ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Desarrollo de una aplicación de cálculo de perfiles radioeléctricos para dispositivos móviles con sistema operativo Android

Proyecto fin de Carrera

Autor: Alonso Javier Sánchez García

Director: Leandro Juan Llácer

Septiembre 2011

Autor	Alonso Javier Sánchez García
E-mail del Autor	Alonsoj22@gmail.com
Director(es)	Leandro Juan Llácer
E-mail del Director	leandro.juan@upct.es
Codirector(es)	
Título del PFC	Desarrollo de una Aplicación de Cálculo de Perfiles radioeléctricos para dispositivos móviles con sistema operativo Android.
Descriptores	

Resumen

En la planificación de un radioenlace juega un papel fundamental la utilización de herramientas de cálculo de perfiles radioeléctricos. El perfil radioeléctrico viene fijado básicamente por la ubicación del transmisor y el receptor, la frecuencia del sistema de radiocomunicaciones y el modelo digital del terreno (MDT), que contenga las alturas del terreno entre el transmisor y el receptor.

Se ha diseñado una aplicación Android, capaz de calcular perfiles radioeléctricos y la primera zona de fresnel, crear y editar emplazamientos, así como ayudar a la orientación de las antenas.

Titulación	Ingeniero Técnico de Telecomunicaciones, especialidad Telemática
Intensificación	
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Septiembre - 2011

ÍNDICE

Índice de figuras **6**

1. Introducción **8**

1.1. Objetivos del Proyecto 9

1.2. Contenido de la Memoria 10

2. Sistema operativo Android **11**

2.1. ¿Qué es Android? 12

2.2. Historia sobre android 13

2.3. Arquitectura 14

2.4. Componentes de una aplicación android 17

2.4.1. Activity [5] 17

2.4.2. Service [6] 18

2.4.3. Intent e Intent Filters [7] 19

2.4.4. Content Providers [8] 19

2.5. Ciclo de Vida de las aplicaciones 20

2.6. Ciclo de vida de una Activity 21

2.6.1. Inicio y Finalización de una Activity 24

2.7. Almacenamiento de datos [9] 25

2.7.1. Shared Preferences 25

2.7.2. Almacenamiento Interno 27

2.7.3. Almacenamiento Externo 28

2.7.4. Bases de Datos SQLite 29

2.7.5. Conexión de red 29
2.8. Seguridad y permisos [10] 30
2.8.1. Permisos de Usuario 30
2.8.2. Firmado de aplicaciones 31
2.9. Interfaz de Usuario [11] 32
2.9.1. Layouts 32
2.9.2. Views y ViewGroups 33
2.10. Andriod Manifest [12] 35
3. Aspectos Teóricos 37
3.1. Cálculo de la zona de fresnel 38
3.2. Curvatura de la tierra 40
4. RF Terrain Profile 42
4.1. Descripción general 43
4.2. Menú Principal 44
4.3. Emplazamientos 47
4.3.1. Creación de Emplazamientos 47
4.3.2. Edición de emplazamientos 51
4.3.3. Borrado de Emplazamientos 52
4.4. Perfiles 53
4.4.1. Crear nuevo perfil 53
4.4.2. Abrir Perfil 60
4.4.3. Borrar Perfil 61
4.5. Orientación 62
4.6. Traducción de la Aplicación 65
4.7. Otros Elementos Utilizados 66

4.7.1. AChartEngine [13] 66

4.7.2. Google Elevation API [14] 67

5. Conclusiones 69

5.1. Conclusions 70

6. Bibliografía 71

ÍNDICE DE FIGURAS

Figura 1. Logo Android 12

Figura 2. Arquitectura Android[4] 14

Figura 3 - Ciclo de vida de una Activity 23

Figura 4 - Esquema herencia View 32

Figura 5 - Ejemplo Gallery 33

Figura 6 - Ejemplo GridView 34

Figura 7 - Ejemplo ListView 34

Figura 8 - Representación de la zona de fresnel 39

Figura 9 - VARIACIÓN DE LA ZONA DE FRESNEL 40

Figura 10 - Icono de la aplicación RF Terrain Profiles 43

Figura 11 - Pantalla inicial 44

Figura 12 - Menú Emplazamientos 47

Figura 13 - Crear Emplazamiento 48

Figura 14 - Obtención de coordenadas por GPS 49

Figura 15 - Obtención de coordenadas a través del mapa 49

Figura 16 - LISTA DE EMPLAZAMIENTOS CREADOS 51

Figura 17 - Borrado de un emplazamiento 52

Figura 18 - Menú perfiles 53

Figura 19 - Emplazamiento perfil 54

Figura 20 - Datos Perfil 55

Figura 21 - Selección de unidades de frecuencia 56

Figura 22 - Representación del perfil en el mapa 57

Figura 23 - *Perfil Radioeléctrico* 57

Figura 24 - Lista de Perfiles Creados 60

Figura 25 - Borrado de Perfiles 61

Figura 26 - Menú Orientación 62

Figura 27 - Seleccionar dirección 62

Figura 28 - Brújula 63

Figura 29 - Diagrama azimut 64

Figura 30 - Diagrama elevación 64

Figura 31 - Ejemplos AChartEngine 66

1. INTRODUCCIÓN

1.1. OBJETIVOS DEL PROYECTO

El Auge del desarrollo de aplicaciones para dispositivos móviles ha supuesto la apertura de un gran abanico de posibilidades, pudiendo desde realizar las básicas acciones de llamar o recibir mensajes, hasta navegar por internet o ejecutar aplicaciones que nosotros mismos desarrollemos. En particular el sistema operativo Android, gracias a su carácter libre y de código abierto, nos permite realizar aplicaciones para uso propio o comercial.

