Universidad
de La Laguna

Escuela Superior de
Ingeniería y Tecnología
Sección de Ingeniería Informática

Trabajo de Fin de Grado

PateosTF

Iván González Suárez

La Laguna, 8 de junio de 2015

D. **Francisco de Sande González**, con N.I.F. 42.067.050-G profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

"*PateosTF*"

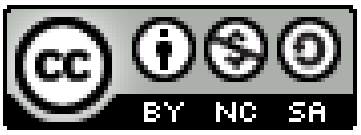
ha sido realizada bajo su dirección por D. **Iván González Suárez**, con N.I.F. 78.643.491-M

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 8 de junio de 2015

Agradecimientos

Agradezco enormemente el trato recibido por Francisco de Sande González en calidad de tutor académico así como su labor como profesor en varias asignaturas del Grado en Ingeniería Informática proyectando siempre una actitud contagiosa orientada al trabajo duro y a la perfección.

Licencia



Resumen

El objetivo de este trabajo ha sido crear una aplicación para dispositivos Android que permita la descarga y visualización de rutas para practicar senderismo.

Partiendo de los conocimientos en Java obtenidos en la asignatura: 'Programación de aplicaciones interactivas' impartida en el itinerario de computación en el tercer curso del Grado en Ingeniería Informática de la Universidad de La Laguna, se espera adquirir los conocimientos básicos necesarios para desarrollar aplicaciones en Android.

Del mismo modo, la inmersión en un proyecto de estas dimensiones implica el aprendizaje de una forma de administrar el tiempo y los recursos que no ha sido precisada anteriormente.

También se espera adquirir soltura y familiarizarse con el uso de APIs y servicios externos como pueden ser la GoogleMapsAPI o Firebase, habituales en el contexto del desarrollo de aplicaciones para Android.

Palabras clave: Aplicaciones Android, GoogleMapsAPI, Firebase.

Abstract

The aim of this work has been the development of an application for Android devices that allows to download and visualize hicking trails.

Based on the knowledge of Java obtained in the subject: "Programming Interactive Applications" taken in the Computer itinerary in the third year of the Universidad de La Laguna Degree in Computing Engineering, in this work we expect to acquire the basic knowledge needed to develop Android applications.

Moreover, a project's immersion at this level requires to develop skills to manage time and resources that were not needed before.

It is expected to use APIs and external services such as the GoogleMapsAPI or Firebase, usual in the context of Android application development.

Keywords: *Application for Android, GoogleMapsAPI, Firebase.*

Índice general

Índice de figuras

Introducción

En este documento se recoge el trabajo de investigación y desarrollo realizado por el autor para completar su Trabajo de Fin de Grado (TFG), finalizando con el mismo los estudios en el Grado en Ingeniería en Informática cursados en la Escuela Superior de Ingerniería y Tecnología - Sección Informática (ESIT-Informática) de la ULL.

Capítulo 1

Objetivos

El presente TFG ha tenido objetivos centrados en varios aspectos de las ciencias de la computación. Destacan los referentes a la producción de software, y aquellos relacionados con la ingeniería del software, así como los relativos a la programación de aplicaciones en Android. A continuación se enumeran detalladamente algunos de los objetivos que nos propusimos abordar en este TFG:

- Abordar por primera vez el diseño y desarrollo de un proyecto de ingeniería de software.
- Adquirir conocimientos básicos sobre conceptos, modelos, técnicas y herramientas asociadas con la programación de aplicaciones en Android.
- Familiarización con el uso de librerías externas para el desarrollo de aplicaciones Android.
- Habituación en la gestión de proyectos utilizando *Bitbucket* [4].
- Puesta en práctica de los conocimientos del Grado en Ingeniería Informática.
- Creación de una memoria técnica sobre la aplicación desarrollada en el Trabajo de Fin de Grado.

- Adquirir conocimientos sobre *LaTeX* [29], un sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica.

Capítulo 2

Herramientas Software de desarrollo en Android

En este capítulo se describen las herramientas software necesarias para la elaboración del TFG.

2.1. Android Studio

Android Studio [11] es un nuevo entorno de desarrollo integrado para el sistema operativo Android comercializado por Google, diseñado para ofrecer nuevas herramientas para el desarrollo de aplicaciones y alternativa al entorno Eclipse [5], hasta ahora el IDE más utilizado.

¿Qué ofrece Android Studio?

- Un entorno de desarrollo claro y robusto.
- Facilidad para testear el funcionamiento en diversos tipos de dispositivos.
- Asistentes y plantillas para los elementos comunes de programación en Android.

- Un completo editor con muchas herramientas extra para agilizar el desarrollo de nuestras aplicaciones.

2.1.1. Instalación

Para instalar Android Studio es necesario disponer del Java Development Kit (JDK) [43]. Para ello se han seguido los pasos listados en el post de Wikihow: *"Cómo instalar Oracle Java en Ubuntu"* [33].

Una vez completada la instalación del JDK, se procede a la descarga del AndroidStudio [11], del SDK de Android [12] y de todos los paquetes del SDK [13] necesarios para asegurar la compatibilidad con los dispositivos Android en los que se desee desarrollar.

2.1.2. Primeros pasos en Android Studio

Antes de comenzar con el desarrollo de la aplicación propuesta para el TFG, se realizaron una serie de tutoriales [19, 15, 17, 16, 14, 18] para familiarizarse con el uso de AndroidStudio así como de todo el entorno Android.

2.2. GoogleMaps API

GoogleMaps API [20] es un conjunto de APIs que permiten superponer datos en un mapa de Google personalizado, crear aplicaciones móviles con la poderosa plataforma de cartografía de Google, incluyendo imágenes de satélite, vista de la calle, etc. Con la cobertura global más precisa del mundo y una comunidad activa llevando a cabo actualizaciones diarias, los usuarios se benefician de un servicio de mejora continua.

2.2.1. Primeros pasos con GoogleMaps API

Para comprender el uso de GoogleMaps API es recomendable seguir los pasos de los tutoriales de Pablo Bascuñana Saiz [36]. *"Crean-*

do una Google Maps Activity con AndroidStudio" [34] muestra cómo crear una Google Maps Activity, actualizar el SDK de Android y cómo obtener la API Key de Google. Y el tutorial *"Jugando con GoogleMaps en Android"* [35] se encarga de mostrar la forma de añadir marcadores, hacer zoom en el mapa, definir la información de los marcadores, mostrar la posición actual del dispositivo en el mapa y cómo cambiar el icono de un marcador y el tipo de mapa.

2.3. Firebase

Firebase [8] es un proveedor de servicios en la nube y *back-end as a service (BaaS)* [42] con sede en San Francisco, California. La compañía fabrica una serie de productos para desarrolladores de software que crean aplicaciones móviles o web. La compañía fue adquirida por Google en octubre de 2014.

El principal producto de Firebase es una base de datos operada en tiempo real que proporciona una API que permite a los desarrolladores almacenar y sincronizar datos a través de múltiples clientes.

Para integrar Firebase en una aplicación para dispositivos Android se recomienda seguir el tutorial *"5 Minutes Quickstart: Android Quickstart"* [7].

Capítulo 3

Descripción de la aplicación

3.1. Introducción

`PateosTF` es una aplicación para dispositivos Android (App) que permite la descarga y visualización de rutas para practicar senderismo. Pondrá al alcance del usuario una base de datos alimentada por desarrolladores y el propio usuario, de tal forma que desde un dispositivo Android se visualice toda la información referente a una ruta (localización, longitud, inclinación, ...). Asimismo, para cada uno de los senderos disponibles se da la posibilidad de crear una ruta desde la localización actual del dispositivo hasta el comienzo un sendero utilizando la aplicación Maps [23] que debe ser instalada previamente.

