre de la ubicación. Puede que al guardarla, con las prisas, introdujésemos mal el nombre o que le diésemos al guardado por defecto, con fecha y hora, y más tarde quisiéramos darle un nombre.

Figura 21. Vista de editar nombre, dentro del menú Opciones

```
@Override
public void onItemClick(AdapterView<?> arg0, View vista,
    final int position, long arg3) {
    // TODO Auto-generated method stub
    //IMPLEMENTAR BORRAR AQUI
    System.out.println(items.get(position).toString());
    AlertDialog.Builder dialogo1 = new AlertDialog.Builder(Editar.this);
    dialogo1.setTitle("Importante");
    dialogo1.setMessage(";Realmente desea editar la ubicación: "+items.get(position).toString()+"?");
    dialogo1.setCancelable(false);
    dialogo1.setPositiveButton("Confirmar", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialogo1, int id) {
            AlertDialog.Builder alert = new AlertDialog.Builder(Editar.this);

            alert.setTitle("Editar");
            alert.setMessage("Introduzca el nuevo nombre para la ubicación");

            // Set an EditText view to get user input
            final EditText input = new EditText(Editar.this);
            alert.setView(input);

            alert.setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int whichButton) {
                    String value = input.getText().toString();
                    System.out.println("value="+value);
                    mdb.open();
                    int conteo = mdb.updateEntry(items.get(position).toString(), value);
                    System.out.println ("conteo="+conteo);
                    items.set(position, value);
                    al.notifyDataSetInvalidated();
                    mdb.close();

                    // Do something with value!
                }
            });
        }
    });
}
```

## 4.5. Otras implementaciones de interés

Aunque no se ha nombrado en los puntos anteriores, durante toda la aplicación se controlan distintos aspectos que se han de tener en cuenta a la hora de desarrollarla.

### 4.5.1. Pantallas de control



Se controla que el nombre de la ubicación no exista, ya que en la base de datos, el nombre corresponde a la “Primary Key”

Figura 22. Error debido a que el nombre ya existe



También, a la hora de obtener alguno de los listados (Quiero volver, borrar, borrar todo...) revisa que hayan ubicaciones en la base de datos. De no ser así, muestra un mensaje tal y como se ve en la imagen de la izquierda.

Figura 23. Error debido a que no existe ninguna ubicación almacenada



#### 4.5.2. Interfaz gráfica

Para la interfaz gráfica, se ha creado un XML donde se ha definido el estilo de la aplicación. Principalmente, se define el estilo de los botones.

```
<?xml version="1.0" encoding="utf-8"?>
<selector
  xmlns:android="http://schemas.android.com/apk/res/android">
  <item>
    <shape>
      <corners
        android:radius="50dp" />
      <gradient
        android:endColor="@color/blanco"
        android:startColor="@color/blanco"/>
      <padding
        android:left="10dp"
        android:top="10dp"
        android:right="10dp"
        android:bottom="10dp" />
    </shape>
  </item>
</selector>
```

Cada XML definido en la aplicación, llama a este, para coger sus propiedades:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical"
  android:gravity="center_vertical|center_horizontal"
  android:background="@drawable/shape_creditos" >
  <TextView
    android:id="@+id/estilo"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/azul2"/>
</LinearLayout>
```

También, se han definido varios XML con propiedades a las que llamamos según necesitemos. Contienen strings y colores que nos son necesarios durante la implementación:

#### Colores.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="azul">#0000FF</color>
  <color name="rojo">#FF0000</color>
  <color name="verde">#00FF00</color>
  <color name="amarillo">#FFFF00</color>
  <color name="gris_oscuro">#111111</color>
  <color name="rosa">#FF0066</color>
  <color name="blanco">#FFFFFF</color>
  <color name="azul2">#1f2b3a</color>
</resources>
```

#### Strings.xml:



```

<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Encuentrame!</string>
  <string name="app_name">Encuentrame!</string>
  <string name="obtener">Recuperar Ubicación</string>
  <string name="donde">¿Dónde estoy?</string>
  <string name="vuelta">¡Quiero volver!</string>
  <string name="guardar">Guardar posición</string>
  <string name="nombre">INTRODUCIR NOMBRE</string>
  <string name="opciones">Opciones</string>
  <string name="borrar">Borrar</string>
  <string name="borrartodo">Borrar Todo</string>
  <string name="editar">Editar</string>
  <string name="cargando">Obteniendo Ubicación</string>
  <string name="vacía">No hay datos</string>
</resources>

```

### 4.5.3. Android Manifest

Aunque no lo hemos nombrado con anterioridad, existe un archivo muy importante en la programación Android: el AndroidManifest. Cada aplicación debe tener un archivo AndroidManifest.xml (precisamente con ese nombre) en su directorio raíz. El manifiesto contiene información esencial acerca de la aplicación para el sistema Android, el sistema de información debe tener antes de que pueda ejecutar cualquiera de código de la aplicación. Entre otras cosas, el manifiesto hace lo siguiente:

- Aquí se pone el nombre del paquete para Java. El nombre del paquete sirve como identificador único para la aplicación.
- En él se describen los componentes de la aplicación - las Activities, servicios, receptores de radiodifusión y los proveedores por los que está compuesta la aplicación. Se nombra a las clases que implementan cada uno de los componentes y publica sus capacidades (por ejemplo, los mensajes "Intent" que pueden manejar). Estas declaraciones hacen que el sistema Android sepa cuáles son los componentes y bajo qué condiciones se puede iniciar.
- Determina qué procesos que componen la aplicación.
- Se declara los permisos que debe tener la aplicación para poder acceder a las partes protegidas de la API y de interactuar con otras aplicaciones.
- También declara los permisos que los demás están obligados a tener con el fin de interactuar con los componentes de la aplicación.
- En él se enumeran las clases de "Instrumentación" que proporcionan perfiles y otra información que la aplicación se está ejecutando. Estas declaraciones están presentes en el Manifest sólo mientras se está desarrollando y probando la aplicación; dichas declaraciones se borran antes de su publicación.
- Declara el nivel mínimo de la API de Android que la aplicación requiere.
- En él se enumeran las librerías a las que debe estar vinculada la aplicación.

En el Anexo A, además del código de la aplicación, se adjunta el AndroidManifest.

## Capítulo 5. Conclusiones y trabajo futuro

El motivo de crear esta aplicación en un entorno Android, se debe principalmente a mi curiosidad por este sistema operativo. A lo largo de los estudios de la carrera (Telemática), se han estudiado muchos y variados lenguajes de programación, sin llegar a profundizar demasiado en ninguno de ellos.

Java es uno de los lenguajes de programación más utilizados actualmente en el desarrollo de muchas aplicaciones y se usa en multitud de empresas. Mi curiosidad por este lenguaje hizo que quisiera profundizar más, y por eso, Android fue una buena herramienta para ello.

La verdad es que el proyecto superó mis expectativas y, a pesar de algunos malos ratos, el resultado ha sido bastante satisfactorio. Esta aplicación toca algo de los rasgos más usados en cualquier aplicación (Activities, interfaz gráfica, base de datos...) y me ha ayudado a aprender y mejorar en muchas cosas. No descarto realizar más aplicaciones en el futuro, aunque sean sólo para mi propio interés o para seguir satisfaciendo mi propia curiosidad a la hora de programar.

### 5.1. Posibles mejoras y trabajo futuro

A la finalización de esta aplicación ha quedado patente que es capaz de gestionar ubicaciones de manera rápida y cómoda. Además, creo que la interfaz es accesible para cualquier usuario, siendo bastante simple y manejable en su uso. Aunque, como toda aplicación informática, es de lógica pensar que siempre hay puntos susceptibles de mejora así como nuevas funcionalidades que se puedan añadir. En este punto se van a presentar dichas posibles mejoras, que no se descartan que se lleven a cabo en un futuro próximo.