Con la realización de este proyecto se pretende proporcionar una herramienta rápida y fiable para el cálculo de perfiles radioeléctricos. Gracias a su desarrollo sobre un sistema operativo para móviles como es Android, proporciona al usuario una movilidad ideal para este tipo de cálculos, característica no disponible en otras herramientas diseñadas para tal mismo fin.

La aplicación a desarrollar debe permitir la creación y edición de emplazamientos, los cuales servirán como puntos de origen y destino de los perfiles que se desee crear. Por último, se incluye una herramienta para la orientación de las antenas de un perfil. Gracias a esta funcionalidad de la aplicación se simplifica mucho el procedimiento de orientación de las antenas, ya que nos proporciona información del azimut y la elevación de un emplazamiento con respecto al otro.

1.2. CONTENIDO DE LA MEMORIA

Este apartado está destinado a ofrecer una visión global de cada uno de los apartados en los que se encuentra dividida la memoria. Este documente se compone de cinco capítulos.

El capítulo 1 se titula “Introducción” y pretende dar una visión global del proyecto, así como de proporcionar un pequeño resumen del resto de capítulos.

En el capítulo 2, “Sistema operativo Android”, se describe de forma muy resumida el sistema operativo Android. Además, se dan algunas nociones básicas sobre programación en Android, necesarias para la realización del proyecto.

El capítulo 3, “Aspectos Teóricos”, servirá para detallar los elementos teóricos que se han tenido en cuenta a la hora de la programación de la aplicación.

El siguiente capítulo, el 4, lleva por título el nombre de la aplicación, “RF Terrain Profiles”. En este capítulo se detallará el funcionamiento de aplicación y de todos los elementos utilizados por ésta.

El capítulo 5, “Conclusión”, pretende dar una valoración global del proyecto realizado, además, se exponen algunas ideas para la futura mejora de la aplicación.

El último capítulo se titula “Bibliografía”, en él se listan todas las fuentes en las que se ha basado el proyecto.

2. SISTEMA OPERATIVO ANDROID

2.1. ¿QUÉ ES ANDROID?

Android es el Sistema Operativo lanzado por Google para dispositivos móviles, que está basado en una modificación del kernel de Linux. Inicialmente este S. O. fue desarrollado por Google al cuál se unió más tarde la *Open Handset Aliance [1]*, formada por 78 compañías de la talla de Motorola, Samsung, LG, HTC e incluso el propio Google entre otros muchos.

A día de hoy, la plataforma Android está distribuida bajo una licencia conocida como *Apache Software License (ASL) [2]*, la cual permite a las operadoras añadir funcionalidades y modificaciones al S. 0. Está modificación no se restringe únicamente a las operadoras de telefonía, sino que cualquier usuario es libre de acceder y/o modificar el código fuente.

Además Google provee del SDK (Software Development Kit) de manera totalmente gratuita, que nos proporciona un plugin para la integración con Eclipse en el que se incluyen todas las API's necesarias para el desarrollo de software, y un emulador integrado.

FIGURA 1. LOGO ANDROID

2.2. HISTORIA SOBRE ANDROID

En 2003, un ingeniero de telecomunicaciones llamado Andy Rubin crea una empresa llamada Android Inc. una empresa dedicada el desarrollo de aplicaciones móviles. Dos años más tarde, en Agosto de 2005, Google adquirió la empresa y Andy Rubin pasó a formar parte del equipo de Google como vicepresidente de ingeniería, cargo que actualmente ocupa.

El 5 de Noviembre de 2007 cuando se hizo oficial el anuncio de Android como Sistema Operativo para móviles, aunque no sería hasta un año después cuando saldría el primer terminal funcionando con la versión Android 1.0, llamado Google Dream o G1. Aunque en esos momentos Android no fuera competidor directo con el que entonces dominaba el mercado, el Iphone de Apple, serviría de base para lo que hoy es uno de los Sistemas Operativos más potentes del mercado.

A partir de este momento empieza la evolución meteórica de Android con las actualizaciones del Sistema Operativo, llegando en la actualidad a poseer casi el 50% de los smartphones [3].

2.3. ARQUITECTURA

En este apartado se darán unas nociones básicas sobre programación en Android, así como de la arquitectura sobre la que se basa el Sistema Operativo y la mayoría de sus componentes.