Wikiloc y OruxMaps son dos de las aplicaciones que se han tomado como referencia a la hora de desarrollar `PateosTF`. Revisamos a continuación las características más relevantes de estos dos servicios en relación con la aplicación objeto de este Trabajo.

3.1.1. Wikiloc

Wikiloc [40] es un servicio web gratuito para visualizar y compartir rutas y puntos de interés GPS. Utilizando software libre y la API de Google Maps, Wikiloc hace la función de base de datos personal de localizaciones GPS. Desde cualquier acceso a Internet un usuario de GPS puede cargar sus datos GPS y al momento visualizar la ruta y puntos de paso o *waypoints* con distinta cartografía de fondo, incluidos servidores de mapas externos WMS (Web Map Service) o descargarlo a Google Earth para ver en 3D. Paralelamente se muestra el perfil de altura, distancia, desniveles acumulados y las fotos o comentarios que el usuario quiera añadir.

El valor añadido es la posibilidad de compartir al instante estas localizaciones con los otros usuarios de Wikiloc, que pueden visualizar, valorar y descargar la ruta desde cualquier punto con conexión a Internet.

En la aplicación para Android (véase Figura 3.1) se implementan las funcionalidades del servicio web y además, como servicio de pago, se ofrece la funcionalidad que permite la navegación sobre una ruta.

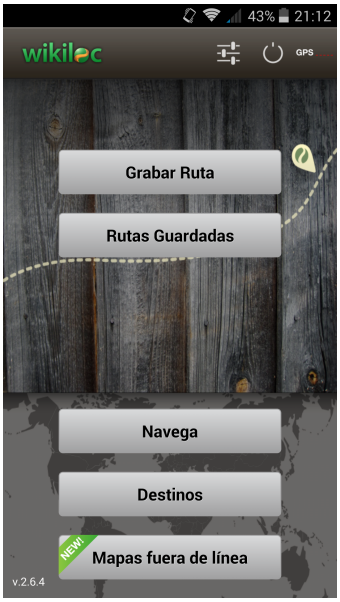


Figura 3.1: Pantalla de inicio de Wikiloc

3.1.2. Oruxmaps

OruxMaps [32] (figura 3.2) es una completísima aplicación para los amantes del deporte al aire libre. Se pueden ver mapas, importarlos, guardar rutas y ver las estadísticas de las rutas realizadas por el usuario de las mismas.

OruxMaps está pensado para los aficionados al senderismo o a la BTT (bicicleta todo terreno); ello se demuestra por lo especializado de sus opciones y características, que cubren las necesidades que pueden surgir a la hora de trazar recorridos usando el GPS: añadir *waypoints*, tomar fotos, etc.

Permite registrar recorridos y guardarlos en los populares formatos GPX y KML. Asimismo, se pueden importar los mapas de otros usuarios y seguirlos. A ello se suman las prácticas estadísticas de OruxMaps y la posibilidad de compartir los recorridos en otros servicios web como Everytrail [6], Gpsies [24] o mapmytracks [30].

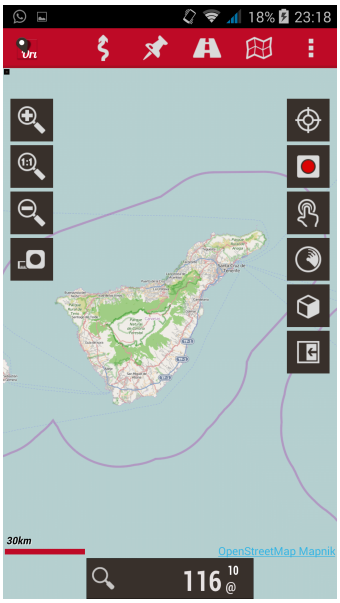


Figura 3.2: Pantalla de inicio de Oruxmaps

3.2. Pantallas

En este apartado se hará una descripción de la interfaz gráfica y de las funcionalidades de cada una de las pantallas de `PateosTF`.

Antes de proseguir en la descripción de la aplicación debemos definir varios términos:

- El término *Activity* es utilizado para refererirse al componente de la aplicación que provee al usuario de una interfaz gráfica para interactuar con ella y realizar una acción específica, en comparación con una aplicación de escritorio, sería como una ventana. En adelante, los términos pantalla y *Activity* serán utilizados indistintamente haciendo referencia al mismo concepto.

- El término *Fragment* podría definirse como una porción de la interfaz de usuario que puede añadirse o eliminarse de una interfaz de forma independiente al resto de elementos de la actividad, y que por supuesto puede reutilizarse en otras actividades.

- Un *Dialog* es una pequeña ventana que sugiere al usuario que debe tomar una decisión o introducir información adicional

En la figura 3.3 se muestra un diagrama de flujo de las *Activities* de la aplicación. Está dividido en tres niveles, y desde cada uno de ellos sólo se puede acceder al nivel inmediatamente superior e inferior. Las pantallas de un mismo nivel no son accesibles entre sí a no ser que esté indicado explícitamente.

InfoFragment, MapFragment y *ChartFragment* forman parte de un mismo *Activity* y todos ellos son accesibles partiendo desde cualquier *Fragment* a través de la selección del icono correspondiente en la barra superior.

