Universitat Oberta de Catalunya

# Cuaderno de Viaje

Memoria


GFDL

**Antonio Fernández Moreno**
TFC – Desarrollo de Aplicaciones Móviles
Consultor: Víctor Carceler Hontoria
30/12/2012

# Contenido

Resumen ..... 5
Presentación ..... 6
Memoria ..... 7
Introducción ..... 7
Definición del proyecto ..... 7
Objetivos ..... 8
Planificacion ..... 8
Análisis ..... 9
Metodología ..... 9
Requisitos Iniciales ..... 11
Diagrama de Casos de Uso ..... 13
Diseño ..... 14
Diagrama de Clases ..... 14
Interfaz de Usuario ..... 15
Modelo de Datos ..... 20
Implementación ..... 20
Tareas Previas ..... 20
Interfaz de Usuario ..... 21
Base de Datos ..... 25
Activities ..... 26
AndroidManifest.xml ..... 39
Plan de Pruebas ..... 41
Planes de Futuro ..... 42
Conclusiones ..... 43
Bibliografía ..... 44
Anexos ..... 45
ANEXO I. Plan de Trabajo ..... 45
ANEXO II. General Public License v3 ..... 45
ANEXO III. GNU Free Documentation License ..... 45

# Índice de Figuras

Figura 1. Ciclo de Vida de AUP ..... 10
Figura 2. Diagrama de Casos de Uso ..... 13
Figura 3. Diagrama de Clases ..... 14
Figura 4. Pantalla Inicial ..... 15
Figura 5. Listado de Viajes ..... 16
Figura 6. Listado de Etapas ..... 17
Figura 7. Listado de Elementos ..... 18
Figura 8. Formulario de Datos ..... 19
Figura 9. Mapa ..... 19
Figura 10. Modelo de Datos ..... 20

A Marta, mi esposa, sin cuyo apoyo y
sacrificio jamás habría llegado hasta aquí

# Resumen

En los últimos años, la tecnología móvil es uno de los campos que han experimentado mayor desarrollo y rápida implantación dentro de nuestra vida diaria. Estos dispositivos han ido sustituyendo, poco a poco, gran cantidad de accesorios que han quedado centralizados en el propio teléfono móvil. La suma de estas circunstancias, unido a la cada vez mayor facilidad para la implementación de aplicaciones que se van integrando en nuestra cotidianidad, ha convertido los *smart phone* en unos instrumentos tremendamente versátiles.

Profundizando en mis inquietudes personales, he detectado una necesidad relacionada con una de las actividades más practicadas por la población en momentos de ocio: **el viaje**. Muchas iniciativas exitosas se han relacionado con el hecho de viajar, desplazarse o buscar localizaciones. Me preguntaba cómo podía aplicar la tecnología al hecho de viajar, e inspirándome en los libros de viaje clásicos, como los cuadernos de viaje de Goethe, y en las anotaciones y reseñas que realizo en mis viajes personales, la idea de realizar un cuaderno de viaje virtual me pareció interesante.

El objetivo de la aplicación *Cuaderno de Viaje* no es otro que ayudar al viajero, aprovechar las ventajas que ofrece la tecnología, como la de aligerar la carga que necesita llevar, posibilitar la contextualización las fotografías realizadas durante el viaje y hacer más sencilla y rápida cualquier anotación que pudiera querer realizar en su cuaderno. La misión no es otra que optimizar el tiempo que dedica al viaje sin tener que prescindir de la posibilidad de almacenar los recuerdos vividos durante el mismo.

Por lo demás, espero que encuentren útil e interesante esta aplicación.

Antonio Fernández Moreno

# Presentación

Este documento es una presentación de la memoria relativa al *Trabajo de Fin de Carrera* (en adelante TFC), que se enmarca dentro del área de *Desarrollo de Aplicaciones Móviles*, y que recoge la totalidad de tareas realizadas para el desarrollo de este proyecto.

En primer lugar, se realiza una definición del proyecto de manera que pueda ofrecer una visión global del alcance del TFC. Se indicarán, además, cuáles son los objetivos que se han perseguido durante la realización del trabajo.

Adicionalmente, se expone un análisis detallado de la aplicación, conformado por los diagramas de casos de uso y de clases, que han servido de base para la elaboración del diseño y construcción de este software. En este punto, se incluye un primer diseño de la interfaz de usuario sobre la que comenzará la codificación, quedando para una fase posterior la implementación de mejoras visuales sobre ella.

# Memoria

## Introducción

### Definición del proyecto

El TFC va a consistir en el desarrollo de una aplicación móvil para el sistema operativo Android en el que, además, se hará uso del API de Google Maps. Esta aplicación pretende ser un *Cuaderno de Viaje* para dispositivos móviles, de manera que el usuario pueda almacenar cualquier recuerdo que considere oportuno durante su viaje, al igual que haría sobre una típica libreta, pero aprovechando las facilidades que ofrece la tecnología.

Las características de la aplicación son las siguientes:

Viajes y Etapas.

Cada viaje que el usuario realice podrá estar dividido en etapas, dentro de las cuales se almacenarán los datos que el viajero desee. De esta forma se dispondrá de todo el contenido perfectamente estructurado y fácilmente accesible.

Rutas

Una de las principales ventajas que ofrece la aplicación es la de almacenar la ruta que realiza el viajero, sin que este se tenga que preocupar de realizar anotaciones a cada momento. Gracias a la capacidad de geolocalización de los dispositivos móviles, se puede registrar el recorrido de manera desatendida, pudiéndolo consultar posteriormente sobre los mapas de Google.

Puntos de Interés

Durante cualquier viaje, siempre existen lugares que llaman la atención del viajero (un restaurante especial, una deliciosa heladería, un monumento, etc.). La aplicación ofrece la oportunidad de almacenar la posición geográfica de este Punto de Interés (en adelante POI) en el sistema, y consultarlo más adelante, así como calcular la ruta para llegar hasta él.

Fotografías

Como en cualquier cuaderno de viaje que se precie, se debe poder almacenar todo recuerdo que interese al viajero, desde una fotografía con amigos, la tarjeta de visita de la persona que ha conocido en el tren, la factura del restaurante, etc. Todos estos recuerdos pueden quedar digitalizados mediante la capacidad de fotografiar de que disponen los actuales dispositivos móviles. Por supuesto, estos datos también serán geoposicionados.

Notas

Todo cuaderno de viaje ha de disponer de las propias anotaciones realizadas por el viajero. Estas, además, podrán incluirse en todos los elementos de la aplicación (etapas, rutas, POIs y fotografías).

## Objetivos

Los objetivos que se buscan con la elaboración del TFC son los siguientes:

- Introducir al desarrollo de aplicaciones móviles para Android.
- Profundizar el conocimiento del proceso de desarrollo y gestión de proyectos.
- Dar respuesta a una necesidad a través de un proyecto informático.
- Mejorar habilidades de planificación, estimación, desarrollo y presentación de proyectos.
- Impulsar la creatividad personal.

## Planificacion

Ver anexo "Plan de Trabajo"

## Análisis

A continuación se detallan los pasos llevados a cabo para realizar el análisis del proyecto, donde destacan las acciones de seleccionar la metodología de desarrollo más adecuada para llevar a cabo la aplicación, establecer los requisitos necesarios y, por supuesto, realizar la descomposición en partes más simples del problema que planteamos resolver, convirtiendo una idea abstracta en factores tangibles simplificados que van a facilitar el diseño de la aplicación.

### Metodología

A la hora de escoger una metodología de desarrollo conveniente para el proyecto se deben considerar las limitaciones, especialmente temporales, inherentes a la asignatura. Puesto que se dispone de un tiempo de desarrollo muy acotado y todo el desarrollo está a cargo de una única persona durante todo el ciclo de vida del proyecto, se establece la metodología AUP (Agile Unified Process) como la más adecuada.

La metodología AUP, creada por Scott W. Ambler, está basada en el modelo RUP (Rational Unified Process), pero de una forma simplificada, de manera que se adapta más apropiadamente al propósito. Se trata, por tanto, de un modelo de desarrollo iterativa e incremental.

La filosofía AUP tiene como base los siguientes criterios:

- El equipo sabe lo que hace
- Simplicidad
- Agilidad
- Centrarse en las actividades más importantes
- Independencia de las herramientas
- Producto adaptable

El sistema AUP consta de 7 flujos de trabajo que se detallan a continuación:

- Modelo: Conocer el negocio, requerimientos y aportar soluciones al problema que se plantea.
- Implementación: Realizar la codificación de manera que se consiga el objetivo perseguido.
- Prueba: Evaluar que el desarrollo cumple con la funcionalidad definida y no tiene errores.
- Despliegue: Poner la aplicación a disposición de los usuarios.
- Gestión de Configuración: Asegurar la calidad del software a través del control de los elementos involucrados en el desarrollo.

- Gestión de Proyecto: Coordinar las actividades necesarias para la consecución de los objetivos.
- Entorno: Establecer las herramientas y los procesos en consonancia con el proyecto.

En esta metodología se utilizan las cuatro fases típicas de las que dispone el modelo Rational Unified Process:

- Inicio: Establecer la justificación y el alcance del proyecto.
- Elaboración: Analizar el dominio teniendo en cuenta necesidades y restricciones, así como definir la arquitectura del proyecto y su planificación.
- Construcción: Desarrollo de la arquitectura establecida en la anterior etapa de forma iterativa e incremental.
- Transición: Entrega y adecuación del producto final hasta convertirlo en definitivo.