La primera mejora consistiría en que la propia aplicación dibujase los mapas. Actualmente, la aplicación salta a la aplicación "Mapas" instalada por defecto en el teléfono, con la ruta dibujada. En un futuro, se quiere lograr que la aplicación sea autónoma y consiga realizar todas sus funciones por ella misma, sin hacer llamadas a aplicaciones externas.

Otra mejora sería la de incluir alguna entrada para indicar si la ruta se quiere hacer en coche o a pie, ya que en estos momentos está definida por momento la ruta en coche. En principio, debería ser igual el uso de una ruta u otra, pero se ha comprobado que para puntos cercanos, si la calle es dirección prohibida, puede hacer que el camino a seguir, si es a pie, sea más largo de lo que podría ser si se optimizara la ruta.

Por último, me gustaría integrar alguna forma de interactividad social. Que sea posible compartir la ubicación por algunas redes sociales, tales como Facebook y Twitter, o que sea posible mandar la ubicación por Whatsapp o correo electrónico, por si a algún conocido le puede interesar. Para esto último, además, la aplicación debe estar preparada para añadir a la lista de ubicaciones guardadas una que hayamos recibido por alguno de estos medios.

## ANEXO A. Código de la aplicación

### A.1. Adaptador Lista

```
package mapas.android;
import java.util.List;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;
public class AdaptadorLista extends ArrayAdapter{
Context context;
int text;
List obj;
public AdaptadorLista(Context context, int textViewResourceId, List
objects) {
super(context, textViewResourceId, objects);
// TODO Auto-generated constructor stub
this.context = context;
this.text = textViewResourceId;
this.obj = objects;
}
@Override
public View getView(int position, View convertView, ViewGroup parent)
{
// TODO Auto-generated method stub
if(convertView!=null){
TextView lista = (TextView) convertView.findViewById(R.id.estilo);
lista.setText(obj.get(position).toString());
return convertView;
}else{
LayoutInflater layout=(LayoutInflater)
getContext().getSystemService(Context.
LAYOUT_INFLATER_SERVICE);
convertView=layout.inflate(text,null);
TextView lista = (TextView) convertView.findViewById(R.id.estilo);
lista.setText(obj.get(position).toString());
return convertView;
}
}
}
```

## A.2. AndroidMapas

```
package mapas.android;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
//import mapas.android.*;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
//import android.content.Context;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import android.graphics.Typeface;
//import android.graphics.Typeface;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
//import android.location.LocationManager;
public class AndroidMapas extends MapActivity implements
LocationListener {
    Button donde;
    Button vuelta;
    Button opciones;
    private MyDBAdapter mdb;
    //Typeface fuente = Typeface.createFromAsset(getAssets(),
    "DroidSans.ttf");
    class MyOverlay extends Overlay {
        GeoPoint point;
        //El constructor recibe el punto donde se dibujará el marker
        public MyOverlay(GeoPoint point) {
            super();
            this.point = point;
        }
        @Override
        public boolean draw(Canvas canvas, MapView mapView, boolean shadow,
        long when) {
            super.draw(canvas, mapView, shadow);
            //se traduce el punto geolocalizado a un punto en la pantalla */
            Point scrnPoint = new Point();
            mapView.getProjection().toPixels(this.point, scrnPoint);
            //se construye un bitmap a partir de la imagen
            Bitmap marker = BitmapFactory.decodeResource(mapView.getResources(),
            R.drawable.
            icon);
            // se dibuja la imagen del marker
            canvas.drawBitmap(marker, scrnPoint.x - marker.getWidth() / 2,
            scrnPoint.y -
```

```

marker.getHeight() / 2, null);
return true;
}
}
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.principal);
Typeface font = Typeface.createFromAsset(getAssets(),
"DroidSans.ttf");
donde = (Button) findViewById(R.id.button1);
vuelta = (Button) findViewById(R.id.button2);
opciones = (Button) findViewById(R.id.button4);
donde.setTypeface(font);
vuelta.setTypeface(font);
opciones.setTypeface(font);
donde.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Intent intent = new Intent (AndroidMapas.this, ViewMaps.class);
startActivity(intent);
}
});
vuelta.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
mdb = new MyDBAdapter(AndroidMapas.this);
mdb.open();
ArrayList lista= new ArrayList<String>();
Cursor listado = mdb.getNombre();
if (listado.moveToFirst()){
listado.moveToFirst();
String c1 ;
do{
c1 = listado.getString(listado.getColumnIndex("name"));
lista.add(c1);
System.out.println(c1);
}while (listado.moveToNext());
Intent intent = new Intent (AndroidMapas.this, Elegir.class);
intent.putExtra("listadonombres", lista);
startActivity(intent);
mdb.close();
/*Intent intent = new Intent (Opciones.this, Borrar.class);
startActivity(intent);*/
}else{
System.out.println("No hay ninguna ubicación guardada");
AlertDialog.Builder builder = new AlertDialog.Builder(AndroidMapas.
this);
builder.setMessage("No hay ninguna ubicación guardada")
.setTitle("Error")
.setCancelable(false)
.setNeutralButton("Aceptar",
new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int
id) {
dialog.cancel();
}
});
AlertDialog alert = builder.create();
alert.show();
}
}
}

```

```

    }
    });
    opciones.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
    Intent intent = new Intent (AndroidMapas.this, Opciones.class);
    startActivity(intent);
    }
    });
    }
    @Override
    protected boolean isRouteDisplayed() {
    return false;
    }
    @Override
    public void onLocationChanged(Location location) {
    updateLocation(location);
    }
    //Para tener el GPS funcionando antes de que la aplicación inicie
    @Override
    public void onProviderDisabled(String provider) {
    Intent intent = new Intent(
    android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS
    );
    startActivity(intent);
    }
    @Override
    public void onProviderEnabled(String provider) {
    // TODO Auto-generated method stub
    }
    @Override
    public void onStatusChanged(String provider, int status, Bundle
    extras) {
    // TODO Auto-generated method stub
    }
    //Actualizaciones de ubicación
    protected void updateLocation(Location location){
    MapView mapView = (MapView) findViewById(R.id.mapa);
    MapController mapController = mapView.getController();
    //Localización!!
    GeoPoint point = new GeoPoint((int) (location.getLatitude() * 1E6),
    (int) (location.
    getLongitude() * 1E6));
    mapController.animateTo(point);
    mapController.setZoom(15);
    Geocoder geoCoder = new Geocoder(this, Locale.getDefault());
    try {
    List<Address> addresses =
    geoCoder.getFromLocation(point.getLatitudeE6() / 1E6,
    point.getLongitudeE6() / 1E6, 1);
    String address = "";
    if (addresses.size() > 0) {
    for (int i = 0; i < addresses.get(0).getMaxAddressLineIndex(); i++)
    address += addresses.get(0).getAddressLine(i) + "\n";
    }
    Toast.makeText(this, address, Toast.LENGTH_SHORT).show();
    } catch (IOException e) {
    e.printStackTrace();
    }
    List<Overlay> mapOverlays = mapView.getOverlays();
    MyOverlay marker = new MyOverlay(point);