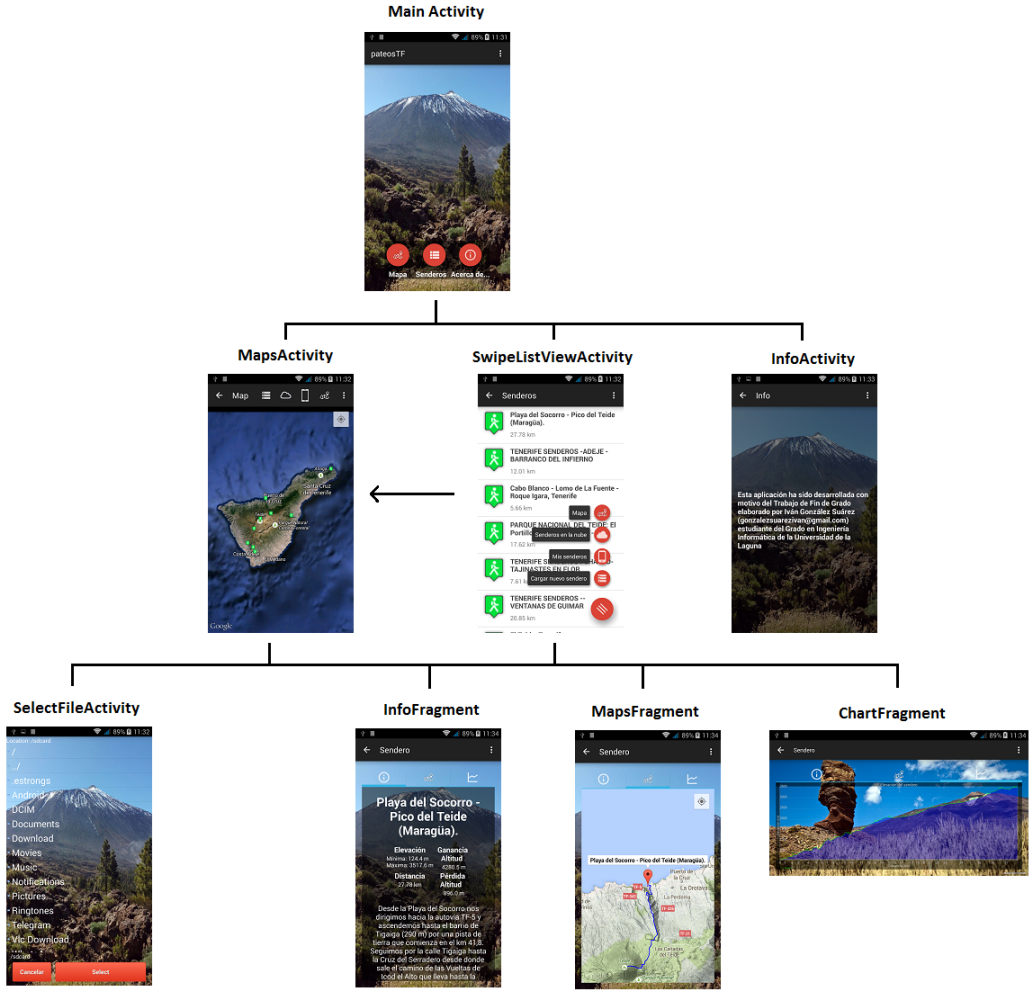


Figura 3.3: Diagrama de flujo de las *Activities* de `PateosTF`

3.2.1. Pantalla principal (*MainActivity*)

La pantalla principal (figura 3.4) es la primera en mostrarse al usuario una vez inicia la aplicación. Encontrará un conjunto de botones que le permite dirigirse a la pantalla que muestra los senderos disponibles sobre un mapa de Tenerife (*MapsActivity*, véase apartado 3.2.2), a la pantalla que muestra un listado de los mismos (*SwipleListViewActivity*, véase apartado 3.2.3) y a la pantalla que muestra la información de la aplicación (*InfoActivity*, véase apartado 3.2.6).

La primera vez que se inicia `PateosTF`, si el dispositivo Android tiene conexión a Internet, se almacenan en la base de datos los senderos guardados en Firebase para evitar un consumo de la tarifa de datos innecesario ya que la base de datos en la nube no es actualizada con frecuencia.

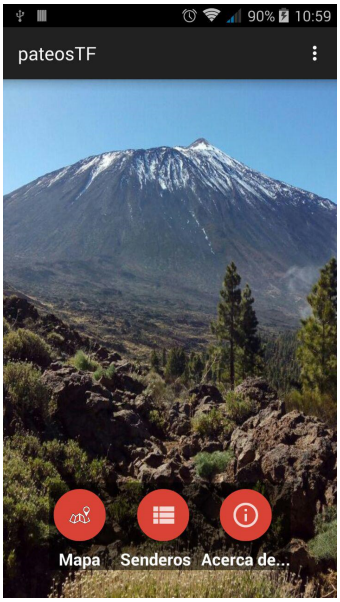


Figura 3.4: MainActivity

3.2.2. Pantalla con el mapa (*MapsActivity*)

En esta pantalla (figura 3.5) aparece el mapa de Tenerife, con un marcador en el punto donde comienza cada uno de los senderos

registrados en la base de datos en la nube. El usuario podrá solicitar que se muestren los senderos que ha introducido manualmente en la base de datos, así como volver a mostrar los senderos descargados de Firebase.

Cuando se seleccione el marcador de un sendero en el mapa, aparecerá su ruta correspondiente renderizada de forma que el sendero ocupe la mayor parte de la pantalla.

Si se vuelve a pulsar el marcador de un sendero ya dibujado, se muestra un *Dialog* con un resumen de la información de un sendero (nombre del sendero, distancia en kilómetros, elevación mínima y máxima en metros y la ganancia y pérdida en altitud en metros).

Si se pulsa sobre el botón volver del *Dialog*, éste se cerrará y volveremos a ver el mapa, en cambio, si pulsamos el botón info, se dirigirá al usuario a la pantalla que muestra toda la información disponible de un sendero (véase apartado 3.2.5).

Como funcionalidades secundarias del *MapsActivity* disponemos de:

- Cargar un nuevo sendero desde un fichero GPX (véase apartado 3.2.4) previamente descargado por el usuario.
- Cambiar el tipo de mapa.

3.2.3. Pantalla con la lista de Senderos (*SwipeListViewActivity*)

Esta pantalla (figura 3.6) es la encargada de mostrar en una lista los senderos descargados del servidor o los proporcionados por el usuario. Cada uno de los elementos de la lista estará compuesto por una imagen, el nombre del sendero y la distancia del mismo.

Cada uno de los items de la lista podrá ser desplazado tanto a la izquierda como a la derecha. Al desplazarlo, aparecen en pantalla dos botones, el botón Info dirige al usuario a la pantalla que muestra toda la información de un sendero (véase apartado 3.2.5), también

Figura 3.5: Sendero seleccionado

se le dirige a esta pantalla en caso de que el usuario, sin deslizar un item, lo seleccione, y el botón Mapa lo dirige al *MapsActivity* con el sendero seleccionado ya dibujado.

En este *Activity* aparece un conjunto de botones que permite al usuario dirigirse al *MapsActivity* (véase apartado 3.2.2), mostrar los senderos del servidor, mostrar los senderos del usuario o dirigirse a la pantalla de selección de ficheros (véase apartado 3.2.4) para cargar un nuevo sendero en la base de datos.

3.2.4. Pantalla para añadir senderos (*SelectFileActivity*)

Esta pantalla (figura 3.7) permite al usuario seleccionar un fichero GPX, que previamente ha descargado, para añadir un nuevo sendero a la base de datos.

Para ello se muestra el sistema de ficheros y se le da al usuario la opción de navegar entre los directorios del mismo. Una vez escogido el deseado, pulsando el botón de selección se comienza el proceso de

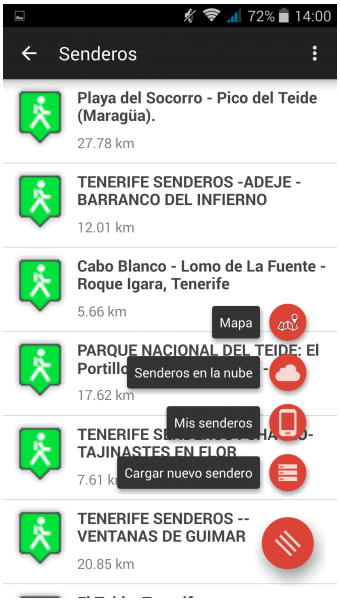


Figura 3.6: Menú de SwipeListViewActivity

parseo del fichero GPX.

Si éste finaliza con éxito se añade a la base de datos el sendero contenido en dicho fichero y se refresca la interfaz del *Activity* desde el cual el usuario seleccionó la opción de añadir un nuevo sendero.

Sólo pueden añadirse nuevos senderos desde *MapsActivity* (véase apartado 3.2.2) y desde la pantalla que muestra la lista de senderos (véase apartado 3.2.3).

3.2.5. Pantalla con la información de un sendero (*TrailActivity*)

En esta pantalla se muestra toda la información que se tiene sobre un sendero. Está formado por tres *Fragments*, los cuales pueden ser seleccionados a través los iconos presentes en la barra localizada en la parte superior.

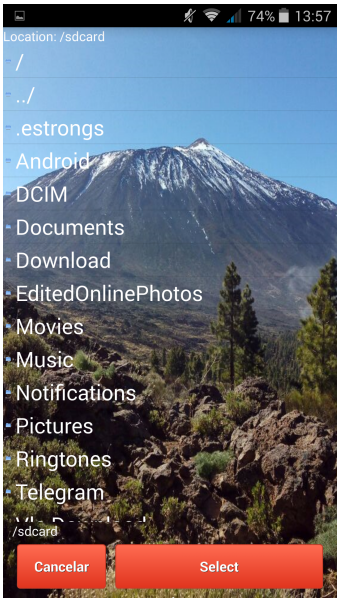


Figura 3.7: SelectFileActivity

InfoFragment

InfoFragment (figura 3.8) es el encargado de mostrar la información recogida en el fichero GPX.