En la siguiente imagen observamos el ciclo de vida de AUP:

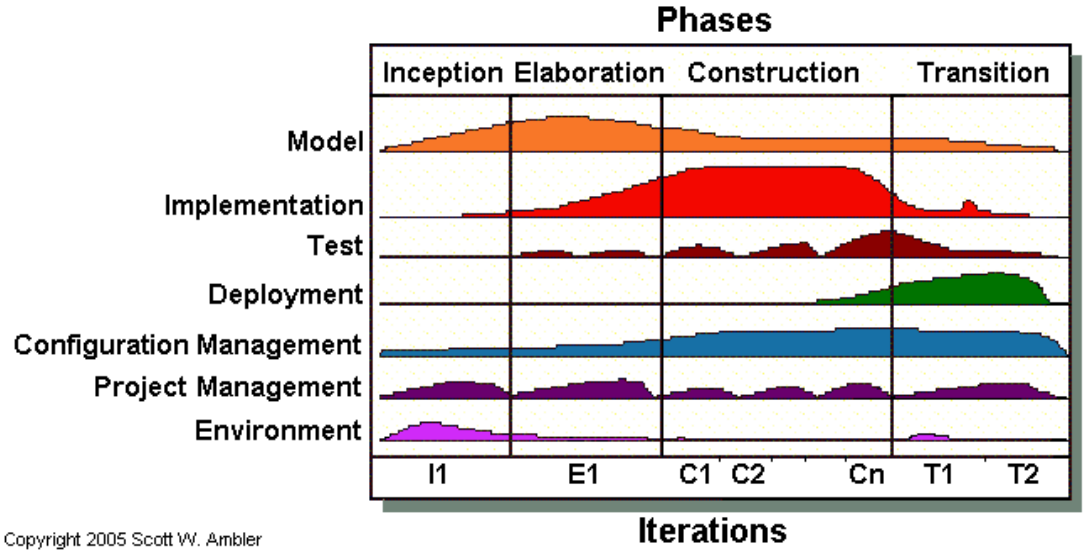


Figura 1. Ciclo de Vida de AUP

Cada una de las anteriores fases incluyen los siguientes componentes:

- Toma de requisitos
- Análisis y Diseño
- Implementación
- Prueba

## Requisitos Iniciales

El primer paso a la hora de realizar el análisis de la aplicación que va a desarrollarse debe ser realizar una reflexión sobre qué es lo que queremos obtener y cómo ha de llevarse a cabo. En este aspecto se deben distinguir dos puntos importantes: los requisitos funcionales del proyecto, es decir, qué debe hacer la aplicación y cómo, y por otro lado los requisitos no funcionales o cuáles son las herramientas que se van a necesitar para conseguirlo.

### *Requisitos Funcionales*

#### Viajes y Etapas

En primer lugar, tiene que establecerse cuál será el núcleo de la aplicación y, como no puede ser de otra manera, este va a ser el concepto de *Viaje*. Por tanto, la aplicación se vertebrará a partir de los viajes que el usuario vaya realizando y sobre estos se irán añadiendo las anotaciones.

Sin embargo, el resultado de añadir elementos directamente sobre el viaje puede resultar un poco caótico, por lo que se hace necesario un nivel de ordenación. Se considera que para una aplicación de este tipo, que trata de virtualizar un cuaderno de viaje físico, existen dos modos principales de clasificación: el formato *Diario*, orientado a la fecha en la que se produce, o el formato *Cuaderno*, que agruparía por capítulos.

En este sentido, la ordenación de elementos de interés durante el viaje que resulta más práctica y elegante es el formato *Cuaderno*, al que se denominará *Etapa*. Este método tiene la ventaja de permitir una clasificación personalizada, a diferencia del otro modelo.

Los elementos accesorios se van a encontrar asociados a la *Etapa*, es decir, para almacenar cualquier información se deberá haber seleccionado inicialmente una. El sistema mantenedrá la última etapa seleccionada como etapa activa, para facilitar el proceso de almacenamiento de información, evitando así tener que realizar la selección cada vez.

#### Rutas

El primer tipo de elemento de interés que podrá almacenar el usuario es la *Ruta* realizada, de forma que simplemente necesite pulsar un botón para comenzar la grabación y volver a presionarlo para finalizarla, asignándole entonces un nombre que la identifique y una descripción si se estima oportuno. Por supuesto, el sistema ha de permitir el almacenamiento de otros elementos mientras guarda la ruta, puesto que de lo contrario la funcionalidad de la aplicación quedaría notablemente mermada.

POIs

El viajero puede en cualquier momento tener la necesidad de almacenar algún comentario sobre el lugar en el que se encuentra en ese momento y para ello sencillamente deberá pulsar un botón que registrará de forma inmediata la posición geográfica del dispositivo, requiriendo al usuario un nombre que identifique el POI y una descripción del mismo.

Foto

Del mismo modo que en los POIs, el viajero puede querer almacenar una fotografía al igual que realizaría un pequeño dibujo en un cuaderno de viaje clásico, o añadiría algún ticket o postal como recuerdo. De este modo, pulsando un botón se activará la cámara del dispositivo, y una vez tomada la instantánea se registrará la posición geográfica y, como en los anteriores casos, se requerirá un nombre y una descripción para la misma.

Nota

Un cuaderno de viaje no se limita a recoger los sitios por los que pasamos durante el recorrido, sino que sirve para almacenar cualquier tipo de pensamiento, detalle o anotación que considere el usuario y que no están sujetos a una posición geográfica concreta. Para este tipo de necesidades simplemente hay que pulsar un botón y comenzar a escribir, así como insertar un identificador para la nota. Este tipo de elementos no incluirán información del GPS.

Modo Consulta

Por supuesto, el cuaderno de viaje no tiene como única finalidad almacenar datos, sino que también es necesario que puedan ser consultados. En este aspecto el usuario indicará que quiere recuperar un viaje, de manera que pueda seleccionarlo de un listado, así como la etapa que le interese de dicho viaje. Una vez seleccionados ambos, el viajero dispondrá de un registro de elementos almacenados, de forma que cada vez que seleccione alguno se mostrará su posición en el mapa. En el caso de las notas, al no estar geoposicionadas, se presentarán directamente como texto.

Se permitirá el movimiento a través del mapa, así como ajustar el zoom del mismo según interese. Si se pulsa sobre un punto de interés en cuestión se mostrará en detalle el mismo. En el caso de las fotografías también se podrá visualizar la instantánea.

### *Requisitos No Funcionales*

Interfaz

La función principal de la aplicación es disponer de un método que permita al viajero realizar anotaciones para que pueda disfrutar completamente del viaje sin tener que

perder el tiempo, y por ello se debe tener esto en cuenta a la hora de diseñar la interfaz de usuario. El principal requisito será disponer de una pantalla inicial desde la que se puedan almacenar directamente los elementos de interés sin tener que navegar por innecesarios menús. Es decir, sacar el dispositivo, pulsar un botón y almacenar la descripción.

Dispositivo

El proyecto va a desarrollarse sobre Android 4.1, puesto que es la última versión disponible en el momento de la realización de la aplicación. De esta forma será necesario un móvil con este sistema operativo y que, además, disponga de cámara fotográfica y GPS integrados para la realización de las instantáneas geolocalizadas.

Herramientas

Para la realización del proyecto se requiere un ordenador sobre el que deberemos instalar el entorno de desarrollo *Eclipse*. Como sistema gestor de bases de datos utilizaremos SQLite, puesto que es un medio ligero y perfectamente integrado con la plataforma Android.

## Diagrama de Casos de Uso

A partir de los requisitos del apartado anterior, se identificarán los casos de uso que compondrán la aplicación. Se encuentran detallados en el siguiente diagrama los casos de uso, así como sus interrelaciones, tanto internamente dentro de la aplicación, como con elementos externos a ella:

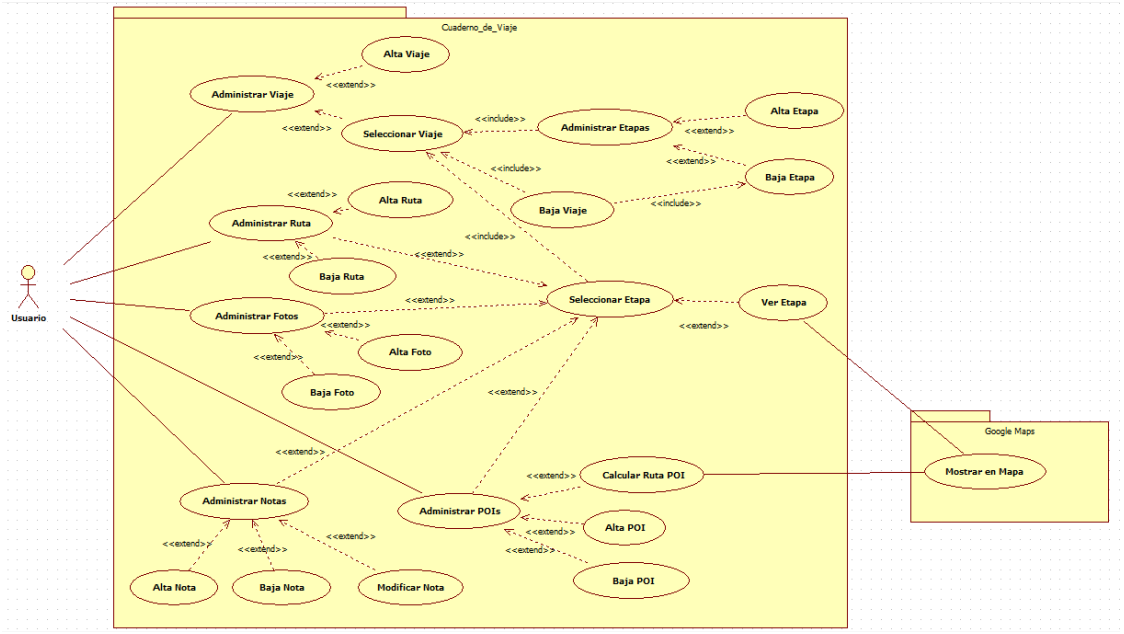


Figura 2. Diagrama de Casos de Uso

## Diseño

Tras la fase de análisis de la aplicación se puede iniciar el diseño de la misma. En este aspecto, se va a comenzar realizando la identificación de las clases de las que constará el proyecto, que quedan plasmadas en el consiguiente diagrama que muestra sus interrelaciones.