```

```
mapOverlays.add(marker);  
mapView.invalidate();  
}  
}
```

### A.3. Borrar

```
package mapas.android;
import java.util.ArrayList;
import android.graphics.drawable.Drawable;
import com.google.android.maps.ItemizedOverlay;
import com.google.android.maps.OverlayItem;
public class MapOverlay extends ItemizedOverlay {
private ArrayList<OverlayItem> mOverlays = new
ArrayList<OverlayItem>();
public MapOverlay(Drawable _defaultMarker) {
super(boundCenterBottom(_defaultMarker));
}
@Override
protected OverlayItem createItem(int i) {
return mOverlays.get(i);
}
public void addOverlay(OverlayItem overlay) {
mOverlays.add(overlay);
populate();
}
@Override
public int size() {
return mOverlays.size();
}
}
```



## A.4. Editor

```
package mapas.android;
import java.util.ArrayList;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.ListView;
public class Editor extends Activity{
private MyDBAdapter mdb;
private Cursor cursor;
ListView lista;
private MyDBCursorAdapter mdbcursor;
ArrayList<String> items= new ArrayList();
AdaptadorLista al;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.lista);
lista = (ListView) findViewById(R.id.listado);
Intent intent = new Intent();
mdb = new MyDBAdapter(this);
intent=getIntent();
if (intent!=null){
items=intent.getStringArrayListExtra("listadonombres");
System.out.println(items.size());
al = new AdaptadorLista(Editor.this, R.layout.estilo,items);
lista.setAdapter(al);
lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> arg0, View vista,
final int position, long arg3) {
// TODO Auto-generated method stub
//IMPLEMENTAR BORRAR AQUI
System.out.println(items.get(position).toString());
AlertDialog.Builder dialogol = new AlertDialog.Builder(Editor.this);
dialogol.setTitle("Importante");
dialogol.setMessage("¿Realmente desea editar la ubicación:
"+items.get(
position).toString()+"?");
dialogol.setCancelable(false);
dialogol.setPositiveButton("Confirmar", new DialogInterface.
OnClickListener() {
public void onClick(DialogInterface dialogol, int id) {
AlertDialog.Builder alert = new AlertDialog.Builder(Editor.this);
alert.setTitle("Editor");
alert.setMessage("Introduzca el nuevo nombre para la ubicación");
// Set an EditText view to get user input
final EditText input = new EditText(Editor.this);
alert.setView(input);
alert.setPositiveButton("Aceptar", new DialogInterface.
OnClickListener() {
public void onClick(DialogInterface dialog, int whichButton) {
String value = input.getText().toString();
System.out.println("value="+value);
mdb.open();
```



## A.5. Elegir

```
package mapas.android;
import java.util.ArrayList;
import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
public class Elegir extends Activity{
private MyDBAdapter mdb;
ListView lista;
ArrayList<String> items= new ArrayList();
AdaptadorLista al;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.lista);
lista = (ListView) findViewById(R.id.listado);
Intent intent = new Intent();
mdb = new MyDBAdapter(this);
intent=getIntent();
if (intent!=null){
items=intent.getStringArrayListExtra("listadonombres");
System.out.println(items.size());
al = new AdaptadorLista(Elegir.this, R.layout.estilo,items);
lista.setAdapter(al);
lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> arg0, View vista,
final int position, long arg3) {
// TODO Auto-generated method stub
//IMPLEMENTAR BORRAR AQUI
mdb.open();
Cursor c1;
String name, latitude, longitude;
System.out.println(items.get(position).toString());
c1=mdb.getAllEntries();
c1.moveToPosition(position);
name=c1.getString(c1.getColumnIndex("name"));
latitude=c1.getString(c1.getColumnIndex("latitude"));
longitude=c1.getString(c1.getColumnIndex("longitude"));
System.out.println("name = "+name);
System.out.println("latitude = "+latitude);
System.out.println("longitude = "+longitude);
String [] ubicacion = {name, latitude, longitude};
Intent intent = new Intent (Elegir.this, ViewReturn.class);
intent.putExtra("ubicaciones", ubicacion);
startActivity(intent);
mdb.close();
c1.close();
}
});
}
}
```

## A.6. MapOverlay

```
package mapas.android;
import java.util.ArrayList;
import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
public class Elegir extends Activity{
private MyDBAdapter mdb;
ListView lista;
ArrayList<String> items= new ArrayList();
AdaptadorLista al;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.lista);
lista = (ListView) findViewById(R.id.listado);
Intent intent = new Intent();
mdb = new MyDBAdapter(this);
intent=getIntent();
if (intent!=null){
items=intent.getStringArrayListExtra("listadonombres");
System.out.println(items.size());
al = new AdaptadorLista(Elegir.this, R.layout.estilo,items);
lista.setAdapter(al);
lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> arg0, View vista,
final int position, long arg3) {
// TODO Auto-generated method stub
//IMPLEMENTAR BORRAR AQUI
mdb.open();
Cursor c1;
String name, latitude, longitude;
System.out.println(items.get(position).toString());
c1=mdb.getAllEntries();
c1.moveToPosition(position);
name=c1.getString(c1.getColumnIndex("name"));
latitude=c1.getString(c1.getColumnIndex("latitude"));
longitude=c1.getString(c1.getColumnIndex("longitude"));
System.out.println("name = "+name);
System.out.println("latitude = "+latitude);
System.out.println("longitude = "+longitude);
String [] ubicacion = {name, latitude, longitude};
Intent intent = new Intent (Elegir.this, ViewReturn.class);
intent.putExtra("ubicaciones", ubicacion);
startActivity(intent);
mdb.close();
c1.close();
}
});
}
}
```

## A.7. MyDBAdapter

```
package mapas.android;
import android.content.ContentValues;
import android.content.Context;
import android.database.*;
import android.database.sqlite.*;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.util.Log;
//import android.widget.EditText;
//import android.view.Gravity;
//import android.widget.Toast;
public class MyDBAdapter {
private static final String DATABASE_NAME = "myDatabase.db";
private static final String DATABASE_TABLE = "mainTable";
private static final int DATABASE_VERSION = 1;
// The index (key) column name for use in where clauses.
//public static final String KEY_ID="order";
// The name and column index of each column in your database.
private static final String KEY_NAME="name";
private static final String KEY_LATITUDE="latitude";
public static final String KEY_LONGITUDE="longitude";
private static final int NAME_COLUMN = 1;
// TODO: Create public field for each column in your table.
// SQL Statement to create a new database.
private String[] columnas = new
String[]{KEY_NAME,KEY_LATITUDE,KEY_LONGITUDE} ;
private static final String DATABASE_CREATE =
"create table if not exists mainTable (name text primary key," +
" latitude text not null, " +
" longitude text not null );";
// Variable to hold the database instance
private static SQLiteDatabase db;
// Context of the application using the database.
private final Context context;
// Database open/upgrade helper
private myDbHelper dbHelper;
public MyDBAdapter(Context _context) {
context = _context;
dbHelper = new myDbHelper(context, DATABASE_NAME, null,
DATABASE_VERSION);
}
public MyDBAdapter open() throws SQLException {
db = dbHelper.getWritableDatabase();
return this;
}
public void close() {
db.close();
}
public int insertEntry(String name, String latitude, String longitude)
{
// TODO: Create a new ContentValues to represent my row
// and insert it into the database.
ContentValues content= new ContentValues();
content.put(getKeyName(), name);
content.put(KEY_LATITUDE, latitude);
content.put(KEY_LONGITUDE, longitude);
db.insert(DATABASE_TABLE, null, content);
return 0;
}
public boolean removeEntry(String nombre) {
String where="name=' "+nombre+" ' ";
```

```

System.out.println(where);
db.delete(DATABASE_TABLE, where, null);
return true;
}
public Cursor getAllEntries () {
return db.query(DATABASE_TABLE, new String[] { getKeyName(),
KEY_LATITUDE,
KEY_LONGITUDE},
null, null, null, null, null);
}
public Cursor getNombre () {
return db.query(DATABASE_TABLE, new String[] { getKeyName() },
null, null, null, null, null);
}
public static String getKeyName() {
return KEY_NAME;
}
/*
public MyObject getEntry(long _rowIndex) {
// TODO: Return a cursor to a row from the database and
// use the values to populate an instance of MyObject
return objectInstance;
}*/
public int updateEntry(String nombre, String value) {
String where=KEY_NAME+"='"+nombre+"'";
ContentValues contentValues = new ContentValues();
contentValues.put(KEY_NAME, value);
return db.update(DATABASE_TABLE, contentValues, where, null);
}
private static class myDbHelper extends SQLiteOpenHelper {
public myDbHelper(Context context, String name,
CursorFactory factory, int version) {
super(context, name, factory, version+1);
}
// Called when no database exists in disk and the helper class needs
// to create a new one.
@Override
public void onCreate(SQLiteDatabase _db) {
_db.execSQL(DATABASE_CREATE);
}
// Called when there is a database version mismatch meaning that the
version
// of the database on disk needs to be upgraded to the current
version.
@Override
public void onUpgrade(SQLiteDatabase _db, int _oldVersion, int
_newVersion) {
// Log the version upgrade.
Log.w("TaskDBAdapter", "Upgrading from version " +
_oldVersion + " to " +
_newVersion + ", which will destroy all old data");
// Upgrade the existing database to conform to the new version.
Multiple
// previous versions can be handled by comparing _oldVersion and
_newVersion
// values.
// The simplest case is to drop the old table and create a new one.
_db.execSQL("DROP TABLE IF EXISTS " + DATABASE_TABLE);
// Create a new one.
onCreate(_db);
}
}