- Nombre del sendero
- Elevación máxima y mínima en metros
- Ganancia y pérdida de altitud en metros
- Distancia en kilómetros
- Descripción aportada por el creador del fichero GPX.

MapFragment

MapFragment (figura 3.9) es el encargado de mostrar el mapa con el sendero dibujado. No se pueden realizar sobre él las acciones implementadas en el *MapsActivity* (véase apartado 3.2.2). En cambio,

Figura 3.8: *InfoFragment*

si se pulsa el marcador, aparecen dos botones los cuales nos dirigen a la aplicación Maps [23] instalada en el dispositivo. Dicha aplicación es capaz de crear una ruta desde el punto donde se encuentre el dispositivo hasta el comienzo del sendero, del mismo modo da al usuario la posibilidad de utilizar el *StreetView* [22] en dicho punto.

ChartFragment

ChartFragment (figura 3.10) es el encargado de mostrar una gráfica con la orografía del sendero. Sobre dicha gráfica se puede hacer zoom, así como desplazarse en todas las direcciones. Cuando el dispositivo está en vertical, inicialmente se muestran en la gráfica los primeros 5 kilómetros, para intentar que la visión sea más real y menos compacta. En cambio, cuando el dispositivo está en horizontal, en la gráfica se muestran todos los puntos que componen un sendero.

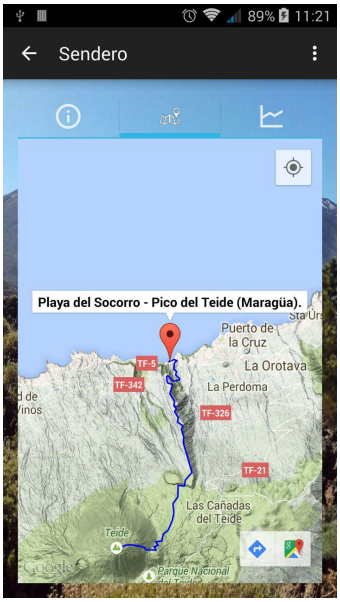


Figura 3.9: Apariencia de *MapFragment* cuando se pulsa el marcador

3.2.6. Pantalla con la información de la aplicación (*InfoActivity*

En esta pantalla (figura 3.11) se muestra el siguiente mensaje al usuario:

"Esta aplicación ha sido desarrollada como Trabajo de Fin de Grado elaborado por Iván González Suárez (gonzalezsuarezivan@gmail.com) estudiante del grado en Ingeniería Informática en la Universidad de la Laguna"

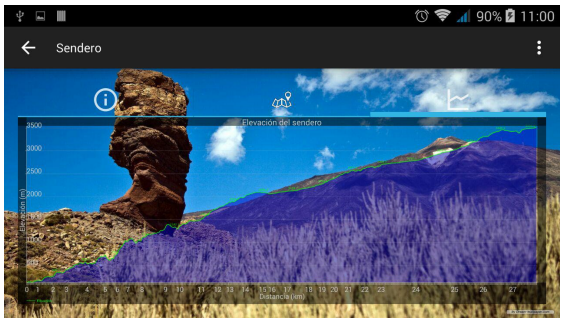


Figura 3.10: *ChartFragment* con el dispositivo en horizontal

Figura 3.11: *InfoActivity*

Capítulo 4

Desarrollo

No es objeto de esta memoría explicar exhaustivamente cada una de las clases y métodos que componen `PateosTF`. En este capítulo se describe la implementación de las *Activities* y clases más relevantes de la aplicación.

4.1. Clase *GPXData*

GPX, o **GPS eXchange Format** (Formato de Intercambio GPS) es un esquema XML ideado para transferir datos GPS entre aplicaciones. El listado 4.1 es un ejemplo de la estructura de este tipo de ficheros. Las etiquetas más importantes, cuya informción es extraída desde `PateosTF`, son las siguientes:

- `<name>Nombre del sendero</name>` (Línea 4)
- `<desc>Descripción del sendero</desc>` (Línea 5)
- `<trkseg>Listado de puntos</trkseg>` (Líneas 6 – 31)

```
<trkpt lat="latitud"lon="longitud">
 <ele>Elevación en metros</ele>
 <time>Hora a la que fue tomado el punto</time>
</trkpt>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx creator="Wikiloc - http://www.wikiloc.com" version="1.1" ...>
  <trk>
    <name>PARQUE NACIONAL DEL TEIDE: El Portillo - La Fortaleza - El Portillo</name>
    <cmt>PARQUE NACIONAL DEL TEIDE: El Portillo - La Fortaleza - El Portillo </cmt>
    <desc>El Parque Nacional del Teide, situado en la parte central de la isla de Tenerife[...]
        colorido.</desc>
    <trkseg>
      <trkpt lat="28.304098" lon="-16.566439">
        <ele>2060.2</ele>
        <time>2014-02-05T09:24:26Z</time>
      </trkpt>
      <trkpt lat="28.304102" lon="-16.566443">
        <ele>2059.8</ele>
        <time>2014-02-05T09:24:32Z</time>
      </trkpt>
      <trkpt lat="28.304182" lon="-16.566458">
        <ele>2061.2</ele>
        <time>2014-02-05T09:25:09Z</time>
      </trkpt>
      <trkpt lat="28.304274" lon="-16.566489">
        <ele>2062.2</ele>
        <time>2014-02-05T09:25:20Z</time>
      </trkpt>
      <trkpt lat="28.304371" lon="-16.566490">
        <ele>2062.2</ele>
        <time>2014-02-05T09:25:33Z</time>
      </trkpt>

      ...

    </trkseg>
  </trk>
</gpx>
```

Listado 4.1: Ejemplo de un fichero GPX

Cada instancia de la clase *GPXData* representa el contenido de un fichero GPX. Como puede observarse en el listado 4.2, se recogen los datos del fichero GPX y se calculan algunos derivados de ellos como son la elevación máxima y mínima, ganancia y pérdida de altitud y la distancia total del sendero.

4.2. Clase *GPXParser*

Esta clase es la encargada de extraer el contenido de un fichero GPX. El listado 4.3 muestra el método que realiza dicha función.

El parámetro *path* contiene la ruta en el sistema de ficheros del dispositivo Android, y con una instancia de *LectorFicheros* en las

```
1  public class GPXData implements Serializable{
2      private String nombreSendero;
3      private String descSendero;
4      private Punto[] puntos;
5      private double alturaMinima;
6      private double alturaMaxima;
7      private double gananciaAltitud;
8      private double perdidaAltitud;
9      private double distancia;
10
11     public GPXData(String nS, String dS, Punto[] p) {
12         setNombreSendero(nS);
13         setDescSendero(dS);
14         setPuntos(p);
15
16         ... // Calculo de altura minima y maxima, ganancia y perdida de gananciaAltitud
17
18         ... // Asignaciones de variables
19     }
20 }
```

Listado 4.2: Definición de la clase GPXData

líneas 2 y 3 se extrae el contenido del fichero en un *String*.

De dicho *String* escogemos el contenido de las etiquetas <`trk`> (línea 5), <`name`> (línea 6), <`desc`> (línea 7) y <`trkseg`>(línea 7), esta última contiene la definición de todos los puntos.

Utilizando el método ilustrado en el listado 4.4, se convierte el *String* que contiene todos los puntos en un array utilizando el método *split(regexp)* de Java [27] (línea 2).