Posteriormente, se presentará otro aspecto no menos importante de la fase de diseño: la interfaz de usuario. Se trata de un elemento particularmente importante puesto que supone el contacto directo entre el usuario y la aplicación.

Finalmente, se detallará el diseño de modelo de datos que permitirá organizar la información necesaria para el correcto funcionamiento de la aplicación de la forma más eficiente posible.

### Diagrama de Clases

En el siguiente diagrama se identifican las clases de las que constará la aplicación, así como sus interrelaciones.

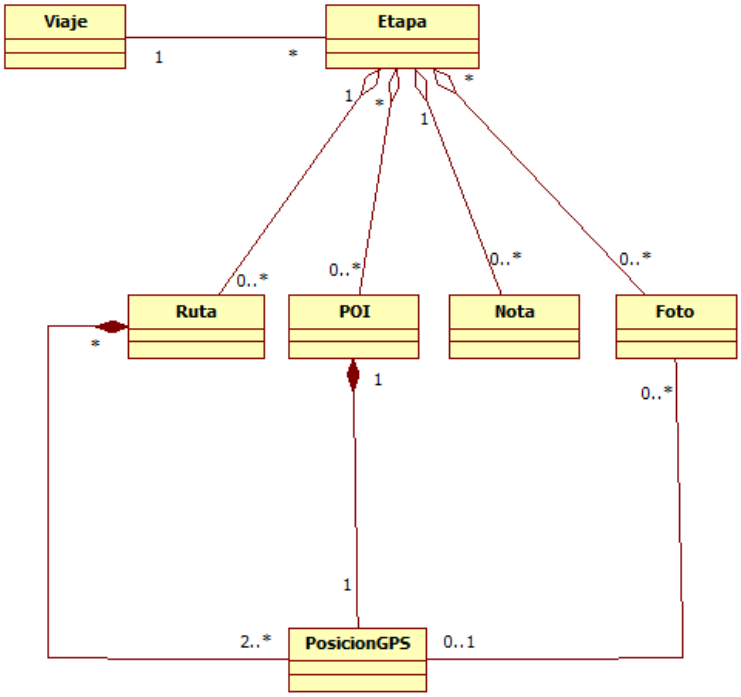


Figura 3. Diagrama de Clases

Como se puede observar, en el gráfico se encuentran las clases necesarias para la aplicación. Cabe destacar que, como se había definido durante la fase de análisis, un viaje puede agrupar varias etapas, y éstas estar compuestas por rutas, POIs, notas y

fotos. Finalmente reseñar que estos elementos, a excepción de las notas, deben almacenar su posición de GPS.

## Interfaz de Usuario

A continuación se especifica el diseño aproximado que tendrá la interfaz de usuario, señalando brevemente la funcionalidad que alojará.

### Pantalla Inicial

Como se ha comentado anteriormente, uno de los objetivos buscados en el diseño de la aplicación es la facilidad para almacenar puntos de interés sin tener que navegar por la aplicación para realizar esta tarea. Por este motivo, la pantalla principal se ha diseñado para contener el grueso de la funcionalidad de la aplicación, como puede comprobarse en la siguiente imagen:


Figura 4. Pantalla Inicial

Según se puede observar en la imagen anterior, se dispone de dos funciones diferenciadas. Por un lado, en la parte superior, se muestran el viaje y la etapa activos en ese instante. Estos elementos pueden ser cambiados en cualquier momento pulsando los botones del lado derecho, lo cual nos dará la posibilidad de cambiar por otro viaje/etapa o crear uno nuevo.

En segundo lugar, se pueden observar los 4 botones que sirven de acceso directo para almacenar la posición geográfica con una simple pulsación de botón. Para el caso de las rutas, se dispone de un tipo específico de botón que permanece pulsado mientras se graba el itinerario y finaliza con una segunda pulsación.

Finalmente, a través de la pulsación del botón físico del dispositivo denominado *Menú*, aparecerá un submenú que permitirá consultar los viajes anteriormente guardados.

Listado de Viajes

Esta pantalla tendrá la funcionalidad de seleccionar entre todos los viajes almacenados en el sistema y se visualizará siguiendo el siguiente diseño:


Figura 5. Listado de Viajes

Desde la pantalla inicial podemos acceder de dos formas al listado de viajes. Por un lado, pulsando el botón de viaje que se ha comentado anteriormente, permitiendo cambiar el viaje en activo. Desde esta pantalla también se podrá dar de alta un nuevo viaje pulsando el botón *Menú* del dispositivo.

La segunda forma de acceder al listado es desde el “modo consulta” que se ha descrito anteriormente, permitiendo seleccionar el viaje que se desea mostrar y derivando al usuario al listado de etapas de dicho viaje, tal como se verá en el siguiente punto.

Listado de Etapas

Al igual que en el caso del listado de viajes, a esta pantalla también se puede acceder para cambiar la etapa en curso desde la pantalla principal, o desde el "modo consulta" como se ha indicado previamente. El diseño sería el siguiente:


Figura 6. Listado de Etapas

Análogamente, desde esta pantalla también se puede dar de alta una nueva etapa pulsando el botón *Menú*.

Listado de Elementos

A diferencia de las anteriores pantallas de listado, en el caso de los elementos de interés almacenados en el sistema solo se podrá acceder desde el "modo consulta". Su diseño sería:

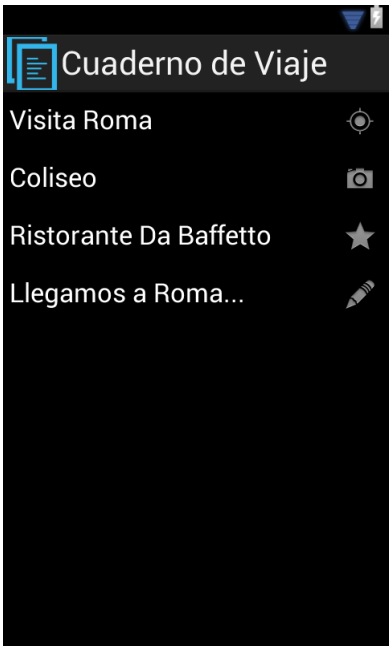


Figura 7. Listado de Elementos

Desde esta pantalla se escogerá el elemento a consultar y este se mostrará posicionado sobre un mapa de Google Maps como se detallará posteriormente. Para el caso de las notas, que no se encuentran geoposicionadas, se abrirá el formulario de datos que se mostrará en el siguiente punto.

Formulario de Datos

Cada vez que se realice la grabación de un elemento de interés a través de los botones de la pantalla inicial, se debe introducir un nombre que lo identifique y una descripción opcional mediante el siguiente formulario:

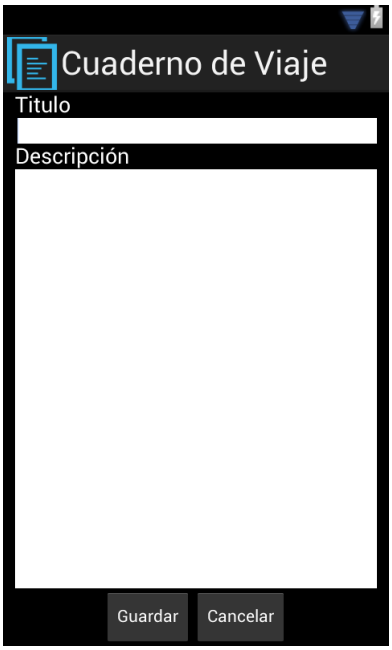


Figura 8. Formulario de Datos

Mapa

Finalmente, se dispondrá de una pantalla que presentará el mapa del lugar con el elemento de interés que se seleccione en el "modo consulta". Tendrá un diseño similar a la siguiente imagen:

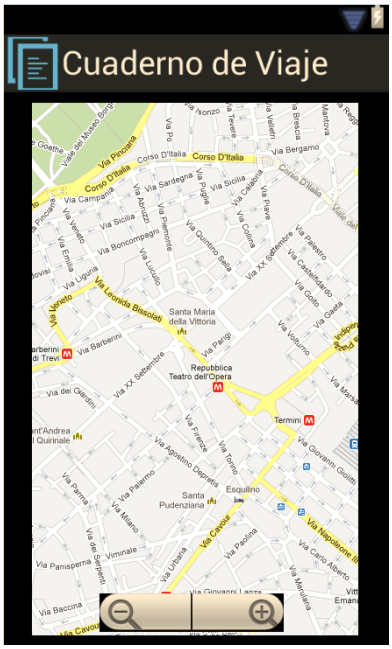


Figura 9. Mapa

Sobre el mapa se podrá aplicar el nivel de zoom que se considere oportuno, además de tener capacidad de movimiento por el mismo con total libertad. Si se realiza una

pulsación sobre el elemento de interés mostrado, se visualizará un mensaje con su descripción correspondiente y, además, permitirá mostrar la imagen en el caso de las fotografías.

### Modelo de Datos

Para el correcto desempeño de la funcionalidad de la aplicación es indispensable disponer de un sistema en el que almacenar la información. Por este motivo, y en base al diagrama de clases diseñado anteriormente, se debe generar un modelo de datos que permita organizar la información para acceder a ella de una forma lo más eficiente posible. El modelo de datos diseñado se muestra a continuación:

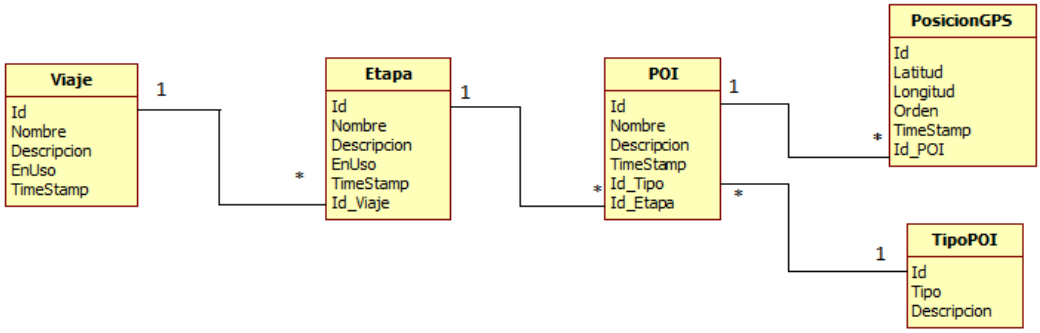


Figura 10. Modelo de Datos

## Implementación

Este punto está dedicado a detallar el proceso de implementación seguido para la construcción de la aplicación.