```

```
}  
public Cursor getCursor() throws SQLException  
{  
    Cursor c = db.query( true, DATABASE_TABLE, columnas, null, null, null,  
        null,  
        null, null);  
    return c;  
}  
}
```

## A.8. MyDBCursorAdapter

```
package mapas.android;
import android.content.Context;
import android.database.Cursor;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.TextView;
public class MyDBCursorAdapter extends CursorAdapter{
private MyDBAdapter mdb = null;
public MyDBCursorAdapter(Context context, Cursor c) {
super(context, c);
mdb = new MyDBAdapter(context);
mdb.open();
}
@Override
public void bindView(View view, Context context, Cursor cursor) {
// TODO Auto-generated method stub
TextView tv = (TextView) view ;
tv.setText(cursor.getString(cursor.getColumnIndex(MyDBAdapter.getKeyName())));
}
@Override
public View newView(Context context, Cursor cursor, ViewGroup parent)
{
final LayoutInflater inflater = LayoutInflater.from(context);
final View view =
inflater.inflate(android.R.layout.simple_dropdown_item_1line,
parent, false);
return view;
}
}
```



## A.9. MyLocation

```
package mapas.android;
import java.util.Timer;
import java.util.TimerTask;
import android.content.Context;
//import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.os.Bundle;
//import android.provider.Settings;
public class MyLocation {
    Timer timer1;
    LocationManager lm;
    LocationResult locationResult;
    LocationProvider provider;
    boolean gps_enabled=false;
    boolean network_enabled=false;
    public boolean getLocation(Context context, LocationResult result)
    {
        //I use LocationResult callback class to pass location value from
        MyLocation to user
        code.
        locationResult=result;
        if(lm==null)
            lm = (LocationManager)
            context.getSystemService(Context.LOCATION_SERVICE);
        provider = lm.getProvider(LocationManager.GPS_PROVIDER);
        //exceptions will be thrown if provider is not permitted.
        try{gps_enabled=lm.isProviderEnabled(LocationManager.GPS_PROVIDER);}ca
        tch(Exception
        ex){}
        try{network_enabled=lm.isProviderEnabled(LocationManager.NETWORK_PROVI
        DER);}catch(
        Exception ex){}
        //don't start listeners if no provider is enabled
        if(!gps_enabled && !network_enabled)
            return false;
        if(gps_enabled)
            lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
            locationListenerGps
            );
        if(network_enabled)
            lm.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0,
            locationListenerNetwork);
        timer1=new Timer();
        timer1.schedule(new GetLastLocation(), 2000);
        return true;
    }
    protected void onStart() {
        // This verification should be done during onStart() because the
        system calls
        // this method when the user returns to the activity, which ensures
        the desired
        // location provider is enabled each time the activity resumes from
        the stopped state.
        LocationManager locationManager;
        locationManager = (LocationManager)
        getSystemService(Context.LOCATION_SERVICE);
```

```

final boolean gpsEnabled =
locationManager.isProviderEnabled(LocationManager.
GPS_PROVIDER);
if (!gpsEnabled) {
// Build an alert dialog here that requests that the user enable
// the location services, then when the user clicks the "OK" button,
// call enableLocationSettings()
}
}
private LocationManager getSystemService(String locationService) {
// TODO Auto-generated method stub
return null;
}
/*private void enableLocationSettings() {
Intent settingsIntent = new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
startActivity(settingsIntent);
}*/
LocationListener locationListenerGps = new LocationListener() {
public void onLocationChanged(Location location) {
timer1.cancel();
locationResult.getLocation(location);
lm.removeUpdates(this);
lm.removeUpdates(locationListenerNetwork);
}
public void onProviderDisabled(String provider){}
public void onProviderEnabled(String provider){}
public void onStatusChanged(String provider, int status, Bundle
extras){}
};
LocationListener locationListenerNetwork = new LocationListener(){
public void onLocationChanged(Location location){
timer1.cancel();
locationResult.getLocation(location);
lm.removeUpdates(this);
lm.removeUpdates(locationListenerGps);
}
public void onProviderDisabled(String provider){}
public void onProviderEnabled(String provider){}
public void onStatusChanged(String provider, int status, Bundle
extras){}
};
class GetLastLocation extends TimerTask {
@Override
public void run() {
lm.removeUpdates(locationListenerGps);
lm.removeUpdates(locationListenerNetwork);
Location net_loc=null, gps_loc=null;
if(gps_enabled)
gps_loc=lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
if(network_enabled)
net_loc=lm.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
//if there are both values use the latest one
if(gps_loc!=null && net_loc!=null){
if(gps_loc.getTime().>net_loc.getTime())
locationResult.getLocation(gps_loc);
else
locationResult.getLocation(net_loc);
return;
}
if(gps_loc!=null){

```

```
locationResult.getLocation(gps_loc);
return;
}
if(net_loc!=null){
locationResult.getLocation(net_loc);
return;
}
locationResult.getLocation(null);
}
}
public static abstract class LocationResult{
public abstract void getLocation(Location location);
}
}
```