4.3. Clase *DBHandler*

Cada uno de los senderos que muestra `PateosTF` está almacenado en una base de datos SQLite [37]. La cual es gestionada desde la clase *DBHandler*. Dicha base de datos contiene dos tablas tal como se muestra en el listado 4.5.

- Tabla SENDEROS: esta tabla almacena la información de un objeto *GPXData*. Además, se lleva un control de cuales de estos senderos son los descargados desde **Firebase** y cuales son los senderos del usuario. Del mismo modo, y sólo para las funciones de administrador, se puede controlar si un sendero añadido

```
public GPXData ParsearFichero(String path) {
    LectorFicheros lf = new LectorFicheros();
    lf.leerFichero(path);

    String sinCabecera = extraerContenido(lf.getFicheroLeido(), TRK_MARK, TRK_MARK_END);
    String nombreSendero = extraerContenido(sinCabecera,    NAME_MARK,  NAME_MARK_END);
    String descSendero = extraerContenido(sinCabecera,      DESC_MARK,  DESC_MARK_END);
    String ruta        = extraerContenido(sinCabecera,      TRKSEG_MARK, TRKSEG_MARK_END);
    String[] puntos    = trkptParser(ruta);

    Punto[] gpx_puntos = new Punto[puntos.length - 1];

    for (int i = 1; i < puntos.length; i++) {
        String latitud = extraerContenido(puntos[i], LAT_INI, LAT_FIN);
        String longitud = extraerContenido(puntos[i], LON_INI, LON_FIN);
        String elevacion = extraerContenido(puntos[i], ELE_MARK, ELE_MARK_END);
        String tiempo  = extraerContenido(puntos[i], TIME_MARK, TIME_MARK_END);
        Punto p = new Punto();
        p.setLatitud(Double.parseDouble(latitud));
        p.setLongitud(Double.parseDouble(longitud));
        p.setElevacion(Double.parseDouble(elevacion));
        p.setTiempo(tiempo);
        gpx_puntos[i - 1] = p;
    }
    return new GPXData(nombreSendero, descSendero, gpx_puntos);
}
```

Listado 4.3: Método que extrae el contenido de un fichero GPX

```
public String[] trkptParser(String cadena) {
    String[] puntos = cadena.split(TRKPT_MARK);
    for (int i = 0; i < puntos.length; i++) {
        String aux = cambiarCaracter(puntos[i], '"', '\'');
        puntos[i] = aux;
    }
    return puntos;
}
```

Listado 4.4: Método que extrae el contenido de la etiqueta <`trkseg`>

```
1  public void onCreate(SQLiteDatabase db) {
2      String CREATE_SENDEROS_TABLE = "CREATE TABLE " +
3             TABLE_SENDEROS + "("
4             + COLUMN_NOMBRE_SENDERO + " TEXT,"
5             + COLUMN_DESCRIPCION   + " TEXT,"
6             + COLUMN_LAT_INI       + " REAL,"
7             + COLUMN_LON_INI       + " REAL,"
8             + COLUMN_ALT_MIN       + " REAL,"
9             + COLUMN_ALT_MAX       + " REAL,"
10            + COLUMN_PER_ALT       + " REAL,"
11            + COLUMN_GAN_ALT       + " REAL,"
12            + COLUMN_DISTANCIA     + " REAL,"
13            + COLUMN_SUBIDO        + " TEXT,"
14            + COLUMN_NUBE          + " TEXT,"
15            + "PRIMARY KEY (" + COLUMN_NOMBRE_SENDERO + ", " + COLUMN_NUBE + ")) ";
16
17     String CREATE_PUNTOS_TABLE = "CREATE TABLE " +
18            TABLE_PUNTOS           + "("
19            + COLUMN_NOMBRE_SENDERO + " TEXT NOT NULL,"
20            + COLUMN_ORDEN         + " INTEGER NOT NULL,"
21            + COLUMN_LATITUD       + " REAL, "
22            + COLUMN_LONGITUD      + " REAL, "
23            + COLUMN_ELEVACION     + " REAL, "
24            + COLUMN_TIEMPO        + " TEXT, "
25            + COLUMN_DISTANCIA     + " REAL, "
26            + COLUMN_NUBE          + " TEXT, "
27            + "PRIMARY KEY (" + COLUMN_NOMBRE_SENDERO + ", " + COLUMN_ORDEN + ", " + COLUMN_NUBE +
                  ") "
28            + "FOREIGN KEY (" + COLUMN_NOMBRE_SENDERO + ") REFERENCES " + TABLE_SENDEROS + "(" +
                  COLUMN_NOMBRE_SENDERO +")"
29            + ")";
30
31     db.execSQL(CREATE_SENDEROS_TABLE);
32     db.execSQL(CREATE_PUNTOS_TABLE);
33 }
```

Listado 4.5: Constructor de la base de datos

manualmente está en **Firebase** o no.

- Tabla PUNTOS: esta tabla contiene los puntos correspondientes a los senderos de la tabla SENDEROS.

Se ha implementado un método de inserción de objetos *GPXData* lo más optimizado posible, ya que, como mínimo, los senderos se componen de 500 puntos. El principal problema es que la ejecución de un INSERT en Android conlleva un *commit*, lo que dilata el tiempo necesario para almacenar un sendero en la base de datos.

Por ello, como se muestra en el listado 4.6 las inserciones de los puntos se hacen por medio de transacciones (líneas 26 – 40).

```java
public void insertRapido(GPXData gpx) {
    SQLiteDatabase db = this.getWritableDatabase();

    String insert_send_ini = "INSERT INTO " + TABLE_SENDEROS + " VALUES (";

    String descripcion_prev = cambiarCaracter(gpx.getDescSendero(), '\'', '.');

    String descripcion = descripcion_prev;
    String insert_send = insert_send_ini
           + "'" + gpx.getNombreSendero()  + "', "
           + "'" + descripcion                + "', "
           + gpx.getPuntos()[0].getLatitud() + ", "
           + gpx.getPuntos()[0].getLongitud() + ", "
           + gpx.getAlturaMinima() + ", "
           + gpx.getAlturaMaxima() + ", "
           + gpx.getPerdidaAltitud() + ", "
           + gpx.getGananciaAltitud() + ", "
           + gpx.getDistancia() + ", "
           + "'" + "NO" + "', "
           + "'" + "NO" + "')";

    db.execSQL(insert_send);

    String insert_punt = "INSERT INTO " + TABLE_PUNTOS + " VALUES (?, ?, ?, ?, ?, ?, ?, ?);";
    SQLiteStatement statement = db.compileStatement(insert_punt);
    db.beginTransaction();
    for (int j = 0; j < gpx.getPuntos().length; j++) {
        statement.clearBindings();
        statement.bindString(1, gpx.getNombreSendero());
        statement.bindLong(2, j);
        statement.bindDouble(3, gpx.getPuntos()[j].getLatitud());
        statement.bindDouble(4, gpx.getPuntos()[j].getLongitud());
        statement.bindDouble(5, gpx.getPuntos()[j].getElevacion());
        statement.bindString(6, gpx.getPuntos()[j].getTiempo());
        statement.bindDouble(7, gpx.getPuntos()[j].getDistancia());
        statement.bindString(8, "NO");
        statement.execute();
    }
    db.setTransactionSuccessful();
    db.endTransaction();
}
```

Listado 4.6: Método para insertar *GPXData* de forma rápida

La ejecución de una transacción conlleva un *commit*, y en cada transacción se puede ejecutar un número indefinido de sentencias. Por tanto, se asocia la inserción de todos los puntos del sendero a una transacción y ésta es la que se ejecuta.