La siguiente imagen es una representación de la arquitectura de cualquier dispositivo Android. Como se puede observar, se encuentra dividida en capas, lo que permite a cada capa obtener servicios de la capa inmediatamente inferior, y a su vez, proporcionar los servicios necesarios a las capas de nivel superior.

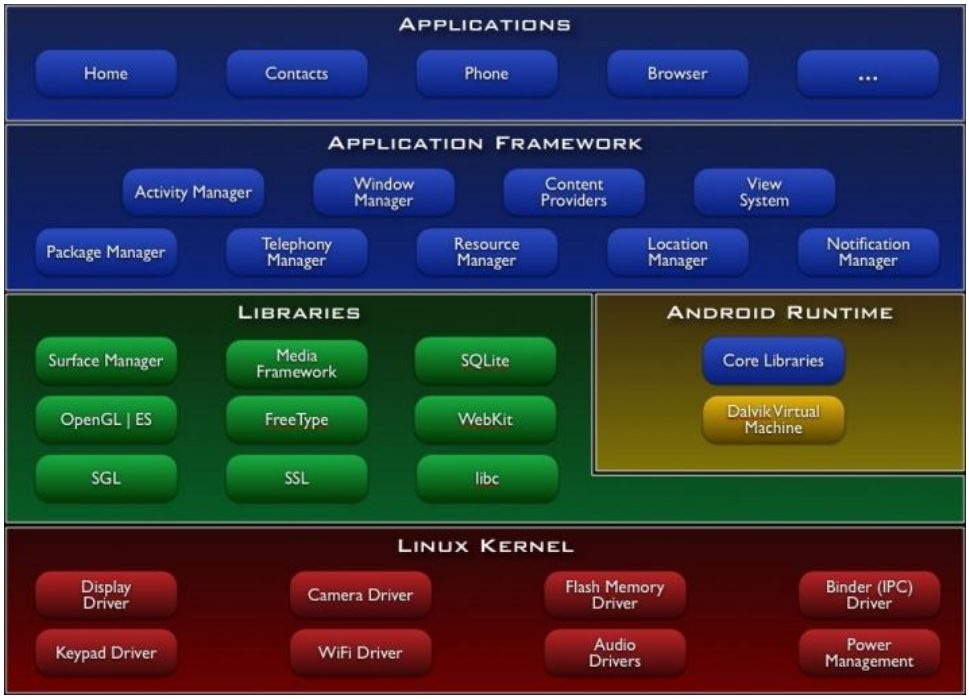


FIGURA 2. ARQUITECTURA ANDROID[4]

A continuación se detallará cada una de las capas, empezando por las capas de nivel inferior hasta la capa que interacciona directamente con el usuario:

- **Linux Kernel:** Esta es la capa que se encuentra más cercana al hardware del dispositivo y por lo tanto interacciona directamente con él. Se trata de una modificación de la versión 2.6 del kernel de Linux y provee al dispositivo de funciones de red, gestión de procesos, seguridad, gestión de memoria entre otros.

- **Android Runtime:** Esta capa está situada en paralelo con la capa de "librerías" ambas capas proporcionan funciones a las capas de nivel superior, en particular esta capa se encarga de proporcionar la mayoría de funcionalidades disponibles en el lenguaje de programación Java. Cada aplicación que se ejecuta en nuestro dispositivo corre en un proceso distinto de la Dalvik Virtual Machine.

- **Librerías:** Android incluye un conjunto de librerías escritas en C/C++ a las que se puede acceder a través del Android Application Framework. Estás librerías son específicas para Android. Algunas de ellas son:

  - System C library: Librería standard de C (libc) optimizada para usarla en dispositivos móviles.
  - Media Libraries: Librería que proporciona funciones para la reproducción y grabación de audio y vídeo en diferentes formatos entre ellos: MPEG4, H.264, MP3, AAC, AMR, JPG, PNG.
  - Surface Manager: Gestiona el acceso al sistema de visualización y realiza la composición de capas 2D y 3D de diversas aplicaciones.
  - LibWebCore: Un motor de navegación web para potenciar el navegador web o vistas webs incluidas en aplicaciones.
  - SGL: Motor para gráficos 2D.
  - 3D Libraries: Una implementación de las APIs de OpenGL ES 1.0, la cual permite el uso de aceleración 3D por hardware y la optimización para 3D por software.
  - FreeType: Renderizado de mapas de bits y fuentes de texto.
  - SQLite: Proporciona un motor de bases de datos relacional potente y ligero, disponible para todas las aplicaciones.

- **Application Framework:** Los desarrolladores tienen pleno acceso a las APIs utilizadas por las aplicaciones básicas. La arquitectura de las

aplicaciones está diseñada para favorecer la reutilización de código. Por ejemplo, una aplicación puede publicar ciertos datos que pueden ser utilizados por cualquier otra aplicación.

- **Applications:** Conjunto de aplicaciones preinstaladas en Android y escritas en Java, entre las que se encuentran un cliente de correo electrónico, un gestor de SMS, un navegador web, un calendario, Google Maps y un gestor de contactos.