### Tareas Previas

Antes de comenzar la implementación es necesaria la realización de las siguientes tareas previas:

- Instalación de JDK Java 1.6
- Instalación del entorno de desarrollo Eclipse
- Instalación del SDK de Android
- Instalación ADT Plugin for Eclipse
- Creación de un dispositivo virtual Android (AVD)

## Interfaz de Usuario

Ya en la fase de diseño, se llevó a cabo la definición de la interfaz de usuario, que serviría de punto de contacto entre el usuario y el sistema. En el desarrollo en Android, la implementación de estas interfaces se realiza mediante un fichero XML, que posteriormente se asociará a la ventana a la hora de programar su funcionalidad. A continuación se va a presentar la implementación de estas interfaces.

Pantalla Inicial

Esta es, sin lugar a dudas, la interfaz más compleja de la aplicación puesto que, como se ha comentado anteriormente, busca concentrar la máxima funcionalidad. Para su construcción se han definido dos *TableLayouts* para ubicar los viajes y etapas en la primera, y los botones de acceso directo en la segunda. Los controles se van a ubicar en las celdas de estas tablas. A continuación se puede observar un fragmento en el que se muestra el formato de este tipo de ficheros XML y que se corresponde con la interfaz de la pantalla inicial.

```
    <TableLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="31dp"
        android:stretchColumns="0" >

        <TableRow
            android:id="@+id/tableRow1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:paddingLeft="10sp"
                android:text="Viajando por"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:textColor="#ffffff" />

            <TextView
                android:id="@+id/LblViaje"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:paddingLeft="10sp"
                android:paddingRight="10sp"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:textColor="#ffffff" />

            <Button
                android:id="@+id/BtnSelViaje"
                style="?android:attr/buttonStyleSmall"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="+"
                android:textColor="#ffffff" />

        </TableRow>

</TableLayout>
<TableLayout
    android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginTop="100dp"
    android:stretchColumns="0">

        <TableRow
            android:id="@+id/tableRow2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">

            <TextView
                android:id="@+id/textView3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:paddingLeft="10sp"
                android:text="Etapa"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:textColor="#ffffff" />

            <TextView
                android:id="@+id/LblEtapa"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:paddingLeft="10sp"
                android:paddingRight="10sp"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:textColor="#ffffff" />

            <Button
                android:id="@+id/BtnSelEtapa"
                style="?android:attr/buttonStyleSmall"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="+"
                android:textColor="#ffffff" />

        </TableRow>

    </TableLayout>

<TableLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true" >


    <TableRow
        android:id="@+id/tableRow3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >

        <ToggleButton
            android:id="@+id/BtnRuta"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Ruta"
            android:textColor="#ffffff"
            android:textOff="Ruta"
            android:textOn="Ruta" />

        <Button
            android:id="@+id/BtnPOI"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="POI"
            android:textColor="#ffffff" />

        <Button
            android:id="@+id/BtnFoto"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Foto"
```

```
            android:textColor="#ffffff" />

        <Button
            android:id="@+id/BtnNota"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Nota"
            android:textColor="#ffffff" />

    </TableRow>

</TableLayout>
```

Como se ha podido ver en el fragmento anterior, se han utilizado además varios controles comunes del API de Android: *Button*, y en concreto *ToggleButton* para las rutas, y TextView para las etiquetas de texto.

Listados de Viajes/Etapas

Para los listados de viajes y etapas se ha empleado un control *ListView* que será cargado en tiempo de ejecución. En la siguiente fracción de código, además, se observa que los controles no se insertan directamente sobre el XML, sino que se incluyen en un *Layout* que indica la disposición de los elementos dentro de la interfaz, siendo en este caso concreto de manera lineal.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:orientation="vertical" >
<ListView android:id="@+id/lstViajes"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Formulario de Datos

Para la interfaz de introducción de datos se incluye un nuevo tipo de control que permite esta funcionalidad, es decir, un control *EditText*. En la parte inferior se ha añadido un *TableLayout* que alojará los botones de Guardar y Cancelar.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#000000"
    android:orientation="vertical">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Titulo"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="#ffffff" />
    <EditText
        android:id="@+id/TxtTitulo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffffff"
        android:ems="10"
```

```
        android:textColor="#000000"
        android:textColorHint="#ffffff" >
    </EditText>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Descripción"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="#ffffff" />
    <EditText
        android:id="@+id/TxtDescripcion"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffffff"
        android:ems="10"
        android:inputType="textMultiLine|textLongMessage"
        android:textColor="#000000"
        android:textColorHint="#ffffff" />
    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TableRow
            android:id="@+id/tableRow1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center" >
            <Button
                android:id="@+id/BtnGuardar"
                style="?android:attr/buttonStyleSmall"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Guardar"
                android:textColor="#ffffff" />
            <Button
                android:id="@+id/BtnCancelar"
                style="?android:attr/buttonStyleSmall"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Cancelar"
                android:textColor="#ffffff" />
        </TableRow>
    </TableLayout>
</LinearLayout>
```

## Mapa

Para poder visualizar un mapa de Google Maps dentro de la aplicación se debe crear un nuevo tipo de control llamado *MapView*, el cual viene incluido dentro del API de Google. Por ese motivo, a la hora de desarrollar se tiene que hacer sobre el API de Google que incluye la versión de Android que se desee. Adicionalmente, se debe crear una *key* en el perfil de Google para que el mapa sea visible, y que se establecerá en la propiedad *apiKey* del control. El fichero XML con la implementación de esta interfaz sería el siguiente:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:background="#000000"
    android:orientation="vertical"
    android:paddingLeft="10sp"
    android:paddingRight="10sp" >

    <com.google.android.maps.MapView
    android:id="@+id/mapa"
```

```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:apiKey="0mGRtcU-VbIF1tEiJDmHr0dXF4Wx9_34CXcdZuQ"
    android:clickable="true" />
</LinearLayout>
```

Foto

Finalmente, queda un último interfaz que se debe implementar, y no es otro que la pantalla que muestra la imagen cuando pulsamos sobre una foto geoposicionada en el mapa. De igual forma que anteriormente, esta funcionalidad se limita a introducir un nuevo tipo de control que puede albergar imágenes, es decir el control *ImageView*. A continuación se expone el XML de esta interfaz:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <ImageView
        android:id="@+id/imgFoto"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:src="@drawable/ic_action_search" />
</LinearLayout>
```

## Base de Datos

Android dispone de una cualidad muy útil: la integración con SQLite de manera nativa, facilitando mucho las cosas para el trabajo con bases de datos. Para la implementación del modelo de datos diseñado anteriormente se ha desarrollado una clase que extiende de la clase *SQLiteOpenHelper*, la cual nos facilita el acceso a bases de datos. Esta clase es instanciada al comienzo de la ejecución de la aplicación, y en caso de que no exista la base de datos indicadas procederá a invocar el método *OnCreate*, el cual ejecutará las sentencias necesarias para inicializarla.

También dispone de un método *OnUpgrade* para actualizar la base de datos en caso de que fuese una versión anterior, por lo que se ha implementado el método *OnUpgrade* para que elimine la versión antigua y vuelva a crear la nueva con sus datos iniciales.

A continuación puede observarse el código de esta clase para ilustrar lo explicado:

```
public class CdVSQLiteHelper extends SQLiteOpenHelper {

        public CdVSQLiteHelper(Context context, String name, CursorFactory factory,
                        int version) {
                super(context, name, factory, version);
        }

    /**
     *  Crear la BD
     */
        public void onCreate(SQLiteDatabase db) {
                // Creamos la BD
                db.execSQL("CREATE TABLE VIAJE (ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
NOMBRE TEXT NOT NULL, DESCRIPCION TEXT, IMAGEN TEXT, ENUSO INTEGER, TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP);");
```

```
                db.execSQL("CREATE TABLE ETAPA (ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
NOMBRE TEXT NOT NULL, DESCRIPCION TEXT, IMAGEN TEXT, ENUSO INTEGER, TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP, ID_VIAJE INTEGER, FOREIGN KEY (ID_VIAJE) REFERENCES VIAJE (ID));");
                db.execSQL("CREATE TABLE TIPO_POI (ID INTEGER PRIMARY KEY AUTOINCREMENT NOT
NULL, TIPO TEXT NOT NULL, DESCRIPCION TEXT);");
                db.execSQL("CREATE TABLE POI (ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
NOMBRE TEXT NOT NULL, DESCRIPCION TEXT, IMAGEN TEXT, TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP, ID_TIPO INTEGER, ID_ETAPA INTEGER, FOREIGN KEY (ID_TIPO) REFERENCES TIPO_POI
(ID), FOREIGN KEY (ID_ETAPA) REFERENCES ETAPA (ID));");
                db.execSQL("CREATE TABLE POSICIONGPS (ID INTEGER PRIMARY KEY AUTOINCREMENT NOT
NULL, LATITUD TEXT NOT NULL, LONGITUD TEXT NOT NULL, TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP, ID_POI INTEGER, ORDEN INTEGER, FOREIGN KEY (ID_POI) REFERENCES POI (ID))");

                // Cargamos tablas maestras
                db.execSQL("INSERT INTO TIPO_POI (TIPO) VALUES ('RUTA')");
                db.execSQL("INSERT INTO TIPO_POI (TIPO) VALUES ('POI')");
                db.execSQL("INSERT INTO TIPO_POI (TIPO) VALUES ('NOTA')");
                db.execSQL("INSERT INTO TIPO_POI (TIPO) VALUES ('FOTO')");
        }