De esta forma se reduce el número de *commits* a uno, en lugar de uno por cada punto que se debe insertar. En conclusión, un método que no utiliza transacciones precisa alrededor de 7 segundos para insertar 1000 puntos, en cambio, el método optimizado no necesita más de un segundo para realizar la misma carga de trabajo.

4.4. Clase *FileParser*

Esta clase (listado 4.7) es la encargada de extraer de forma asíncrona el contenido de un fichero GPX. Se ha optado por esta solución debido a que el parseo de un fichero GPX y la inserción del contenido del mismo en la base de datos consume entre 5 y 20 segundos, dependiendo de la cantidad de puntos del sendero. Por lo que se debe bloquear la interfaz gráfica de forma que el usuario no vea una pantalla negra sino un *Dialog* con un mensaje que le indique lo que ocurre en cada momento.

La funcionalidad que permite al usuario añadir un nuevo sendero a la base de datos sólo está implementada en la pantalla con la lista de senderos y en el *MapsActivity*.

4.5. Pantalla principal (*MainActivity*)

Debido a que la base de datos en **Firebase** no es actualizada con frecuencia, se descarga su contenido en la base de datos local la primera vez que el usuario inicia `PateosTF`, de forma que se accede a ella en lugar de realizar su descarga y se evita así un consumo innecesario de la tarifa de datos.

Siguiendo la guía de **Firebase**: *"Java Android Guide: Saving Data"* [10] se ha implementado el método (listado 4.8) que nos permite

```
1  private class FileParser extends AsyncTask<String, Void, String> {
2
3      ... // Instanciacion del ProgressDialog, definicion de variables y metodo onPreExecute
4
5      @Override
6      protected String doInBackground(String... path) {
7          GPXData ruta = null;
8          String mensaje = "";
9          try
10         {
11             GPXParser gpx = new GPXParser(path[0]);
12             ruta = gpx.getDatos();
13             changeMessage("Actualizando la base de datos...");
14             DBHandler db = new DBHandler(context);
15             if (!db.isGuardado(ruta)) {
16                 db.insertRapido(ruta);
17                 done = true;
18                 mensaje = "La base de datos fue actualizada correctamente";
19             } else {
20                 mensaje = "Ya existe un sendero con ese nombre";
21             }
22         }
23         catch (Exception ex)
24         {
25            Log.e("Ficheros", "Error al leer fichero desde memoria interna \n" + ex.getMessage());
26            mensaje = "Error al leer el fichero desde memoria interna";
27         }
28         return mensaje;
29
30     }
31
32     public void changeMessage(final String mensaje) {
33         new Thread(new Runnable() {
34             @Override
35             public void run() {
36                 try {
37                     ringProgressDialog.setMessage(mensaje);
38                 } catch (Exception e) {
39                 }
40             }
41         }).start();
42     }
43
44     ... // Definicion del metodo onPostExecute que actualiza las estructuras de datos que
           contienen los senderos en la clase padre
45 }
```

Listado 4.7: Clase para parsear ficheros GPX

```
1 public void subirSendero(GPXData sendero) {
2     Firebase myfirebase = new Firebase("https://pateostf.firebaseio.com/");
3     Firebase postRef = myfirebase.child("rutas");
4     postRef.push().setValue(sendero);
5     DBHandler db = new DBHandler(this);
6     db.setSubido(sendero.getNombreSendero());
7 }
```

Listado 4.8: Método que sube un sendero a **Firebase**

almacenar senderos en la nube. Dicho método no está al alcance del usuario, siendo el desarrollador el único que podrá hacer uso de esta funcionalidad.

Del mismo modo, siguiendo la guía: *"Java Android Guide: Retrieving Data"* [9] se ha implementado el método que descarga los senderos almacenados en **Firebase** (listado 4.9) y los registra en la base de datos local. El proceso seguido para la descarga es el siguiente:

- Instanciación de la base de datos con la URL del proyecto en **Firebase** (Línea 2).
- Descarga de la información de la base de datos (Línea 10).
- Los objetos almacenados en la base de datos son instancias de la clase *GPXData*, por lo que la información descargada se corresponde con una lista de dichos objetos.

  Cada objeto contenido en la lista tiene asociados tantos pares *<Clave, Valor>* como atributos tiene la clase *GPXData*. El conjunto de claves está formado por los identificadores de los atributos, y los valores han de ser parseados al tipo de dato que les corresponde (líneas 13 – 24).

```
1  public void getDatosFirebase() {
2      Firebase myfirebase = new Firebase("https://pateostf.firebaseio.com/rutas");
3
4      ... // Instanciacion del ProgressDialog y declaracion de variables
5
6      myfirebase.addValueEventListener(new ValueEventListener() {
7        @Override
8        public void onDataChange(DataSnapshot snapshot) {
9        HashMap<String, Object> _rutas = new HashMap<String, Object>();
10       _rutas = (HashMap<String, Object>) snapshot.getValue();
11       for (Object key : _rutas.keySet() ) {
12           HashMap<String, Object> o = (HashMap<String, Object>) _rutas.get(key);
13           String nombre = (String)o.get("nombreSendero");
14           String descrp = (String)o.get("descSendero");
15           ArrayList<Object> puntos = (ArrayList<Object>)o.get("puntos");
16
17           Punto[] ps = new Punto[puntos.size()];
18           for (int i = 0; i < puntos.size(); i++) {
19           Punto p = new Punto();
20           HashMap<String, Object> aux = (HashMap<String, Object>)puntos.get(i);;
21           p.setLatitud((Double)aux.get("latitud"));
22           p.setLongitud((Double)aux.get("longitud"));
23           p.setElevacion((Double)aux.get("elevacion"));
24           p.setTiempo((String)aux.get("tiempo"));
25           ps[i] = p;
26           }
27           GPXData gpx = new GPXData(nombre, descrp, ps);
28           rutas.add(gpx);
29       }
30
31       DBHandler db = new DBHandler(contexto);
32       for (int i = 0; i < rutas.size(); i++) {
33           db.insertRapidoCloud(rutas.get(i));
34       }
35
36       _ringProgressDialog.dismiss();
37       }
38
39       ... //Definicion del metodo onCancelled()
40     });
41 }
```

Listado 4.9: Método que descarga los senderos de **Firebase**

4.6. Pantalla con el mapa (*MapsActivity*)

Este *Activity* es el encargado de mostrar un mapa con un marcador en el punto donde comience cada uno de los senderos de la base de datos. Por defecto, se centrará de forma que Tenerife ocupe la mayor parte de la pantalla, pero pueden mostrarse senderos localizados en cualquier punto.

Al crear un *MapsActivity* a través de Android Studio se autogeneran los métodos mostrados en los listados 4.10, 4.11, 4.12.

- *SetUpMapIfNeeded()*: es el método encargado de instanciar el mapa si es necesario. Ha sido modificado para que asigne el tipo de mapa: *TERRAIN* (Línea 9)
- *SetUpMap()*: instancia el controlador de los marcadores y la cámara, que es centrada utilizando la latitud y longitud de Tenerife (Línea 3).
- *onMarkerClick(Marker marker)*: dibuja el sendero asociado a un marcador la primera vez que se pulsa el mismo, si se vuelve a pulsar, muestra un *Dialog* con un resumen de su información. El sendero deja de ser mostrado cuando se pulse por tercera vez su marcador o en el momento que se seleccione cualquier otro.