2.4. COMPONENTES DE UNA APLICACIÓN ANDROID

Una aplicación Android está formada por, como mínimo, uno de estos componentes, pudiendo utilizar cada bloque de los que se describen a continuación libremente:

2.4.1. Activity [5]

Una Activity es el componente principal y más usado en las aplicaciones Android, ya que permite al usuario interactuar directamente con el dispositivo, normalmente cada Activity tiene asociada una única View (Vista), que posteriormente explicaremos.

Por lo general, una aplicación Android está formada por varias de estas Activities, siendo una de ellas la principal y la que se muestra al usuario cuando se arranca la aplicación.

El concepto de Activity va estrechamente ligado al de Intent, ya que gracias a este último es posible el paso de una Activity a otra. Este concepto se detallará con más detalle en la sección Ciclo de vida y Otros componentes.

Las Activities tienen que estar declaradas expresamente en el archivo Android Manifiest de la siguiente manera:

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```

2.4.2. Service [6]

Un Service es un componente de la aplicación que realiza operaciones de larga duración, se ejecuta en Background y no proporciona una interfaz al usuario, por lo que este no tiene conocimiento de su ejecución. Además un Service puede ejecutarse en paralelo con una Activity, o incluso si el usuario cambia de aplicación, éste sigue ejecutándose en segundo plano. Básicamente un Service tiene dos formas de ejecución:

- **Started:** Se conoce con este nombre a los Services que son iniciados por otros componentes de la aplicación -por ejemplo una Activity, mediante el método StartService()-, y continúan ejecutándose aunque el componente que lo creó se destruya. Este tipo de Services se inician para realizar un objetivo, como puede ser descarga de un archivo, y después de realizar la acción deberían destruirse. Por lo general, no devuelven una respuesta.

- **Bound:** Estos Services son creados mediante el método bindService(), y ofrecen al componente que lo creó una interfaz tipo cliente-servidor que permite a los distintos componentes interactuar con el Service. Un Service Bound puede estar ligado a varios componentes de la aplicación, pero a diferencia del tipo "Started", éste se destruye si deja de tener "clientes" con los que comunicarse.

Al igual que una Activity, los Services tienen que ser declarados en el archivo Android Manifest.

```
<manifest ... >
  ...
  <application ... >
      <service android:name=".ExampleService" />
      ...
  </application>
</manifest>
```

El atributo "name" es el único obligatorio, se corresponde con el nombre del Service.

2.4.3. Intent e Intent Filters [7]

Un Intent es, básicamente, el encargado de iniciar tanto una Activity como un Service, y estos, pueden pertenecer o no a la misma aplicación desde la que se inicia. Se dispone de métodos diferentes para iniciar cada uno de los componentes. Por ejemplo, las Activities se inician mediante el método startActivity() o startActivityForResult(), mientras que los Services deben iniciarse con startService() o bindService() dependiendo del tipo de servicio que se quiera iniciar.

Existen Intents predefinidos para las acciones que tienen un mayor uso, como pueden ser: llamar por teléfono, enviar un correo electrónico, abrir el navegador, etc.

Al intentar iniciar un componente, ya sea Activity o Service, el sistema operativo intenta buscar la aplicación adecuada para tal fin, y en el caso de existir más de una, nos muestra las opciones para escoger la que creamos conveniente.

2.4.4. Content Providers [8]

Este componente de la arquitectura Android favorece el intercambio de información entre aplicaciones, actuando como una zona de almacenamiento común a la que pueden acceder todas las aplicaciones. Algunos de los Providers definidos no pueden aportar datos como audio, video, imágenes, información de contacto personal, etc. Además de estos, un desarrollador puede crear un Provider para publicar los datos de sus aplicaciones y que otras puedan utilizarlos.

2.5. CICLO DE VIDA DE LAS APLICACIONES

En Android cada vez que es necesario ejecutar una aplicación, ésta lo hace corriendo en un proceso independiente en la Dalvik VM. Este proceso seguirá vivo hasta que la zona de memoria ocupada por él necesite ser liberada. El tiempo de vida de una aplicación no es controlado por la propia aplicación, sino que es el sistema operativo el que destruye la aplicación.

Para determinar el orden en el que se deberían eliminar los procesos en caso de baja memoria, Android establece unas prioridades en cuanto a importancia de los procesos que se encuentran corriendo en el dispositivo. La jerarquía de importancia es la siguiente, ordenada de mayor a menor importancia:

- **Foreground process:** Es un proceso con el que el usuario está interactuando y hospeda una Activity y, por lo tanto, se encuentra en primer plano, son los procesos que menos probabilidad tienen de ser eliminados, únicamente se destruirán en el caso de que la memoria sea críticamente baja.

- **Visible process:** Procesos que hospedan una Activity, se encuentra visible en pantalla, pero su método onPause() ha sido llamado. Esto suele ocurrir cuando en una Activity se crea un cuadro de diálogo con el que el usuario debe interactuar.

- **Service Process:** Al contrario que los dos anteriores, este proceso hospeda un Service iniciado con el método startService(). Estos procesos son destruidos únicamente en el caso de que el estado de la memoria comprometa el estado de los "Visible process", ya que pueden estar realizando tareas importantes para el usuario.