    /**
     *  Actualizar BD
     */
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
                // Eliminamos la BD Antigua
                db.execSQL("DROP TABLE IF EXISTS VIAJE");
                db.execSQL("DROP TABLE IF EXISTS ETAPA");
                db.execSQL("DROP TABLE IF EXISTS TIPO_POI");
                db.execSQL("DROP TABLE IF EXISTS POI");

                // Creamos la nueva
                onCreate(db);

        }
```

## Activities

En este punto, se va a especificar el proceso de implementación de las *Activities* de la aplicación. Para simplificar, una *Activity* se define como una ventana en la que se establece su funcionamiento y se presenta al usuario con el diseño de interfaz que se le indique (como se ha mostrado con anterioridad).

Si bien, una *Activity* está ligada a una interfaz de usuario y por lo tanto a los controles que contenga, será necesario establecerlo en el código, es decir, indicar en la *Activity* qué interfaz debe mostrar y referenciar los controles necesarios para poder operar con ellos. En el siguiente fragmento de código se puede apreciar cómo se realiza esta tarea:

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Referencias a los Controles
        final TextView lblViaje = (TextView)findViewById(R.id.LblViaje);
        final TextView lblEtapa = (TextView)findViewById(R.id.LblEtapa);
        final Button BtnSelViaje = (Button)findViewById(R.id.BtnSelViaje);
        final Button BtnSelEtapa = (Button)findViewById(R.id.BtnSelEtapa);
        final Button BtnPOI = (Button) findViewById(R.id.BtnPOI);
        final ToggleButton BtnRuta = (ToggleButton) findViewById(R.id.BtnRuta);
        final Button BtnFoto = (Button) findViewById(R.id.BtnFoto);
        final Button BtnNota = (Button) findViewById(R.id.BtnNota);
```

Revisando el código anterior se puede observar que mediante la función *setContentView* se establece que el interfaz que cargará el *Activity*, que es este caso será *activity_main.xml*, y mediante la función *findViewById* se referenciará a cada uno de los controles contenidos en ella, siendo esta la forma habitual para implementar la funcionalidad de las *Activities*, como se comprobará en los puntos siguientes dónde se detallan cada una de las *Activity* creadas en el proyecto.

Pantalla Inicial

Como se ha comentado previamente en numerosas ocasiones, la pantalla inicial es la que incluye todo el grueso de la funcionalidad de la aplicación, y por lo tanto, también es dónde se incluyen la mayoría de las particularidades del desarrollo.

En un primer momento, al arrancar la aplicación, se realiza la conexión con la base de datos, creándola en caso de no existir o actualizándola en caso de tratarse de una versión antigua. Una vez realizada la conexión con la base de datos, en función de si se ha seleccionado un viaje y una etapa se habilitarán los botones del formulario que hemos referenciado anteriormente, como puede verse en el siguiente código:

```
// Si se abre correctamente la BD
if(db!=null){
// Cargamos el viaje/etapa en uso
vActivo=vActivo.getActiveViaje(db);

if (vActivo.getId() != null) {
        lblViaje.setText(vActivo.getNombre());
        eActiva=vActivo.getActiveEtapa(db, vActivo.getId());
        BtnSelEtapa.setEnabled(true); // Activamos el botón de Etapas
        if (eActiva.getId() != null) {
                lblEtapa.setText(eActiva.getNombre());
                // Activamos los botones de accion
                BtnRuta.setEnabled(true);
                BtnPOI.setEnabled(true);
                BtnFoto.setEnabled(true);
                BtnNota.setEnabled(true);
        } else {
                // Desactivamos los botones de accion
                BtnRuta.setEnabled(false);
                BtnPOI.setEnabled(false);
                BtnFoto.setEnabled(false);
                BtnNota.setEnabled(false);
        }
}else {
        BtnSelEtapa.setEnabled(false); // Desactivamos el botón de Etapas
}

}

db.close();
```

El siguiente paso es definir qué funcionalidad van a ejecutar los botones cuando sean pulsados. En el caso de los botones dedicados a seleccionar viaje y etapa, deberán dirigirse a otra *Activity* en la que seleccionarán otro diferente o, en caso de no tener ninguno activo, ir al *Activity* que permitirá crear uno:

```
/**
 *  Si pulsamos el boton de Viajes
 */
BtnSelViaje.setOnClickListener(new OnClickListener() {
```

```
            @Override
    public void onClick(View v) {

                    // Vamos a pantalla a seleccionar viaje
            if (lblViaje.getText()==""){ // Si no hay viajes vamos a dar de alta
                    Intent intent = new Intent(MainActivity.this,ActivityData.class);
                    // Pasamos el objetivo del Alta
                    Bundle b = new Bundle();
                    b.putString("OBJETIVO", "VIAJE");

                    // Añadimos la información al intent
                    intent.putExtras(b);

                    startActivity(intent);
            } else {
                    Intent intent = new Intent(MainActivity.this,ActivityViajes.class);
                    // Pasamos el modo de consulta
                    Bundle b = new Bundle();
                    b.putString("ACCION", "MOD");

                    // Añadimos la información al intent
                    intent.putExtras(b);
                    startActivity(intent);
            }
    }

});
```

Una vez elegidos un viaje y una etapa, ya se puede comenzar a guardar elementos del viaje, para lo cual hay que completar, al igual que en la fase anterior, el evento *onClick* de cada uno de los botones. En el caso del botón que para guardar POI se ha implementado la utilidad que permite hacer uso del GPS. Esta utilidad se emplea de tal forma que al pulsar el botón el usuario se registra en el proveedor de GPS indicando el tiempo y la distancia mínima sobre la que se desea recibir posiciones (en este caso 5 segundos y 20 metros) y, posteriormente, ejecuta el código que se le indique en el evento *onLocationChanged* cuando recibe una nueva posición. El proveedor permite otras funcionalidades y eventos, pero para este caso no se han implementado. Una vez registrada la posición geográfica y almacenada en la base de datos, se redirige a la *Activity* que permite introducir los contenidos que identifican al POI. A continuación se muestra el fragmento de código que realiza esta función:

```
 /**
  *  Si pulsamos el boton de guardar POI
  */
BtnPOI.setOnClickListener(new OnClickListener() {

 public void onClick(View v) {
        // Nos registramos al provider para recibir el POI
 pd = ProgressDialog.show(MainActivity.this, "POI", "Esperando Localización...");
 locListener = new LocationListener() {

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {


        }

        @Override
        public void onProviderEnabled(String provider) {
                // TODO Auto-generated method stub

        }
```

```
                @Override
                public void onProviderDisabled(String provider) {
                        // TODO Auto-generated method stub

                }

                @Override
                public void onLocationChanged(Location location) {
                        // Guardamos la posición recibida
                        String IDPosicion;
                        pd.dismiss();
                        IDPosicion=guardaPosicion(location.getLatitude(),
location.getLongitude());
                        if(!IDPosicion.equals("")){
                                // Vamos a pantalla de almacenamiento de POI
                                Intent intent = new
Intent(MainActivity.this,ActivityData.class);
                                //Información que se pasa a la Activity Viajes
                                  Bundle b = new Bundle();
                                  b.putString("OBJETIVO", "POI");
                                  b.putString("IDETAPA", eActiva.getId());
                                  b.putString("IDPOSICION", IDPosicion);
                                  //Añadimos la información al intent
                                  intent.putExtras(b);
                                startActivity(intent);
                                locManager.removeUpdates(locListener);
                        }
                }
        };
        locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000, 20, locListener);
}
});
```

En cuanto al botón de grabar Ruta, el funcionamiento es, obviamente, muy similar al expuesto previamente, pero con alguna particularidad. En este caso, se debe controlar si el botón se ha pulsado o despulsado al ejecutar el evento, haciendo que el usuario se registre al proveedor de GPS en la primera pulsación (configurado para 15 segundos y 5 metros) y que deje de recibir posiciones con la siguiente pulsación, como se muestra a continuación:

```
        /**
         *  Si pulsamos el boton de guardar Ruta
         */
    BtnRuta.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {
                // Nos registramos al provider para recibir el POI
                locListenerRuta = new LocationListener() {

                        @Override
                        public void onLocationChanged(Location location) {
                                // Guardamos la posición recibida
                                intOrden++;
                                guardaPosicionRuta(location.getLatitude(),
location.getLongitude(),intOrden);

                        }

                        @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

                        }

                };
                if (BtnRuta.isChecked()){
                        locManagerRuta.requestLocationUpdates(LocationManager.GPS_PROVIDER,
15000, 5, locListenerRuta);
                        intOrden=0; // Inicializamos el contador con el orden de la ruta
                } else {
                        locManagerRuta.removeUpdates(locListenerRuta);
                        // Vamos a pantalla de almacenamiento de Datos
                        Intent intent = new Intent(MainActivity.this,ActivityData.class);
                        //Información que se pasa a la Activity Viajes
                        Bundle b = new Bundle();
                        b.putString("OBJETIVO", "RUTA");
                        b.putString("IDETAPA", eActiva.getId());
                        //Añadimos la información al intent
                        intent.putExtras(b);
                        startActivity(intent);
                }

        }

    });
```

Al igual que con los otros botones, se efectúa una llamada a otra *Activity* para realizar otras tareas. Como se observa, la aplicación no se limita a indicar qué *Activity* invocar, sino que envía cierta información que pueda ser necesaria. Esto se produce, como se puede ver en el fragmento de código anterior, creando un *Bundle* al que se le añaden los datos que recibirá la *Activity* indicada en el *Intent*.

En el caso del botón de almacenar Foto se van a introducir un par de conceptos distintos a los ya vistos: el acceso a la tarjeta SD, y la invocación a la aplicación de cámara nativa del dispositivo.