## A.10. MyLocationListener

```
package mapas.android;
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapController;
import android.os.Bundle;
import android.widget.Toast;
public class MyLocationListener implements LocationListener {
private Context context;
private LocationManager locationManager;
private MapController mapController;
public MyLocationListener(MapController mapController, LocationManager
locationManager,
Context context)
{
this.mapController = mapController;
this.locationManager = locationManager;
this.context = context;
}
public void onLocationChanged(Location loc) {
Toast.makeText(context, "onLocationChanged",
Toast.LENGTH_SHORT).show();
if (loc!=null) {
Toast.makeText(context,
"Location changed : Lat: " + loc.getLatitude() + " Lng: " + loc.
getLongitude(), Toast.LENGTH_SHORT).show();
GeoPoint p = new GeoPoint(
(int) (loc.getLatitude() * 1E6),
(int) (loc.getLongitude() * 1E6));
mapController.animateTo(p);
mapController.setZoom(12);
}
else
{
Toast.makeText(context, "location changed to null value",
Toast.LENGTH_SHORT).
show();
}
// por el emulador??
locationManager.removeUpdates(this);
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
0, 0,
new MyLocationListener(mapController, locationManager, context));
}
public void onProviderDisabled(String arg0) {
Toast.makeText(context, "Provider Disabled",
Toast.LENGTH_SHORT).show();
}
public void onProviderEnabled(String arg0) {
Toast.makeText(context, "Provider Enabled",
Toast.LENGTH_SHORT).show();
}
public void onStatusChanged(String provider, int status, Bundle arg2)
{
Toast.makeText(context, "Status on " + provider + " is " +
getStatusMessage(status
),
Toast.LENGTH_SHORT).show();
}
```

```
}  
private String getStatusMessage(int status) {  
    // TODO Auto-generated method stub  
    if (status == 1)  
        return "contecting";  
    else if (status == 2)  
        return "conected";  
    return "unknown";  
}  
}
```

## A.11. MyOverlay

```
package mapas.android;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Point;
import android.graphics.RectF;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.Projection;
public class MyOverLay extends Overlay
{
private GeoPoint gp1;
private GeoPoint gp2;
private int mRadius=6;
private int mode=0;
private int defaultColor;
private String text="";
private Bitmap img = null;
public MyOverLay(GeoPoint gp1,GeoPoint gp2,int mode) // GeoPoint is a
int. (6E)
{
this.gp1 = gp1;
this.gp2 = gp2;
this.mode = mode;
defaultColor = 999; // no defaultColor
}
public MyOverLay(GeoPoint gp1,GeoPoint gp2,int mode, int defaultColor)
{
this.gp1 = gp1;
this.gp2 = gp2;
this.mode = mode;
this.defaultColor = defaultColor;
}
public void setText(String t)
{
this.text = t;
}
public void setBitmap(Bitmap bitmap)
{
this.img = bitmap;
}
public int getMode()
{
return mode;
}
@Override
public boolean draw
(Canvas canvas, MapView mapView, boolean shadow, long when)
{
Projection projection = mapView.getProjection();
if (shadow == false)
{
Paint paint = new Paint();
paint.setAntiAlias(true);
Point point = new Point();
projection.toPixels(gp1, point);
// mode=1;Gstart
if(mode==1)
```

```

{
if(defaultColor==999)
paint.setColor(Color.BLUE);
else
paint.setColor(defaultColor);
RectF oval=new RectF(point.x - mRadius, point.y - mRadius,
point.x + mRadius, point.y + mRadius);
// start point
canvas.drawOval(oval, paint);
}
// mode=2;Gpath
else if(mode==2)
{
if(defaultColor==999)
paint.setColor(Color.RED);
else
paint.setColor(defaultColor);
Point point2 = new Point();
projection.toPixels(gp2, point2);
paint.setStrokeWidth(5);
paint.setAlpha(120);
canvas.drawLine(point.x, point.y, point2.x,point2.y, paint);
}
/* mode=3;Gend */
else if(mode==3)
{
/* the last path */
if(defaultColor==999)
paint.setColor(Color.GREEN);
else
paint.setColor(defaultColor);
Point point2 = new Point();
projection.toPixels(gp2, point2);
paint.setStrokeWidth(5);
paint.setAlpha(120);
canvas.drawLine(point.x, point.y, point2.x,point2.y, paint);
RectF oval=new RectF(point2.x - mRadius,point2.y - mRadius,
point2.x + mRadius,point2.y + mRadius);
/* end point */
paint.setAlpha(255);
canvas.drawOval(oval, paint);
}
/* mode=4;Gcar */
else if(mode==4)
{
if(defaultColor==999)
paint.setColor(Color.GREEN);
else
paint.setColor(defaultColor);
Point point2 = new Point();
projection.toPixels(gp2, point2);
paint.setTextSize(20);
paint.setAntiAlias(true);
canvas.drawBitmap(img, point2.x, point2.y,paint);
canvas.drawText(this.text, point2.x, point2.y, paint);
// Log.d(TAG, "Draw the text="+this.text+ " at point="+point2.x + ","
+ point2.y);
}
else if(mode==5)
{
if(defaultColor==999)

```

```
paint.setColor(Color.GREEN);
else
paint.setColor(defaultColor);
Point point2 = new Point();
projection.toPixels(gp2, point2);
paint.setTextSize(20);
paint.setAntiAlias(true);
canvas.drawBitmap(img, point2.x, point2.y, paint);
// Log.d(TAG, "Draw the text="+this.text+ " at point="+point2.x + ", "
+ point2.y);
}
}
return super.draw(canvas, mapView, shadow, when);
}
}
```



## A.12. Opciones

```
package mapas.android;
import java.util.ArrayList;
import com.google.android.maps.MapActivity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Typeface;
import android.location.Location;
import android.location.LocationListener;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
public class Opciones extends MapActivity implements LocationListener{
    Button borrar;
    Button borrar_todo;
    Button editar;
    private MyDBAdapter mdb;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.opciones);
        Typeface font = Typeface.createFromAsset(getAssets(),
            "DroidSans.ttf");
        borrar_todo = (Button) findViewById(R.id.borrar_todo);
        borrar = (Button) findViewById(R.id.borrar);
        editar = (Button) findViewById(R.id.editar);
        borrar_todo.setTypeface(font);
        borrar.setTypeface(font);
        editar.setTypeface(font);
        borrar_todo.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mdb = new MyDBAdapter(Opciones.this);
                mdb.open();
                final Cursor listado = mdb.getNombre();
                if (listado.moveToFirst()){
                    AlertDialog.Builder dialogol = new AlertDialog.Builder(Opciones.this
                    );
                    dialogol.setTitle("Importante");
                    dialogol.setMessage("¿Realmente desea borrar todas las ubicaciones?
                    Si le da a Aceptar, perderá toda la información guardada");
                    dialogol.setCancelable(false);
                    dialogol.setPositiveButton("Confirmar", new DialogInterface.
                    OnClickListener() {
                        public void onClick(DialogInterface dialogol, int id) {
                            do{
                                mdb.removeEntry(listado.getString(listado.
                                getColumnIndex("name")));
                            }while (listado.moveToNext());
                                mdb.close();
                                String mensaje = "Se han borrado todas las ubicaciones";
                                Toast.makeText(Opciones.this, mensaje, Toast.LENGTH_SHORT
                                ).show();
                            }
                        });
                    dialogol.setNegativeButton("Cancelar", new DialogInterface.
                    OnClickListener() {
                        public void onClick(DialogInterface dialogol, int id) {
```

```

}
});
dialogo1.show();
}else{
System.out.println("No hay ninguna ubicación guardada");
AlertDialog.Builder builder = new AlertDialog.Builder(Opciones.
this);
builder.setMessage("No hay ninguna ubicación guardada")
.setTitle("Error")
.setCancelable(false)
.setNeutralButton("Aceptar",
new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int
id) {
dialog.cancel();
}
});
AlertDialog alert = builder.create();
alert.show();
}
}
});
borrar.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
mdb = new MyDBAdapter(Opciones.this);
mdb.open();
ArrayList lista= new ArrayList<String>();
Cursor listado = mdb.getNombre();
if (listado.moveToFirst()){
listado.moveToFirst();
String c1 ;
do{
c1 = listado.getString(listado.getColumnIndex("name"));
lista.add(c1);
System.out.println(c1);
}while (listado.moveToNext());
Intent intent = new Intent (Opciones.this, Borrar.class);
intent.putExtra("listadonombres", lista);
startActivity(intent);
mdb.close();
}else{
System.out.println("No hay ninguna ubicación guardada");
AlertDialog.Builder builder = new AlertDialog.Builder(Opciones.this);
builder.setMessage("No hay ninguna ubicación guardada")
.setTitle("Error")
.setCancelable(false)
.setNeutralButton("Aceptar",
new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int id) {
dialog.cancel();
}
});
AlertDialog alert = builder.create();
alert.show();
}
}
});
editar.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {

```

```

mdb = new MyDBAdapter(Opciones.this);
mdb.open();
ArrayList lista= new ArrayList<String>();
Cursor listado = mdb.getNombre();
if (listado.moveToFirst()){
listado.moveToFirst();
String c1 ;
do{
c1 = listado.getString(listado.getColumnIndex("name"));
lista.add(c1);
System.out.println(c1);
}while (listado.moveToNext());
Intent intent = new Intent (Opciones.this, Editar.class);
intent.putExtra("listadonombres", lista);
startActivity(intent);
mdb.close();
/*Intent intent = new Intent (Opciones.this, Borrar.class);
startActivity(intent);*/
}else{
System.out.println("No hay ninguna ubicación guardada");
AlertDialog.Builder builder = new AlertDialog.Builder(Opciones.this);
builder.setMessage("No hay ninguna ubicación guardada")
.setTitle("Error")
.setCancelable(false)
.setNeutralButton("Aceptar",
new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int id) {
dialog.cancel();
}
});
AlertDialog alert = builder.create();
alert.show();
}
});
}
@Override
public void onLocationChanged(Location location) {
// TODO Auto-generated method stub
}
@Override
public void onProviderDisabled(String provider) {
// TODO Auto-generated method stub
}
@Override
public void onProviderEnabled(String provider) {
// TODO Auto-generated method stub
}
@Override
public void onStatusChanged(String provider, int status, Bundle
extras) {
// TODO Auto-generated method stub
}
@Override
protected boolean isRouteDisplayed() {
// TODO Auto-generated method stub
return false;
}
}
}

```

## A.13. ViewMaps

```
package mapas.android;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.List;
import java.util.Locale;
import mapas.android.MyDBAdapter;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import android.graphics.Typeface;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
public class ViewMaps extends MapActivity implements LocationListener{
private MapView mapa = null;
Button guardar;
EditText nombre;
ProgressDialog dialog;
double latitud;
double longitud;
int control;
@Override
protected void onCreate(Bundle savedInstanceState) {
// TODO Auto-generated method stub
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
Typeface font = Typeface.createFromAsset(getAssets(),
"DroidSans.ttf");
guardar=(Button)findViewById(R.id.button3);
nombre=(EditText)findViewById(R.id.nombre);
guardar.setTypeface(font);
nombre.setTypeface(font);
final MyDBAdapter mdb = new MyDBAdapter(ViewMaps.this);
dialog = new ProgressDialog(ViewMaps.this);
dialog.setTitle("Cargando");
dialog.setMessage("Obteniendo ubicación GPS");
dialog.show();
guardar.setOnClickListener(new View.OnClickListener() {
@Override
```

```

public void onClick(View v){
String sacoNombre=nombre.getText().toString();
String comprueboNombre;
if(sacoNombre.equals("")){
Calendar c = Calendar.getInstance();
SimpleDateFormat df3 = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss a");
String formattedDate3 = df3.format(c.getTime());
System.out.println(formattedDate3);
mdb.open();
mdb.insertEntry(formattedDate3, Double.toString(latitud),
Double.toString
(longitud));
mdb.close();
String mensaje = "La ubicación se ha guardado correctamente";
Toast pantalla = Toast.makeText(ViewMaps.this, mensaje, Toast.
LENGTH_SHORT);
pantalla.setGravity(Gravity.CENTER, 0, 0);
pantalla.show();
}else{
mdb.open();
Cursor c1;
c1=mdb.getNombre();
c1.moveToFirst();
int control = 0;
if(c1.moveToFirst()){
do{
comprueboNombre=c1.getString(c1.getColumnIndex(mdb.getKeyName().
toString()));
System.out.println("c1="+c1.getString(c1.getColumnIndex(mdb.
getKeyName().toString())));
if(comprueboNombre.equals(sacoNombre)){
control=1;
AlertDialog.Builder builder = new AlertDialog.Builder(ViewMaps.
this);
builder.setMessage("Ese nombre ya está asignado a otra
ubicación. Por favor, cambielo.")
.setTitle("Error")
.setCancelable(false)
.setNeutralButton("Aceptar",
new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog,
int id) {
dialog.cancel();
}
});
AlertDialog alert = builder.create();
alert.show();
}
}while(c1.moveToNext());
if(control==0){
control=0;
mdb.insertEntry(nombre.getText().toString(), Double.toString(
latitud), Double.toString(longitud));
String mensaje = "La ubicación se ha guardado correctamente";
Toast pantalla = Toast.makeText(ViewMaps.this, mensaje, Toast.
LENGTH_SHORT);
pantalla.setGravity(Gravity.CENTER, 0, 0);
pantalla.show();
}
mdb.close();
}else{

```



```

}
}
Geocoder geoCoder = new Geocoder(this, Locale.getDefault());
try {
List<Address> addresses =
geoCoder.getFromLocation(point.getLatitudeE6() /
1E6, point.getLongitudeE6() / 1E6, 1);
String address = "";
if (addresses.size() > 0) {
for (int i = 0; i < addresses.get(0).getMaxAddressLineIndex(); i++)
address += addresses.get(0).getAddressLine(i) + "\n";
}
Toast.makeText(this, address, Toast.LENGTH_SHORT).show();
} catch (IOException e) {
e.printStackTrace();
}
List<Overlay> mapOverlays = mapView.getOverlays();
MyOverlay marker = new MyOverlay(point);
mapOverlays.add(marker);
mapView.invalidate();
}
/**
 *
 *
 * **/
class MyOverlay extends Overlay {
GeoPoint point;
//El constructor recibe el punto donde se dibujará el marker
public MyOverlay(GeoPoint point) {
super();
this.point = point;
}
@Override
public boolean draw(Canvas canvas, MapView mapView, boolean shadow,
long when) {
super.draw(canvas, mapView, shadow);
//se traduce el punto geolocalizado a un punto en la pantalla */
Point scrnPoint = new Point();
mapView.getProjection().toPixels(this.point, scrnPoint);
//se construye un bitmap a partir de la imagen
Bitmap marker = BitmapFactory.decodeResource(getResources(),
R.drawable.icon);
// se dibuja la imagen del marker
canvas.drawBitmap(marker, scrnPoint.x - marker.getWidth() / 2,
scrnPoint.y -
marker.getHeight() / 2, null);
return true;
}
}
@Override
protected boolean isRouteDisplayed() {
// TODO Auto-generated method stub
return false;
}
}
}