- **Background process:** Es un proceso que hospeda una Activity, pero ésta no es visible actualmente; eliminarlo no supone una gran repercusión para el usuario, Por normal general, existen muchos de estos procesos corriendo, y el sistema mantiene todos estos procesos en una lista de tipo FIFO, para asegurarse de que el último proceso visto por el usuario es el último en ser eliminado.

- **Empty process:** Es un proceso que no hospeda ningún componente de la aplicación. En ocasiones suelen mantenerse activos para

mejorar el tiempo de inicio de otro componente de la aplicación, pero por lo general, son eliminados con frecuencia.

2.6. CICLO DE VIDA DE UNA ACTIVITY

El ciclo de vida del componente Activity es muy parecido al de una aplicación Android. Se dispone de una pila de actividades llamada "stack de Activities" donde se van apilando las Activities por orden de creación, quedando las más recientes en la parte superior. En caso de falta de memoria, el sistema operativo comenzará eliminando las Activities que ocupan un lugar más bajo en la pila de actividades. Una Activity puede encontrarse en uno de estos tres estados:

- **Resumed o Running:** Una Activity se encuentra en este estado cuando está en la parte superior de la pila e interactuando directamente con el usuario,

- **Paused:** Otra Activity ha ocupado la parte superior de la pila de actividades, ésta ha quedado relegada a un segundo plano, pero aún es visible por el usuario aunque no pueda interactuar con ella.

- **Stopped:** Se produce la misma situación que en el estado "Paused", pero con la diferencia de que en este caso la Activity queda totalmente tapada por la que se ha iniciado.

Al igual que ocurre con las aplicaciones, el sistema operativo intenta favorecer las acciones realizadas por el usuario, siendo tanto las Activities como las aplicaciones que interactúan con él las últimas en ser destruidas.

La siguiente tabla resume los métodos que son llamados en el ciclo de vida de la Activity:

Método	Descripción	Siguiente
onCreate( )	Este método es llamado cuando se crea la Activity por primera vez, es donde se debe inicializar la Activity (crear vistas, recuperar datos guardados).	onStart( )
onRestar( )	Llamado cuando se ha parado, justo antes de que se inicie de nuevo.	onStart( )
onStart( )	Se ejecuta cuando la aplicación se vuelve visible al usuario.	onResume( ) o onStop( )
onResume( )	Llamado justo antes de que la aplicación interactúe con el usuario, la Activity se encuentra en estado Resume.	onPause( )
onPause( )	Este método se ejecuta justo antes de iniciar otra actividad, debe ejecutarse rápidamente, ya que hasta que no termine no se iniciará la siguiente.	onResume( ) o onStop( )
onStop( )	Se ejecuta cuando la Activity pasa a segundo plano y deja de ser visible al usuario.	onRestar( ) o onDestroy( )
onDestroy( )	Llamado justo antes de la destrucción de la Activity, será el último método que ejecute la Activity.	Ninguno

La siguiente imagen representa un diagrama de flujo del ciclo de vida de las aplicaciones. En ella podemos observar los métodos que son llamados al pasar por un cierto estado:

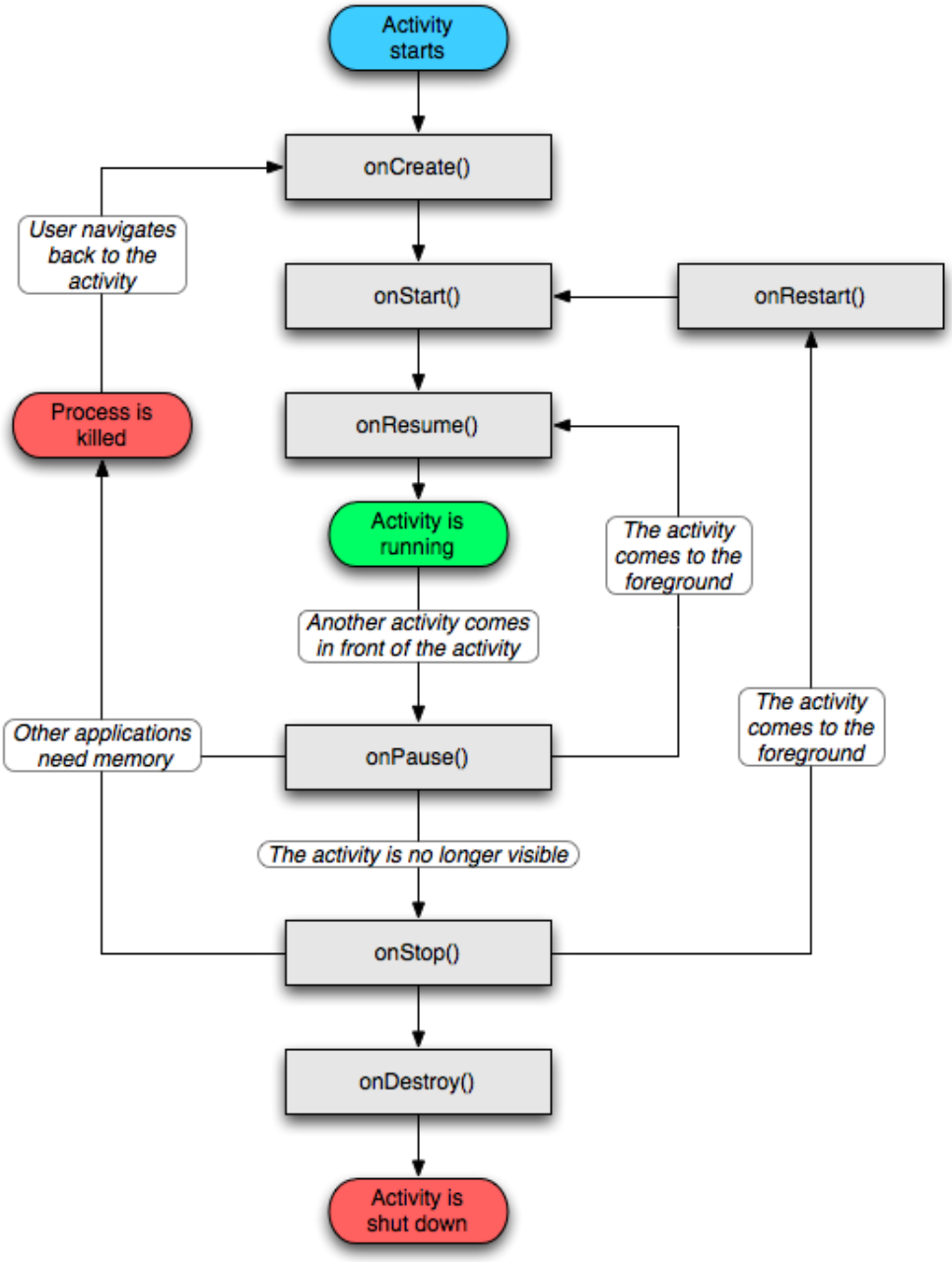


FIGURA 3 - CICLO DE VIDA DE UNA ACTIVITY

2.6.1. Inicio y Finalización de una Activity

Las Activities se inician mediante el método startActivity( ), al que se le debe pasar un objeto Intent, el cual describe la Activity que deseamos iniciar o la acción que pretendemos realizar (llamar por teléfono, enviar sms, abrir una página web…). En este último caso el sistema operativo escogerá la aplicación adecuada de entre las instaladas en el teléfono.

Para lanzar una Activity cuyo nombre es conocido, únicamente se necesita el Intent, al que se le pasa el nombre de la Activity que queremos ejecutar y la que lo lanza.

```
Intent intent = new Intent(this, Activity2.class);
startActivity(intent);
```

En este ejemplo se intenta lanzar una Activity llamada “Activity2” desde otra que se pasa como primer argumento al crear el Intent. Como comentábamos anteriormente es posible que no se quiera iniciar una Activity, sino realizar una acción concreta. Para dicha acción el sistema operativo buscará la aplicación que lo puede llevar a cabo, y en el caso de existir más de una, dejará libertad al usuario para escoger la que considere oportuna. En el siguiente ejemplo se pretende enviar un email:

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, recipientArray);
```

El parámetro Intent.ACTION_SEND indica que se desea enviar un email. En el String recipientArray se deben incluir los datos del email, destinatario, asunto, etc.

En cuanto a la finalización de una Activity es una tarea más sencilla, basta con llamar al método finish( ) para cerrar la Activity desde la que se invocó. También es posible cerrar una Activity, que no es la que se está ejecutando actualmente, con el método finishActivity( ).

2.7. ALMACENAMIENTO DE DATOS [9]

Android provee de varios tipos de almacenamiento, cada uno con unas características diferentes. La elección de un tipo u otro dependerá de varios factores como pueden ser la exclusividad o no de los datos para nuestra aplicación, la cantidad de datos a almacenar, etc.

2.7.1. Shared Preferences

La clase SharedPreferences provee de un framework que ~~nos~~ permite almacenar y recuperar la información. Los datos son almacenados en pares nombre-valor, este último puede ser cualquier tipo de datos primitivo: boolean, floats, int, longs o String. Este tipo de almacenamiento suele utilizarse para el almacenamiento de preferencias de usuario.

Para obtener un objeto de la clase SharedPreferences, se pueden utilizar uno de estos dos métodos:

- **getSharedPreferences( ):** Se utiliza este método cuando necesitamos utilizar varios ficheros de preferencias. El nombre del fichero debe ir incluido en el primer parámetro.

- **getPreferences( ):** Se utiliza este método si sólo es necesario un archivo de preferencias para nuestra aplicación. No es necesario pasarle un nombre como parámetro.

Para almacenar los datos se deben seguir los siguientes pasos:

1. Llamar al método edit( ) para obtener un objeto SharedPreferences.Editor.

2. Añadir los valores deseados con los métodos putBoolean( ), putString( ), putInt( ), etc.
3. Establecer los nuevos valores con el métodos commit( ).

Para la lectura de los datos almacenados, se deben utilizar los métodos get (getString( ), getInt( ), getBoolean( ), etc.) de la clase SharedPrefereneces.

A continuación se muestra un ejemplo del funcionamiento de SharedPreferences.