Antes de realizar la foto, se debe comprobar si está preparada la tarjeta SD, a través del método *getExternalStorageState* de la clase *Environment*:

```
        /**
         *  Comprueba si está preparada la tarjeta SD
         *  @return estado
         */
        private boolean SDLista(){
                String estado = Environment.getExternalStorageState();
                if (estado.equals(Environment.MEDIA_MOUNTED)){
                        return true;
                } else {
                        return false;
                }
        }
```

Una vez comprobado que la tarjeta SD se encuentra montada se debe invocar la aplicación nativa del dispositivo que permite realizar fotografías. Para ello se utiliza un

*Intent*, como ya se ha visto anteriormente, pero indicándole que la *Activity* a iniciar será *MediaStore.ACTION_IMAGE_CAPTURE*.

```
public void onClick(View v) {
        if (SDLista()){
                // Tomamos la fotografía
                SimpleDateFormat formato = new SimpleDateFormat("ddMMyyyyHHmmss");
                fichero = formato.format(new Date()) + ".jpg";
                File rutaSD = Environment.getExternalStorageDirectory();
                Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(new File(rutaSD + "/" +
fichero)) );
                startActivityForResult(intent, 0);
        } else {
                Toast aviso = Toast.makeText(MainActivity.this, "La tarjeta no está lista",
Toast.LENGTH_SHORT);
                aviso.show();
        }

};
```

En el código precedente, se observa una sutil diferencia a la hora de inicializar la *Activity*, ya que se está utilizando *startActivityForResult()* en lugar de *startActivity()* como se ha efectuado en los anteriores ejemplos. Esto va a permitir que vuelva la ejecución una vez realizada la fotografía para operar con ella en el evento *onActivityResult()*, en el cual se guardará, además, la posición geográfica correspondiente y se redirigirá al formulario de datos al igual que el caso de los POIs:

```
    /**
     *  Una vez realizada la foto con la app de camara
     */
protected void onActivityResult(int requestCode, int resultCode, Intent data){
        if (resultCode==RESULT_OK){
                // Nos registramos al provider para recibir el POI
                pd = ProgressDialog.show(MainActivity.this, "Foto", "Esperando Loc...");
                locListener = new LocationListener() {

                @Override
                public void onLocationChanged(Location location) {
                        // Guardamos la posición recibida
                        String IDPosicion;
                        pd.dismiss();
                        IDPosicion=guardaPosicion(location.getLatitude(),
                                                  location.getLongitude());
                        if(!IDPosicion.equals("")){
                                // Vamos a pantalla de almacenamiento de POI
                                Intent intent = new
                                        Intent(MainActivity.this,ActivityData.class);
                                //Información que se pasa a la Activity Viajes
                                Bundle b = new Bundle();
                                b.putString("OBJETIVO", "FOTO");
                                b.putString("IDETAPA", eActiva.getId());
                                b.putString("IDPOSICION", IDPosicion);
                                b.putString("FOTO", fichero);
                                //Añadimos la información al intent
                                intent.putExtras(b);
                                startActivity(intent);
                                locManager.removeUpdates(locListener);
                        }

                }

                @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

                }

        };
        locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000, 20, locListener);
}

}
```

El último elemento de interés que queda por registrar son las Notas, que puesto que no van geoposicionadas, se limitan a invocar el formulario de datos de igual modo que se ha visto anteriormente:

```
       /**
        *  Si pulsamos el boton de guardar Nota
        */
     BtnNota.setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {
                        // Guardamos la nota
                        Intent intent = new Intent(MainActivity.this,ActivityData.class);
                        //Información que se pasa a la Activity Viajes
                        Bundle b = new Bundle();
                        b.putString("OBJETIVO", "NOTA");
                        b.putString("IDETAPA", eActiva.getId());
                        //Añadimos la información al intent
                        intent.putExtras(b);
                        startActivity(intent);
                }
        });
```

Y para concluir con la funcionalidad de la pantalla inicial, se debe mencionar la implementación del submenú que permitirá la visualización de los elementos almacenados en cada viaje, mencionado anteriormente como “Modo Consulta”. A continuación se muestra el código relacionado con este punto:

```
    /**
     *  Crear Submenú para "Modo Consulta"
     */
    public boolean onCreateOptionsMenu(Menu menu) {
        // Menu para crear una nueva Etapa
        menu.add(Menu.NONE, 1, Menu.NONE, "Ver Viaje");
        return true;
    }

    /**
     *  Si seleccionamos "Modo Consulta"
     */
    public boolean onOptionsItemSelected(MenuItem item) {
                Intent intent = new Intent(MainActivity.this,ActivityViajes.class);
                //Información que se pasa a la Activity Viajes
        Bundle b = new Bundle();
        b.putString("ACCION", "VER");
        //Añadimos la información al intent
```

```
        intent.putExtras(b);
                startActivity(intent);
        return true;
    }
```

Listados de Viajes/Etapas/Elementos

Como se ha comentado en el apartado de Interfaz de Usuario, estas *Activies* se limitan a un simple *ListView* con el listado de viajes, etapas o elementos disponibles. La implementación de estas tres pantallas es bastante similar. Por un lado, se realiza la recepción de parámetros enviados desde la pantalla anterior, tales como la acción (recordar que a estos listados se puede acceder desde el "Modo Consulta" o bien desde los botones de selección de la pantalla principal), o el identificador del viaje o etapa; y a continuación, se realiza una consulta en base de datos para obtener el listado de entradas a cargar en el control:

```
        Bundle bnd = getIntent().getExtras();
        accion[0] = bnd.getString("ACCION");
                Cursor cViajes= db.rawQuery("SELECT NOMBRE FROM VIAJE", null);
                viajes = new String[cViajes.getCount()];
                int i=0;
                while (cViajes.moveToNext()){
                        viajes[i]=cViajes.getString(0);
                        i++;
                }
```

Posteriormente, se "inyectan" los datos en el *ListView* mediante un Adaptador:

```
// Creamos el Adaptador para la lista
ArrayAdapter<String> adViajes = new
                ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,viajes);
lstViajes.setAdapter(adViajes);
```

Por otro lado, también se debe definir e implementar la funcionalidad a ejecutar cuando se selecciona alguno de los componentes del listado, contenido en el método *setOnItemClickListener()* del componente:

```
        /**
         *  Si seleccionamos un item
         */
       lstViajes.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView<?> a, View v, int position, long id) {
                // Desactivamos todos los viajes y activamos el seleccionado
                CdVSQLiteHelper dbHelper = new CdVSQLiteHelper(ActivityViajes.this,
                                                        "CuadernoDeViajeBD", null, 1);
                SQLiteDatabase db = dbHelper.getWritableDatabase();
                // Si se abre correctamente la BD
                if(db!=null){
                        // Hallamos el ID del viaje seleccionado
                        String[] strNombreViaje={""};
                        strNombreViaje[0]=viajes[position];
                        Cursor cViajes=db.rawQuery("SELECT ID FROM VIAJE WHERE
                                                        NOMBRE=?",strNombreViaje);
                        if (cViajes.moveToNext()){
                                strIdViaje[0]=cViajes.getString(0);
                        }
                        if (accion[0].equals("MOD")){
                                // Seleccionamos el nuevo Viaje en uso
                                db.execSQL("UPDATE VIAJE SET ENUSO=0");
                                db.execSQL("UPDATE VIAJE SET ENUSO=1 WHERE ID=?", strIdViaje);
```

```
                }

            }
            db.close();
            if (accion[0].equals("MOD")){
                    // Volvemos al menú principal
                    Intent intent = new Intent(ActivityViajes.this,MainActivity.class);
                    startActivity(intent);
            } else {
                // Vamos a seleccionar la Etapa
                    Intent intent = new
                    Intent(ActivityViajes.this,ActivityEtapas.class);
                    //Información que se pasa a la Activity Etapas
                    Bundle b = new Bundle();
                    //b.putString("OBJETIVO", "ETAPA");
                    b.putString("ACCION", accion[0]);
                    b.putString("VIAJE", strIdViaje[0]);
                    //Añadimos la información al intent
                    intent.putExtras(b);
                    startActivity(intent);
            }
        }

    });
```

En el anterior fragmento se observa que, a partir del elemento seleccionado, se cargan sus datos desde la base de datos y en función de la Acción se realizan distintas tareas ("Ver" para modo consulta o "MOD" para el modo selección).

Formulario de Datos

El formulario de datos, como conviene recordar, es común para todos los elementos que requieren ser dados de alta, ya sean viajes, etapas o cualquier tipo de elemento de interés. Por este motivo, lo primero que debe realizar es recuperar la información recibida del *Intent*, que mediante la clave "OBJETIVO" diferenciará qué tipo de datos hay que almacenar, y en función de esto también recuperará el resto de datos necesarios:

```
// Recuperamos la información
Bundle bundle = getIntent().getExtras();
strObjetivo=bundle.getString("OBJETIVO");
if (strObjetivo.equals("ETAPA")) strIdViaje=bundle.getString("IDVIAJE");
if (strObjetivo.equals("RUTA") || strObjetivo.equals("NOTA"))
        strIdEtapa=bundle.getString("IDETAPA");
if (strObjetivo.equals("POI")) {
        strIdEtapa=bundle.getString("IDETAPA");
        strIdPosicion=bundle.getString("IDPOSICION");
}
if (strObjetivo.equals("FOTO")) {
        strIdEtapa=bundle.getString("IDETAPA");
        strIdPosicion=bundle.getString("IDPOSICION");
        strFoto=bundle.getString("FOTO");
}
```