4.7. Pantalla con la lista de senderos (*SwipeListViewActivity*)

Esta pantalla es la encargada de mostrar los senderos en una lista. Cada uno de los elementos de la lista está compuesto por una imagen, el nombre y la distancia de un sendero. Los elementos de la lista pueden ser desplazados tanto a la izquierda como a la derecha, de forma que, una vez desplazados, se revelan dos botones, los cuales

```
1  private void setUpMapIfNeeded() {
2      // Do a null check to confirm that we have not already instantiated the map.
3      if (mMap == null) {
4          // Try to obtain the map from the SupportMapFragment.
5          mMap = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map))
6                          .getMap();
7          if (mMap != null) {
8              mMap.setMyLocationEnabled(true);
9              mMap.setMapType(mMapTypes[indMapa]);
10             setUpMap();
11         }
12     }
13 }
```

Listado 4.10: SetUpMapIfNeeded() sobreescrito

```
1  private void setUpMap() {
2      mMap.setOnMarkerClickListener((GoogleMap.OnMarkerClickListener) this);
3      mCamera = CameraUpdateFactory.newLatLngZoom(tfLatLng, 9);
4      mMap.animateCamera(mCamera);
5  }
```

Listado 4.11: SetUpMap() sobreescrito

pueden dirigir al usuario a la pantalla que muestra el mapa o a la que muestra la información de un sendero.

La lista ha sido implementada por la empresa **47Degrees** [1], se ha configurado el proyecto tal como se indica en su repositorio de github [2] y se ha implementado el método (listado 4.13) de la siguiente forma:

- Cuando se pulsa sobre un sendero, se muestra la pantalla con la información de un sendero (Líneas 10–18).
- Se activa el modo deslizamiento (*Swipe*) de los *items* tanto a la izquierda como a la derecha, incluso cuando se hace *LongClick* sobre ellos (Líneas 24 – 26).
- A pesar de que la configuración de la lista puede ser modificada en la clase *SettingsManager*, no se le da la posibilidad al usuario de cambiar dicha configuración.

En esta pantalla también se muestra un conjunto de botones implementado por Jerzy Chalupski [28]. Para utilizarlos (listado 4.14) se

```java
@Override
public boolean onMarkerClick(Marker marker) {
    // TODO Auto-generated method stub
    for (int i = 0; i < Marcadores.size(); i++) {
        if (marker.equals(Marcadores.get(i).getMarcador())) {
            // Si no hay nada pintado o se hace click sobre otro
            // sendero (aunque haya uno pintado), se pinta el sendero
            if ((mapasinpintar) || (pintado != i)){
                dialogsinmostrar = true;
                pintarMarcador(i);
                show_dialog = true;
                mapasinpintar = false;
                ruta_showed = rutas.get(i);
                return true;
            }
            // Si se ha seleccionado un sendero que esta pintado se muestra
            // la informacion del sendero
            else if (dialogsinmostrar) {
                dialogsinmostrar = false;
                FragmentManager fragmentManager = getFragmentManager();
                InfoDialog dialogo = new InfoDialog();
                dialogo.setCancelable(true);
                dialogo.show(fragmentManager, "tagAlerta");
                return true;
            }
            // Sino se llama a pintarMarcador(i) que lo despinta, y se
            // inician de nuevo las estructuras de datos
            else {
                pintarMarcador(i);
                ruta_showed = null;
                mapasinpintar = true;
                dialogsinmostrar = true;
                return true;
            }

        }
    }
    return false;
}
```

Listado 4.12: onMarkerClick(Marker marker) sobreescrito

```
1  private void reload() {
2         swipeListView = (SwipeListView) findViewById(R.id.example_lv_list);
3
4         ... // Definicion de acciones dependiendo del SDK del dispositivo
5
6         swipeListView.setSwipeListViewListener(new BaseSwipeListViewListener() {
7
8             ... // Definicion en blanco de eventos no controlados en PateosTF
9
10            @Override
11            // Cuando se selecciona un item
12            public void onClickFrontView(int position) {
13                Toast.makeText(SwipeListViewActivity.this, "Cargando Info", Toast.LENGTH_SHORT).show
                      ();
14                Intent intent = new Intent(SwipeListViewActivity.this, SenderoViewer.class);
15                intent.putExtra("sendero", data.get(position));
16                startActivity(intent);
17                Log.d("swipe", String.format("onClickFrontView %d", position));
18            }
19
20        });
21
22        swipeListView.setAdapter(adapter);
23        SettingsManager settings = SettingsManager.getInstance();
24        swipeListView.setSwipeMode(settings.getSwipeMode());
25        swipeListView.setSwipeActionLeft(settings.getSwipeActionLeft());
26        swipeListView.setSwipeActionRight(settings.getSwipeActionRight());
27        swipeListView.setOffsetLeft(convertDpToPixel(settings.getSwipeOffsetLeft()));
28        swipeListView.setOffsetRight(convertDpToPixel(settings.getSwipeOffsetRight()));
29        swipeListView.setAnimationTime(settings.getSwipeAnimationTime());
30        swipeListView.setSwipeOpenOnLongPress(settings.isSwipeOpenOnLongPress());
31    }
```

Listado 4.13: Método encargado de gestionar eventos en la lista

```
<com.github.clans.fab.FloatingActionMenu
    android:id="@+id/menu_swipe"
    android:text="Menu"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_marginRight="10dp"
    android:layout_marginBottom="10dp"
    android:layout_marginLeft="10dp"
    android:layout_gravity="bottom|right"
    ... <!-- Definicion del resto de caracteristicas del menu -->
    fab:menu_labels_maxLines="-1">

    <com.github.clans.fab.FloatingActionButton
        android:id="@+id/mapa_swipe"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/map_icon_def_white"
        android:onClick="irMapa"
        fab:fab_size="mini"
        fab:fab_label="Mapa" />
... <!-- Definicion del resto de botones -->

</com.github.clans.fab.FloatingActionMenu>
```

Listado 4.14: Definición del menú y botones en la *SwipeListViewActivity*

han seguido los pasos ilustrados en la sección *Usage* del repositorio, de forma que finalmente permiten al usuario:

- Dirigirse al *MapsActivity*
- Mostrar los senderos descargados del servidor
- Mostrar los senderos del usuario
- Añadir un nuevo sendero

4.8. Pantalla con la información de un sendero (*TrailActivity*)

Este *Activity* está formado por tres *Fragments*. Desde cada uno de ellos se puede acceder a los dos restantes. Inicialmente, se podía

transitar de uno a otro, además de con los iconos correspondientes en la barra superior, deslizándolos hacia un lado (*sliding*), pero debido a que se necesita registrar el evento de deslizarse en el *Fragment* que muestra el mapa y en el que muestra la gráfica, se ha desactivado la posibilidad de transitar haciendo *sliding*.

4.8.1. *MapFragment*

Este *Fragment* se encarga de mostrar un mapa con el sendero dibujado de forma permanente. La principal diferencia entre esta pantalla y el *MapsActivity* es la definición del método *onMarkerClick(Marker marker)*.

En esta pantalla se hace la llamada al método por defecto, el cual añade a la interfaz dos botones encargados de dirigir al usuario a la aplicación Maps de su dispositivo, obteniendo de ella una ruta desde la localización actual del dispositivo hasta el inicio del sendero que se mostraba en el mapa. También permite al usuario utilizar el *StreetView* [22] en dicho punto.

4.8.2. *ChartFragment*

En esta pantalla se muestra una gráfica con la orografía de un sendero. Para ello se ha utilizado la librería externa AChartEngine.jar [3] siguiendo el tutorial *"ACHARTENGINE"* [38].

Debido a que cada sendero está compuesto por más de 500 puntos, la visión de la gráfica cuando el dispositivo está en vertical es demasiado compacta, por ello, se ha optado por mostrar sólo los primeros 5 kilómetros de un sendero cuando el dispositivo se encuentre en esta orientación, indicando al usuario con un mensaje que puede deslizarse por la gráfica para verla de forma completa.