```
public class Calc extends Activity {
    public static final String PREFS_NAME = "MyPrefsFile";

    @Override
    protected void onCreate(Bundle state){
        super.onCreate(state);
        . . .

        // Restore preferences
        SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
        boolean silent = settings.getBoolean("silentMode", false);
        setSilent(silent);
    }

    @Override
    protected void onStop(){
        super.onStop();

      // We need an Editor object to make preference changes.
      // All objects are from android.context.Context
      SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
      SharedPreferences.Editor editor = settings.edit();
      editor.putBoolean("silentMode", mSilentMode);

      // Commit the edits!
      editor.commit();
    }
}
```

2.7.2. Almacenamiento Interno

Se pueden guardar archivos directamente en la memoria interna del teléfono, no tendrán acceso a estos datos ni el usuario por medio de un gestor de archivos ni otras aplicaciones, únicamente nuestra aplicación. Además los datos son eliminados cuando se desinstala la aplicación.

Para crear y escribir en archivos en la memoria interna del teléfono:

1. Se debe llamar a openFileOutput( ) con el nombre del fichero y el modo de operación, para obtener un objeto FileOutPutStream.
2. Escribir en el archivo con write( ).
3. Cerrar el archivo con close( ).

Algunos modos para la apertura del fichero son: MODE_PRIVATE, crea un archivo privado para nuestra aplicación; MODE_APPEND, añade datos al final de un fichero ya creado; MODE_WORLD_READABLE, permite el acceso al archivo a otras aplicaciones, pero no su modificación; MODE_WORLD_WRITABLE, permite tanto leer como escribir en el archivo a otras aplicaciones.

Para leer de un fichero previamente creado:

1. Llamar al método openFileInput( ), con el nombre del fichero como parámetro. Obtendremos un objeto FileInputStream.
2. Escribir con el método read( ).
3. Cerrar el archivo con close( ).

A continuación se muestra un ejemplo de escritura:

```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME,
Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

2.7.3. Almacenamiento Externo

El almacenamiento externo es soportado por todos los dispositivos Android, ya sea en una unidad extraíble, como una tarjeta SD, o en la memoria interna no extraíble. Este tipo de almacenamiento está disponible para todas las aplicaciones, incluso el usuario puede acceder a estos archivos mediante un gestor de archivos o conectando el móvil al ordenador por USB.

En ocasiones es posible que no se encuentre disponible el almacenamiento externo por diferentes motivos, por ejemplo, que el dispositivo no dispone de tarjeta SD. Por eso es importante comprobar la disponibilidad de este tipo de almacenamiento antes de intentar acceder, el método getExternalStorageState( ).

```
boolean mExternalStorageAvailable = false;
boolean mExternalStorageWriteable = false;
String state = Environment.getExternalStorageState();

if (Environment.MEDIA_MOUNTED.equals(state)) {
    // We can read and write the media
    mExternalStorageAvailable = mExternalStorageWriteable = true;
} else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
    // We can only read the media
    mExternalStorageAvailable = true;
    mExternalStorageWriteable = false;
} else {
    // Something else is wrong. It may be one of many other states,
but all we need
    //  to know is we can neither read nor write
    mExternalStorageAvailable = mExternalStorageWriteable = false;
```

Este es un ejemplo de comprobación de la disponibilidad del almacenamiento externo.

2.7.4. Bases de Datos SQLite

Android proporciona un sistema gestor de bases de datos SQLite. Cualquier base de datos es únicamente accesible desde la aplicación que la creó, no desde fuera de ella. Este sistema gestor permite ejecutar cualquier sentencia SQL: creación de bases de datos relacionales, navegación entre tablas, etc.

La documentación de Android recomienda crear una clase que herede de SQLiteOpenHelper, y en el método onCreate( ) ejecutar un comando SQL para crear las tablas necesarias.

```
public class DictionaryOpenHelper extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 2;
    private static final String DICTIONARY_TABLE_NAME = "dictionary";
    private static final String DICTIONARY_TABLE_CREATE =
                "CREATE TABLE " + DICTIONARY_TABLE_NAME + " (" +
                KEY_WORD + " TEXT, " +
                KEY_DEFINITION + " TEXT);";

    DictionaryOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DICTIONARY_TABLE_CREATE);
    }
```

En este ejemplo se muestra la creación de una base de datos y una tabla.

2.7.5. Conexión de red

Es posible utilizar la conexión de red a modo de almacenamiento, siempre que esta esté disponible. Se puede utilizar una página web a modo de servidor para obtener los datos directamente de ésta. La principal ventaja es la necesidad de tener una conexión permanente a Internet.