```

## A.14. ViewReturn

```
package mapas.android;
//import mapas.android.MyDBAdapter;
import java.util.List;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
//import com.google.android.maps.Projection;
import android.net.Uri;
import android.os.Bundle;
//import android.view.View;
//import android.widget.Button;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
public class ViewReturn extends MapActivity implements
LocationListener{
private MapView mapa = null;
ProgressDialog dialog;
double latitud, latitud2;
double longitud, longitud2;
int control;
String [] ubicaciones;
MapController mc;
@Override
protected boolean isRouteDisplayed() {
// TODO Auto-generated method stub
return false;
}
@Override
protected void onCreate(Bundle savedInstanceState) {
// TODO Auto-generated method stub
super.onCreate(savedInstanceState);
setContentView(R.layout.ruta);
dialog = new ProgressDialog(ViewReturn.this);
dialog.setTitle("Cargando");
dialog.setMessage("Obteniendo ubicación GPS");
dialog.show();
Intent intent = new Intent();
intent=getIntent();
if (intent!=null){
ubicaciones=intent.getStringArrayExtra("ubicaciones");
}
//Obtenemos una referencia al control MapView
mapa = (MapView)findViewById(R.id.mapa2);
//Mostramos los controles de zoom sobre el mapa
mapa.setBuiltInZoomControls(true);
//Mostrar el mapa en modo satélite y no callejero
mapa.setSatellite(false);
```



```

LocationManager locationManager =
(LocationManager) getSystemService(Context.
LOCATION_SERVICE);
//updateLocation(locationManager.getLastKnownLocation(LocationManager.
GPS_PROVIDER));
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
6000, 50, this);
}
@Override
public void onLocationChanged(Location location) {
updateLocation(location);
}
//Para tener el GPS funcionando antes de que la aplicación inicie
@Override
public void onProviderDisabled(String provider) {
Intent intent = new Intent( android.provider.Settings.
ACTION_LOCATION_SOURCE_SETTINGS);
startActivity(intent);
}
@Override
public void onProviderEnabled(String provider) {
// TODO Auto-generated method stub
}
@Override
public void onStatusChanged(String provider, int status, Bundle
extras) {
// TODO Auto-generated method stub
}
//Actualizaciones de ubicación
protected void updateLocation(Location location){
MapView mapView = (MapView) findViewById(R.id.mapa2);
MapController mapController = mapView.getController();
//Localización!!
GeoPoint point = new GeoPoint((int) (location.getLatitude() * 1E6),
(int) (
location.getLongitude() * 1E6));
//dialog = new ProgressDialog(ViewMaps.this);
mapController.animateTo(point);
mapController.setZoom(15);
//PUNTO ACTUAL
latitud=point.getLatitudeE6()/1E6;
longitud=point.getLongitudeE6()/1E6;
//PUNTO RETORNO
latitud2=Double.parseDouble(ubicaciones[1])/1E6;
longitud2=Double.parseDouble(ubicaciones[2])/1E6;
GeoPoint point2 = new GeoPoint ((int)(latitud2*1E6),
(int)(longitud2*1E6));
if (point != null){
if (dialog != null){
dialog.dismiss();
}
}
}
List<Overlay> mapOverlays = mapView.getOverlays();
MyOverlay marker = new MyOverlay(point, point2);
mapOverlays.add(marker);
mapView.invalidate();
//Salto a google maps con las ubicaciones, para dibujar la ruta
Intent intent = new Intent(Intent.ACTION_VIEW,Uri.parse(
"http://maps.google.com/maps?" + "saddr="+
latitud + "," + longitud + "&daddr=" + latitud2 + "," + longitud2));
intent.setClassName("com.google.android.apps.maps",

```

```

"com.google.android.maps.MapActivity");
startActivity(intent);
}
class MyOverlay extends Overlay {
GeoPoint point, point2;
//El constructor recibe el punto donde se dibujará el marker
public MyOverlay(GeoPoint point, GeoPoint point2) {
super();
this.point = point;
this.point2 = point2;
}
@Override
public boolean draw(Canvas canvas, MapView mapView, boolean shadow,
long when) {
super.draw(canvas, mapView, shadow);
//se traduce el punto geolocalizado a un punto en la pantalla
Point scrnPoint = new Point();
mapView.getProjection().toPixels(this.point, scrnPoint);
Point scrnPoint2 = new Point();
mapView.getProjection().toPixels(this.point2, scrnPoint2);
//se construye un bitmap a partir de la imagen
Bitmap marker = BitmapFactory.decodeResource(getResources(),
R.drawable.icon);
Bitmap marker2 = BitmapFactory.decodeResource(getResources(),
R.drawable.
icon2);
// se dibuja la imagen del marker
canvas.drawBitmap(marker, scrnPoint.x - marker.getWidth() / 2,
scrnPoint.y -
marker.getHeight() / 2, null);
canvas.drawBitmap(marker2, scrnPoint2.x - marker2.getWidth() / 2,
scrnPoint2.
y - marker2.getHeight() / 2, null);
return true;
}
}
class ThreadGetPoints extends Thread{
GeoPoint src;
GeoPoint dest;
int color;
MapView mMapView01;
public ThreadGetPoints( GeoPoint src,GeoPoint dest, int color,MapView
mMapView01){
this.src=src;
this.dest=dest;
this.color=color;
this.mMapView01=mMapView01;
}
@Override
public void run() {
}
}
}
}