Capítulo 5

Despliegue de la aplicación

Debido a que el uso de **Firebase** implica el pago de una cuota mensual de 49$, `PateosTF` se ha puesto en producción eliminando toda funcionalidad asociada a su uso de forma que, podrá ser utilizada únicamente para añadir senderos manualmente. En la base de datos se han almacenado por defecto tres senderos que podrán ser visualizados por todos los usuarios de la aplicación.

Para publicar la aplicación en *GooglePlay* [21] se han seguido los pasos del post: *"Subir y distribuir aplicaciones"* [39]. Para realizarlos es necesario generar un fichero APK [41], se recomienda seguir el video tutorial: *"Android Studio - Como exportar APK"*.

Para el registro de la aplicación se solicita un vídeo ilustrativo del funcionamiento de la aplicación, el cual fue elaborado y publicado en Youtube [26]

Finalmente, `PateosTF` puede ser descargada en *GooglePlay* [25].

Capítulo 6

Conclusiones y líneas futuras de trabajo

Al concluir el TFG, se han adquirido los conocimientos mínimos para desarrollar aplicaciones para dispositivos Android. Hemos diseñado, desarrollado y desplegado una aplicación que es capaz de gestionar senderos, dibujarlos sobre un mapa, acceder a una base de datos remota así como al sistema de ficheros del dispositivo para añadir senderos manualmente.

Utilizando librerías externas, es capaz de crear una gráfica que represente la orografía de un sendero así como de añadir elementos a la interfaz de usuario, lo que implica una mejora en la apariencia y usabilidad de la aplicación.

La gestión del tiempo ha sido clave en el desarrollo, ya que se ha compaginado con la asignatura de *Prácticas Externas* de cuarto curso del Grado en Ingeniería Informática. Gracias a ello, se ha obtenido experiencia en la gestión de tareas relacionadas con la Ingeniería del Software.

También se ha trabajado en la elaboración de una memoría técnica con *LaTeX*, desconocido hasta el momento.

El desarrollo de `PateosTF` continuará implementando las siguientes funcionalidades:

- Visualizar la previsión meteorológica de un sendero. Para ello

se utilizará la API gratuita de OpenWeatherMap [31]

- Clasificar los senderos por dificultad en relación a distancia y orografía.
- Dar al usuario la posibilidad de crear un sendero utilizando el GPS de su dispositivo Android.
- Ser capaz de ofrecer un seguimiento mientras se recorre un sendero a la vez que se registra el tiempo y la distancia.
- Dar la posibilidad de hacer fotos y asociarlas a puntos de un sendero.
- Imitar a Wikiloc como servicio donde descargar y compartir senderos con los demás usuarios de `PateosTF`.

Capítulo 7

Summary and Conlusions

At the conclusion of the work, I have acquired the minimum knowledge needed to develop applications for Android devices. I have designed, developed and desployed an application that is able to manage trails, represent them on a map, access a remote database and access the device file system to add trails manually.

Using external libraries, `PateosTF` creates a chart representing the topography of a trail. I have learned how to add items to the user interface, which means an improvement in the appearance and usability of the application.

Time management has been very important in this project, because it has combined with the subject "Prácticas Externas" also studied in the fourth year of the Degree in Computing Engineering. As a result, I have improved my skills in managing tasks related to software engineering.

I have also learned to draw up a Technical Report using *LaTeX*, unknown so far.

The development of `PateosTF` could be continued by implementing some of the following features:

- Watch the weather.

- Set the difficulty of a trail related to the distance and the topography.
- Record a trail using the the Android device internal GPS.
- Being able to track a trail and record the time and distance.
- Take photos and relate them to a trail.
- Imitate Wikiloc as a downloading and sharing service.

Capítulo 8

Presupuesto

La puesta en producción de `PateosTF` implica el pago de una cuota de mantenimiento en **Firebase**. Dicha cuota es de 49$ al mes y proporciona los siguientes servicios:

- 200 conexiones simultáneas a la base de datos
- 20 GB de transferencia de datos
- 10 GB de almacenamiento
- Escalado automático

Además, para publicar la aplicación en *GooglePlay* se debe abonar la cuota de registro de 25$.

En conclusión, se precisa un desembolso inicial de 74$ y el pago de los 49$ mensualmente en concepto de la cuota de **Firebase**

Bibliografía

[1] 47deg. 47Degrees, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 31

[2] 47deg. SwipeList, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 31

[3] 4ViewSoft. AChartEngine, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 35

[4] Bitbucket. Bitbucket official page, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 2

[5] Eclipse Foundation. Eclipse, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 4

[6] Everytrail. Everytrail official page, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 9

[7] Firebase. Android QuickStart Firebase, 2011. [Disponible electrónicamente; Último acceso junio de 2015]. 6

[8] Firebase. Firebase, 2011. [Disponible electrónicamente; Último acceso junio de 2015]. 6

[9] Firebase. Java Android Guide: Retrieving data, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 28

[10] Firebase. Save Data, Firebase, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 26

[11] Google. Descarga de AndroidStudio, 2014. [Disponible electrónicamente; Último acceso junio de 2015]. 4, 5

[12] Google. Instalación del SDK de Android, 2014. [Disponible electrónicamente; Último acceso junio de 2015]. 5

[13] Google. Adding SDK Packages, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 5

[14] Google. Building Apps with Connectivity & the Cloud, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 5

[15] Google. Building Apps with content sharing, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 5

[16] Google. Building Apps with Graphics & Animation, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 5

[17] Google. Building Apps with Multimedia, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 5

[18] Google. Data Storage, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 5

[19] Google. Getting Started, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 5

[20] Google. GoogleMaps API, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 5

[21] Google. GooglePlay, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 36

[22] Google. StreetView, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 17, 35

[23] Google Inc. Maps, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 7, 17

[24] Gpsies. Gpsies official page, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 9

[25] Iván González Suárez. PateosTF - GooglePlay, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 36

[26] Iván González Suárez. PateosTF - Trabajo de Fin de Grado, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 36

[27] Java-Spain. Método Split Java, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 22

[28] Jerzy Chalupski. Floating Buttons, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 31

[29] LaTeX. LaTeX - A document preparation system, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 3

[30] Mapmytracks. Mapmytracks official page, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 9

[31] OpenWeatherMap. OpenWeatherMap API, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 38

[32] Oruxmaps. Oruxmaps, 2009. [Disponible electrónicamente; Último acceso junio de 2015]. 9

[33] OscarVila2. Cómo instalar Oracle Java en Ubuntu, 2014. [Disponible electrónicamente; Último acceso junio de 2015]. 5

[34] Pablo Bascuñana Saiz . Creando una Google Maps Activity con Android Studio, 2014. [Disponible electrónicamente; Último acceso junio de 2015]. 6

[35] Pablo Bascuñana Saiz . Jugando con GoogleMaps en Android, 2014. [Disponible electrónicamente; Último acceso junio de 2015]. 6

[36] Pablo Bascuñana Saiz. Pablo Bascuñana Saiz, 2014. [Disponible electrónicamente; Último acceso junio de 2015]. 5

[37] SGoliver. SQLite Android, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 22

[38] skholingua. AChartEngine, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 35

[39] Support Google. Subir y distribuir aplicaciones, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 36

[40] Wikiloc. Wikiloc, 2006. [Disponible electrónicamente; Último acceso junio de 2015]. 7

[41] Wikipedia. APK (Format), 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 36

[42] Wikipedia. Backend as a service, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 6

[43] Wikipedia. Java Development Kit, 2015. [Disponible electrónicamente; Último acceso junio de 2015]. 5