Una vez se dispone del "objetivo", se implementa la funcionalidad al pulsar el botón "Guardar", que almacenará en base de datos la información introducida:

```
/**
 *  Si pulsamos el boton de Guardar
 */
btnGuardar.setOnClickListener(new OnClickListener(){

                @Override
                public void onClick(View v) {
                // Abrimos la BD
```

```
CdVSQLiteHelper dbHelper = new CdVSQLiteHelper(ActivityData.this,
                                        "CuadernoDeViajeBD", null, 1);
SQLiteDatabase db = dbHelper.getWritableDatabase();
// Si se abre correctamente la BD
if(db!=null){
// Comprobamos si el nombre ya ha sido usado
String[] Titulo = {""};
Titulo[0]=txtTitulo.getText().toString();
Cursor cObjetivo;
if(strObjetivo.equals("VIAJE")){
        cObjetivo= db.rawQuery("SELECT NOMBRE FROM VIAJE WHERE
                                        NOMBRE=?", Titulo);
} else {
        cObjetivo= db.rawQuery("SELECT NOMBRE FROM ETAPA WHERE
                                        NOMBRE=?", Titulo);
}

if(!cObjetivo.moveToNext()){
        if(strObjetivo.equals("VIAJE")){
                // Insertamos datos
                String[] datosObjetivo = {"","",""};
                datosObjetivo[0]=txtTitulo.getText().toString();
                datosObjetivo[1]=txtDescripcion.getText().toString();
                datosObjetivo[2]="1";
                db.execSQL("UPDATE VIAJE SET ENUSO=0");
                db.execSQL("INSERT INTO VIAJE (NOMBRE,DESCRIPCION,ENUSO)
                        VALUES (?,?,?)",datosObjetivo);
        } else if (strObjetivo.equals("ETAPA")) {
                // Insertamos datos
                String[] datosObjetivo = {"","","",""};
                datosObjetivo[0]=txtTitulo.getText().toString();
                datosObjetivo[1]=txtDescripcion.getText().toString();
                datosObjetivo[2]="1";
                datosObjetivo[3]=strIdViaje;
                db.execSQL("UPDATE ETAPA SET ENUSO=0");
                db.execSQL("INSERT INTO ETAPA
(NOMBRE,DESCRIPCION,ENUSO,ID_VIAJE) VALUES (?,?,?,?)",datosObjetivo);

        } else if (strObjetivo.equals("POI")) {
                // Insertamos datos
                String[] datosObjetivo = {"","","",""};
                String[] posGPS={"",""};
                String IDPOI="";
                datosObjetivo[0]=txtTitulo.getText().toString();
                datosObjetivo[1]=txtDescripcion.getText().toString();
                datosObjetivo[2]=strIdEtapa;
                datosObjetivo[3]="2";
                db.execSQL("INSERT INTO POI
(NOMBRE,DESCRIPCION,ID_ETAPA,ID_TIPO) VALUES (?,?,?,?)",datosObjetivo);
                // Enlazamos la posicion con el POI almacenado
                Cursor cPosicion= db.rawQuery("SELECT ID FROM POI WHERE
NOMBRE=? AND DESCRIPCION=? AND ID_ETAPA=? AND ID_TIPO=?", datosObjetivo);
                if(cPosicion.moveToNext()) {
                        IDPOI=cPosicion.getString(0);
                }
                posGPS[0]=IDPOI;
                posGPS[1]=strIdPosicion;
                db.execSQL("UPDATE POSICIONGPS SET ID_POI=? WHERE
                                        ID=?",posGPS);
        } else if (strObjetivo.equals("RUTA")) {
                // Insertamos datos
                String[] datosObjetivo = {"","","",""};
                String[] posGPS={""};
                String IDPOI="";
                datosObjetivo[0]=txtTitulo.getText().toString();
                datosObjetivo[1]=txtDescripcion.getText().toString();
                datosObjetivo[2]=strIdEtapa;
                datosObjetivo[3]="1";
                db.execSQL("INSERT INTO POI
(NOMBRE,DESCRIPCION,ID_ETAPA,ID_TIPO) VALUES (?,?,?,?)",datosObjetivo);
                // Enlazamos la posicion con el POI almacenado
                Cursor cPosicion= db.rawQuery("SELECT ID FROM POI WHERE
 NOMBRE=? AND DESCRIPCION=? AND ID_ETAPA=? AND ID_TIPO=?", datosObjetivo);
                if(cPosicion.moveToNext()) {
```

```
                                        IDPOI=cPosicion.getString(0);
                                }
                                posGPS[0]=IDPOI;
                                db.execSQL("UPDATE POSICIONGPS SET ID_POI=? WHERE
                                                                ID_POI=0",posGPS);
                        }else if (strObjetivo.equals("FOTO")) {
                                // Insertamos datos
                                String[] datosObjetivo = {"","","","","","};
                                String[] posGPS={"",""};
                                String IDPOI="";
                                datosObjetivo[0]=txtTitulo.getText().toString();
                                datosObjetivo[1]=txtDescripcion.getText().toString();
                                datosObjetivo[2]=strIdEtapa;
                                datosObjetivo[3]=strFoto;
                                datosObjetivo[4]="3";
                                db.execSQL("INSERT INTO POI
        (NOMBRE,DESCRIPCION,ID_ETAPA,IMAGEN,ID_TIPO) VALUES (?,?,?,?,?)",datosObjetivo);
                                // Enlazamos la posicion con el POI almacenado
                                Cursor cPosicion= db.rawQuery("SELECT ID FROM POI WHERE
NOMBRE=? AND DESCRIPCION=? AND ID_ETAPA=? AND IMAGEN=? AND ID_TIPO=?", datosObjetivo);
                                if(cPosicion.moveToNext()) {
                                        IDPOI=cPosicion.getString(0);
                                }
                                posGPS[0]=IDPOI;
                                posGPS[1]=strIdPosicion;
                                db.execSQL("UPDATE POSICIONGPS SET ID_POI=? WHERE
                                                                ID=?",posGPS);
                        }  else if (strObjetivo.equals("NOTA")) {
                                // Insertamos datos
                                String[] datosObjetivo = {"","","","","};
                                datosObjetivo[0]=txtTitulo.getText().toString();
                                datosObjetivo[1]=txtDescripcion.getText().toString();
                                datosObjetivo[2]=strIdEtapa;
                                datosObjetivo[3]="4";
                                db.execSQL("INSERT INTO POI
                (NOMBRE,DESCRIPCION,ID_ETAPA,ID_TIPO) VALUES (?,?,?,?)",datosObjetivo);
                        }
                        db.close();
                }

                }
                db.close();
                finish();
                }

});
```

O el botón “Cancelar” que no guardará la información y eliminará cualquier dato asociado para evitar inconsistencias:

```
/**
 *  Si pulsamos el boton de Cancelar
 */
btnCancelar.setOnClickListener(new OnClickListener(){

public void onClick(View v) {
                // Abrimos la BD
                CdVSQLiteHelper dbHelper = new CdVSQLiteHelper(ActivityData.this,
                                                "CuadernoDeViajeBD", null, 1);
                SQLiteDatabase db = dbHelper.getWritableDatabase();
                // Si se abre correctamente la BD
                if(db!=null){
                if (!strIdPosicion.equals("")){
                        // Eliminamos la posicion de los POIs
                        String[] posGPS={""};
                        posGPS[0]=strIdPosicion;
                        db.execSQL("DELETE FROM POSICIONGPS WHERE ID=?",posGPS);
                } else if (strObjetivo.equals("RUTA")){
                        // Eliminamos las posiciones de la ruta
                        String[] posGPS={""};
                        posGPS[0]="0";
```

```
                        db.execSQL("DELETE FROM POSICIONGPS WHERE ID_POI=?",posGPS);
                }
                }
                db.close();
        finish();
    }
    });
```

Mapa

Para mostrar el mapa en el que se visualizarán los elementos de interés, se tiene que referenciar el control y, a partir del mismo, crear un objeto *MapController* que permitirá operar con el mapa. También se va a establecer la propiedad que muestra los controles de zoom sobre el mapa y se indicará el nivel de zoom por defecto:

```
    //Obtenemos una referencia al control MapView
    mapa = (MapView)findViewById(R.id.mapa);
    control=mapa.getController();
            control.setZoom(16);
    //Mostramos los controles de zoom sobre el mapa
    mapa.setBuiltInZoomControls(true);
```

Una vez que se visualiza el mapa, se debe proceder a agregar los elementos de interés. Para ello, se ha creado una clase *POIOverlay* que extiende de *ItemizedOverlay<OverlayItem>*. Esta clase va a permitir crear una capa sobre la que se añadirán los *GeoPoints*, o puntos de localización geográfica obtenidos a partir de la latitud y longitud registrados a través del GPS durante el proceso de guardado de elementos de interés. Esta capa se incorporará a los *Overlay* que ya tuviese el mapa, de forma que se muestre sobre estas. A continuación se puede observar el código en el que se aprecia lo menciado:

```
  // Abrimos la BD
  CdVSQLiteHelper dbHelper = new CdVSQLiteHelper(this, "CuadernoDeViajeBD", null, 1);
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    // Si se abre correctamente la BD
    if(db!=null){
            Bundle bndElementos = getIntent().getExtras();
            strIdElemento[0] = bndElementos.getString("ELEMENTO");
            strIdTipo = bndElementos.getString("TIPO");
            strTitulo = bndElementos.getString("TITULO");
            strDescripcion = bndElementos.getString("DESCRIPCION");
            Cursor cElementos= db.rawQuery("SELECT LATITUD,LONGITUD FROM POSICIONGPS WHERE
                                                    ID_POI=?", strIdElemento);
            List<Overlay> capas = mapa.getOverlays();
            if(strIdTipo.equals("1")){ //Ruta
                            capaPOI = new
            POIOverlay(this,getResources().getDrawable(R.drawable.ic_action_search));
            } else if(strIdTipo.equals("2")){ //POI
                            capaPOI = new
                    POIOverlay(this,getResources().getDrawable(R.drawable.ic_poi));
            } else if(strIdTipo.equals("3")){ //Foto
                            capaPOI = new
POIOverlay(this,getResources().getDrawable(R.drawable.ic_photo),bndElementos.getString("FOTO"));
            }

    while (cElementos.moveToNext()){
            Double latitud = Double.valueOf(cElementos.getString(0)).doubleValue() * 1E6;
            Double longitud = Double.valueOf(cElementos.getString(1)).doubleValue() * 1E6;
            //latitud = 40.20*1E6;
            //longitud = -3.65*1E6;
            GeoPoint punto = new GeoPoint(latitud.intValue(), longitud.intValue());
```

```
            control.setCenter(punto);
            //Añadimos la capa con el POI
            capaPOI.addPOI(punto,strTitulo,strDescripcion);
            capas.add(capaPOI);
            mapa.postInvalidate();
        }
}
```