```

## A.14. AndroidManifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="mapas.android"
android:versionCode="1"
android:versionName="1.0">
<uses-sdk android:minSdkVersion="10" />
<uses-permission android:name="android.permission.INTERNET" />
<!-- REDES -->
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"
></uses-permission>
<!-- GPS -->
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"></uses-
permission>
<application
android:icon="@drawable/icono"
android:label="@string/app_name"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen">
<uses-library android:name="com.google.android.maps" />
<activity android:name=".AndroidMapas"
android:label="@string/app_name"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
android:screenOrientation="portrait">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".ViewMaps"
android:label="@string/app_name"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
android:screenOrientation="portrait">
<intent-filter>
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".MyDBAdapter"
android:label="@string/app_name"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
android:screenOrientation="portrait">
<intent-filter>
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".ViewReturn"
android:label="@string/app_name"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
android:screenOrientation="portrait">
<intent-filter>
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".Opciones"
android:label="@string/app_name"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
android:screenOrientation="portrait">
<intent-filter>
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

```
</activity>
<activity android:name=".Borrar"
android:label="@string/app_name"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
android:screenOrientation="portrait">
<intent-filter>
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".Elegir"
android:label="@string/app_name"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
android:screenOrientation="portrait">
<intent-filter>
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".Editar"
android:label="@string/app_name"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
android:screenOrientation="portrait">
<intent-filter>
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".MyDBCursorAdapter"
android:label="@string/app_name"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen">
<intent-filter>
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

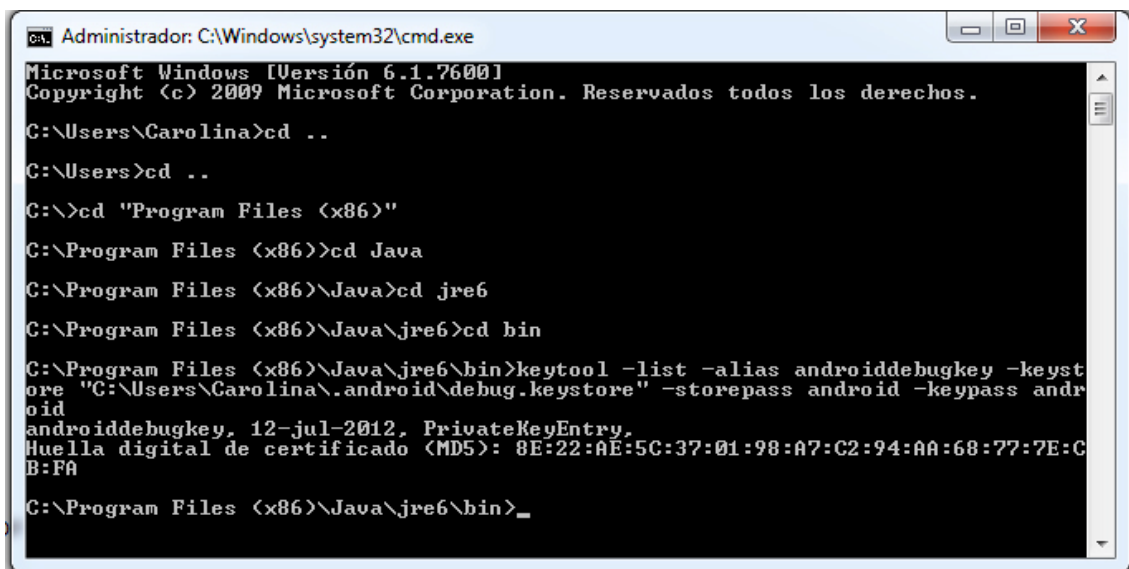
## Anexo B. Obtención de la API Key de Google

Todas las aplicaciones del API de Google Maps deben cargar el API de Google Maps mediante una clave de API. Utilizar una clave de API permite controlar cómo utiliza la aplicación el API de Google Maps y asegurar de que Google puede ponerse en contacto con el desarrollador de la aplicación si fuese necesario.

Para la obtención de los mapas, es necesario cargar esta API dentro de los XML, en la parte donde usamos el MapView. Para obtener la API Key de google, debemos seguir los siguientes pasos:

Lo primero que tenemos que hacer es escribir la siguiente sentencia en una consola de Windows, situándonos en la ruta donde tenemos instalado el jdk de Java:

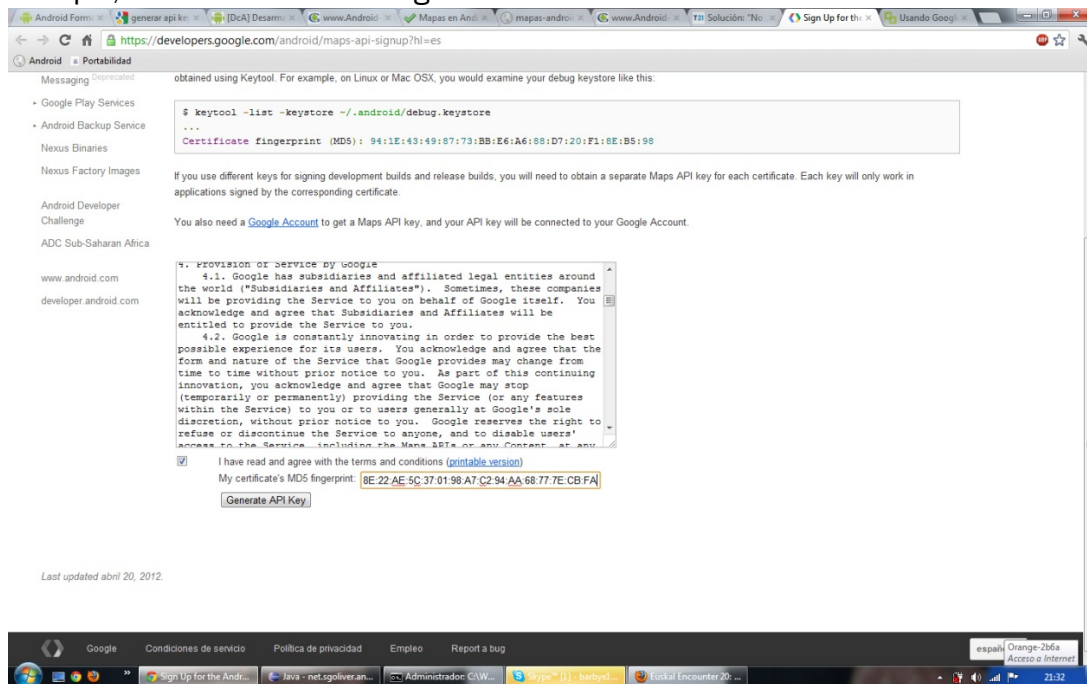
```
keytool -list -alias androiddebugkey -keystore  
"C:\Users\Carolina\.android\debug.keystore" -storepass android -keypass  
android
```



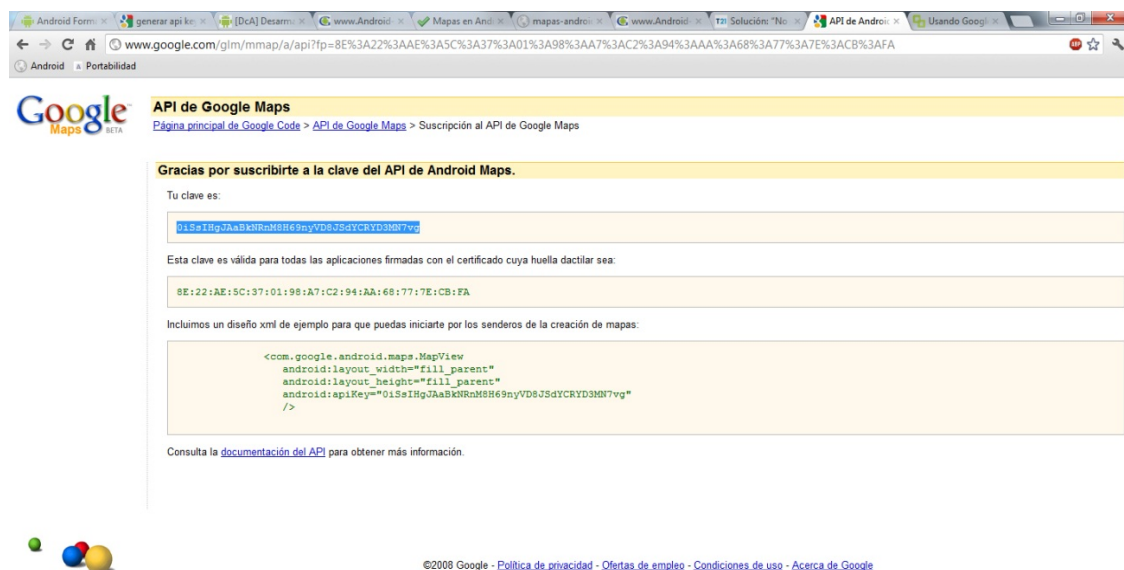
```
ca. Administrador: C:\Windows\system32\cmd.exe  
Microsoft Windows [Versión 6.1.7600]  
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.  
C:\Users\Carolina>cd ..  
C:\Users>cd ..  
C:\>cd "Program Files (x86)"  
C:\Program Files (x86)>cd Java  
C:\Program Files (x86)\Java>cd jre6  
C:\Program Files (x86)\Java\jre6>cd bin  
C:\Program Files (x86)\Java\jre6\bin>keytool -list -alias androiddebugkey -keystore "C:\Users\Carolina\.android\debug.keystore" -storepass android -keypass android  
androiddebugkey, 12-jul-2012, PrivateKeyEntry,  
Huella digital de certificado (MD5): 8E:22:AÉ:5C:37:01:98:A7:C2:94:AA:68:77:7E:C  
B:FA  
C:\Program Files (x86)\Java\jre6\bin>_
```

Esto nos devuelve una huella digital de certificado (MD5) necesaria para la obtención de la API.

Después, debemos ir a la web de desarrolladores de android y, en el apartado de las apis, introducir la huella digital obtenida anteriormente.



Aceptamos los términos y condiciones, y pulsamos el botón para generar nuestra API.



Esa clave, como hemos dicho, debemos copiarla en el MapView del XML donde visualizamos los mapas. Cada cierto tiempo, esta clave caduca, y debemos volverla a generar.

## **Anexo C. Bibliografía y Referencias**

Referencias web:

<http://developer.android.com/index.html>

<http://stackoverflow.com/>

<http://www.sgoliver.net/>

Bibliografía:

Professional Android™ 2 Application Development; Reto Meier