Para dibujar los puntos en el mapa se ha sobrescrito el método *draw* de la clase POIOverlay, de forma que también dibuje las rutas uniendo los puntos de la misma con una línea:

```
        /**
         * Dibujar los POIs en el mapa
         */
    public void draw(Canvas canvas, MapView map, boolean shadow){

        super.draw(canvas, map, false);
        if (listaPOIs.size()>1){
                Paint mPaint = new Paint();
                mPaint.setDither(true);
                mPaint.setColor(Color.RED);
                mPaint.setStyle(Paint.Style.FILL_AND_STROKE);
                mPaint.setStrokeJoin(Paint.Join.ROUND);
                mPaint.setStrokeCap(Paint.Cap.ROUND);
                mPaint.setStrokeWidth(2);

                Point p1 = new Point();
                Point p2 = new Point();
                for(int i=1;i<listaPOIs.size();i++){
                        Projection projection = map.getProjection();
                        projection.toPixels(listaPOIs.get(i-1).getPoint(), p1);
                        projection.toPixels(listaPOIs.get(i).getPoint(), p2);
                        canvas.drawLine(p1.x, p1.y, p2.x, p2.y, mPaint);
                }
        }
    }
```

Adicionalmente, se ha implementado un cuadro emergente que muestra la información sobre el elemento de interés cuando se pulsa sobre él. Para ello se ha sobrescrito el método *onTap* de la siguiente forma:

```
        /**
         * Al pulsar sobre el POI muestra sus datos
         */
    protected boolean onTap(int index) {
            OverlayItem item = listaPOIs.get(index);
            AlertDialog.Builder dialog = new AlertDialog.Builder(contexto);
        dialog.setTitle(item.getTitle());
        dialog.setMessage(item.getSnippet());
        dialog.setIcon(icono);
        if (isPhoto){
            dialog.setNegativeButton("Ver Foto", new OnClickListener() {

                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                    // Mostramos la foto
                    // Vamos al mapa y geoposicionamos el elemento
                    Intent intent = new Intent(contexto,ActivityFoto.class);
                                    //Información que se pasa al Mapa
                    Bundle b = new Bundle();
                    b.putString("FOTO", strFoto);
                    intent.putExtras(b);
                    contexto.startActivity(intent);

                            }
```

```
                    });
            }
            dialog.show();
            return true;
        }
```

Foto

Desde el cuadro emergente mencionado en el anterior punto, se observa que, para el caso de las fotografías, se ofrece la opción de visualizarlas. Para mostrar la instantánea, simplemente se debe referenciar la ruta en la que se encuentra almacenada en la tarjeta SD y establecer la propiedad *setImageBitmap* del *ImageView* mencionado en el apartado de interfaz de usuario, tal como se expone a continuación:

```
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_photo);

                // Referenciamos los controles
                ImageView foto = (ImageView) findViewById(R.id.imgFoto);

                // Recuperamos la foto
        Bundle bndElementos = getIntent().getExtras();
        String strFoto = bndElementos.getString("FOTO");
        File rutaSD = Environment.getExternalStorageDirectory();
        Bitmap bmFoto = BitmapFactory.decodeFile(rutaSD + "/" + strFoto);
        foto.setImageBitmap(bmFoto);
        }
```

## AndroidManifest.xml

Finalmente, dentro del apartado de implementación, se deben establecer una serie de aspectos formales de la aplicación como son los componentes de la misma, su configuración, permisos, etc. Estos elementos se especifican dentro del fichero AndroidManifest.xml:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tfc.cuadernodeviaje"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
    <permission
          android:name="com.example.mapdemo.permission.MAPS_RECEIVE"
          android:protectionLevel="signature"/>
<uses-permission android:name="com.example.mapdemo.permission.MAPS_RECEIVE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
```

```
        <uses-library android:name="com.google.android.maps" />
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".ActivityViajes"></activity>
        <activity android:name=".ActivityEtapas"></activity>
        <activity android:name=".ActivityData"></activity>
        <activity android:name=".ActivityElementos"></activity>
        <activity android:name=".ActivityMap"></activity>
        <activity android:name=".ActivityFoto"></activity>

    </application>

</manifest>
```

## Plan de Pruebas

La aplicación ha seguido un plan de pruebas unitarias durante el desarrollo a través del AVD, de forma que se han testeado todos los componentes por separado. Una vez implementada se han realizado un conjunto global de pruebas sobre un dispositivo real siguiendo la siguiente planificación:

- Crear Viajes
- Crear Etapas sobre todos los viajes anteriores
- Cambiar Viaje
- Cambiar Etapa
- Grabar POI
- Grabar Foto
- Grabar Nota
- Grabar Ruta
- Durante la grabación de Ruta:
    - Grabar POI
    - Grabar Foto
    - Grabar Nota
- Ver en el mapa:
    - Ver POI
    - Ver Ruta
    - Ver Foto
        - Visualizar la foto
- Ver Nota
- Modificar Nota

## Planes de Futuro

La aplicación Cuaderno de Viaje no se limita a la funcionalidad mostrada en este documento, sino que ofrece numerosas líneas de desarrollo futuras, partiendo de la base de esta primera versión. Algunas de estas funcionalidades serían:

- Desarrollo de funcionalidades no implementadas en la primera versión debido a las limitaciones temporales, como son las indicaciones de "Como llegar" a los elementos seleccionados, o dar de baja viajes, etapas y elementos de interés.
- Posibilidad de mostrar todos los elementos de un viaje o etapa juntos en el mapa.
- Mejora visual de la interfaz de usuario.
- Respaldo de los datos a través de Internet, asociados a un perfil personal de usuario.
- Compartir POIs, rutas, fotos o incluso viajes completos con otros usuarios.
- Soporte para tablets aprovechando las posibilidades de una mayor pantalla.
- Sistema de visualización de viajes a modo de presentación.
- Posibilidad de consultar los viajes a través de una página web.

# Conclusiones

Llegados a este punto y si echamos la vista atrás para observar todo el camino recorrido, no solo durante el proceso de desarrollo de este TFC, sino a lo largo de toda mi etapa de formación dentro de la UOC, puedo concluir que han sido esa experiencia y aprendizaje los que me han permitido poder llevar a buen término el proyecto.

Han pasado varios meses desde que, al comenzar este trabajo, se apuntaban unos objetivos bien marcados, y que en estos momentos, haciendo balance, podemos dar por satisfechos; sin obviar las metas alcanzadas durante este tiempo y que, a priori, no se encontraban definidas como tales.

Este TFC ha sido mi primer paso dentro del mundo del desarrollo de las aplicaciones móviles y me ha permitido adquirir capacidades y habilidades para reforzar mi perfil profesional.

Por otro lado, he podido experimentar en primera persona toda la evolución del desarrollo y gestión de un proyecto completo, desde la planificación, pasando por la implementación y posterior presentación, asumiendo, además, todos los roles implicados en el proceso.

Por supuesto, llegar hasta aquí me ha permitido ser consciente de que, gracias a la formación recibida y a la práctica realizada, soy capaz de dar respuesta a una necesidad por medio de un proyecto informático, habiendo conseguido, de forma paralela, una mejora de mis habilidades creativas y técnicas.

Pero, finalmente, si algo destaca de todo este proceso, es la experiencia adquirida en base a los problemas que iban apareciendo durante el transcurso del proyecto y que, sin duda, reforzará mi competencia para afrontar futuras iniciativas.

## Bibliografía


BURNETTE, Ed. “Hello, Android. Introducing Google’s Mobile Development Platform. Third Edition” (2010). Pragmatic Programmers.

EOI MEDIA. Desarrollo en Android [En línea] http://www.youtube.com/watch?v=Qw5AB8tKLOs&list=PL17B02FFA8CDA7966 [fecha de consulta: Octubre 2012]

GOMEZ, Salvador . sgoliver.net blog [En línea] http://www.sgoliver.net/blog/?page_id=3011 [fecha de consulta: Septiembre – Diciembre 2012]

GOOGLE. Android Developers [En línea] http://developer.android.com/index.html [fecha de consulta: Septiembre – Diciembre 2012]

KASYLES. Procesos Unificados y AUP [En línea] http://kasyles.blogspot.com.es/2008/10/procesos-unificados-y-aup.html [fecha de consulta: Octubre 2012]

NOSINMIUBUNTU. Google Maps en Android [En línea] http://www.nosinmiubuntu.com/2012/05/google-maps-en-android-iv.html [fecha de consulta: Noviembre 2012]

STACKOVERFLOW. Questions [En línea] http://stackoverflow.com/questions [fecha de consulta: Noviembre - Diciembre 2012]

UC3M. Curso Android [En línea] https://sites.google.com/site/androiduc3m/ [fecha de consulta: Octubre – Noviembre 2012]

WIKIMEDIA. Wikipedia, la enciclopedia libre [En línea] http://es.wikipedia.org [fecha de consulta: Octubre – Diciembre 2012]

SQLITE. Documentation [En línea] http://www.sqlite.org/docs.html [fecha de consulta: Noviembre 2012]

# Anexos

## ANEXO I. Plan de Trabajo

El contenido correspondiente al Plan de Trabajo del Proyecto, que contiene la planificación del mismo se encuentra en el fichero anexo llamado “ANEXO I. Plan de Trabajo”.

## ANEXO II. General Public License v3

El código de la aplicación se encuentra licenciado bajo los términos descritos en la licencia GPLv3 que puede ser consultada en el documento anexo titulado “ANEXO II. General Public License”.

## ANEXO III. GNU Free Documentation License

El presente documento se encuentra licenciado bajo los términos descritos en la licencia GFDL que puede ser consultada en el documento anexo titulado “ANEXO III. GNU Free Documentation License”.