2.8. SEGURIDAD Y PERMISOS [10]

Como ya dijimos anteriormente, cada aplicación Android se ejecuta en un proceso diferente, identificado mediante el ID de proceso. En Android, existen determinadas acciones que pueden causar un efecto negativo en el móvil, en otras aplicaciones o en los datos del usuario. Por ello, se necesita de algún tipo de control que determine qué procesos pueden acceder a las acciones restringidas, como pueden ser obtener la ubicación del móvil por GPS o redes inalámbricas, llamar por teléfono, acceder a Internet, etc.

2.8.1. Permisos de Usuario

Cada aplicación que quiera acceder a un recurso restringido, deberá indicarlo explícitamente en Android Manifest durante el proceso de desarrollo, de manera que el usuario podrá conocer los permisos que necesita una aplicación al instalarla.

En la clase "android.Manifest.Permission" se especifican todos los posibles permisos que se pueden conceder a una aplicación. A continuación se muestra un ejemplo de declaración de permisos.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.VIBRATE"/>
```

En este ejemplo, se autorizan los permisos de escritura en la memoria externa, acceso a Internet, acceso a la localización del dispositivo y al recurso hardware vibración. Como se puede observar, los recursos se declaran con el elemento "**<uses-permission>**", el cual dispone de una serie de atributos:

- **android:name:** Nombre del permiso que se desea conceder, el permiso debe existir en la clase android.Manifest.Permission.
- **android:label:** Etiqueta que se le puede dar al permiso para que el usuario pueda identificarlo.
- **android:permissionGroup:** Permite añadir el permiso a un grupo de permisos, los grupos de permisos se encuentran en android.Manifest.permission_group.
- **android:protectionLevel:** Determina el nivel de riesgo del permiso, el sistema puede decidir si otorga o no el permiso en función de este valor. Puede tomar valores de 0 a 3.
- **android:description:** Descripción del permiso.
- **android:icon:** Icono para la representación del permiso.

2.8.2. Firmado de aplicaciones

Todas las aplicaciones distribuidas en el Android Market deben estar firmadas por el desarrollador. Esto se usa como medida de seguridad y de garantía, ya que evita que otro usuario pueda hacerse pasar por nosotros y modificar nuestra aplicación. La firma de realiza mediante un certificado con los datos del creador de la aplicación. Una aplicación subida al Market no podrá ser actualizada, si el autor que intenta actualizarla no coincide con el que la subió en un primer momento.

2.9. INTERFAZ DE USUARIO [11]

En Android las interfaces de usuario se crean utilizando View y ViewGroup. Existen muchos tipos de views y viewgroups, todas ellas descendientes de una superclase View.

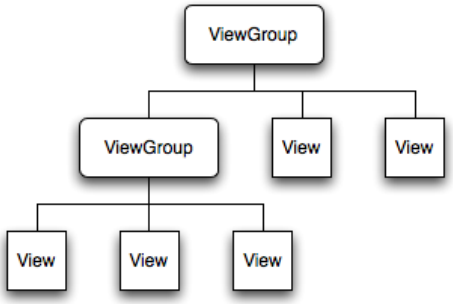


FIGURA 4 - ESQUEMA HERENCIA VIEW

El plugin de Android para Eclipse proporciona una potente herramienta para la creación de interfaces de usuario. Las interfaces son creadas mediante archivos XML guardados en el directorio /res/layout de nuestro proyecto. Estos archivos están compuestos de una serie de *views* que relacionadas entre sí dan lugar a la vista de usuario. Estos elementos son los que proporcionan una interactuación con el usuario, ya que podemos encontrar botones, cajas de texto, imágenes, etc.

2.9.1. Layouts

Además de los "widget" disponemos de otros elementos llamados "layouts", estos sirven para proporcionar una organización de los widgets, dependiendo del objetivo que queramos conseguir escogeremos uno u otro. Estos layouts pueden contener otros layouts, views o widget. A continuación se proporciona una descripción de los layouts disponibles:

- **Linear Layout:** Este tipo de layout se caracteriza por representar los elementos a los que contiene uno seguido de otro, ya sea en vertical o en horizontal.

- **Frame Layout:** Es el tipo más simple de layout, los elementos a los que contiene pueden relacionarse entre sí.

- **Table Layout:** Organiza los elementos a los que contiene en filas y columnas.

- **Relative Layout:** Permite agregar elementos con respecto a uno que se ha añadido anteriormente.

- **Absolute Layout:** Nos permite dibujar un elemento en pantalla en la posición que nosotros queramos, debemos indicar la posición en pixeles, teniendo en cuenta que las coordenadas (0,0) se encuentran en la esquina superior izquierda

2.9.2. Views y ViewGroups

Estos dos elementos se utilizan para la formación de las vistas, y proporcionan interactuación con el usuario ya que permiten insertar texto, pulsar botones, etc.

Los ViewGroup son conjuntos de Views predefinidas, las más comunes son:

- **Gallery:** Son un conjunto de imágenes que podemos cambiar deslizando el dedo por la pantalla.

FIGURA 5 - EJEMPLO GALLERY

- **GridView:** Despliega una tabla con scroll de m columnas y n filas.

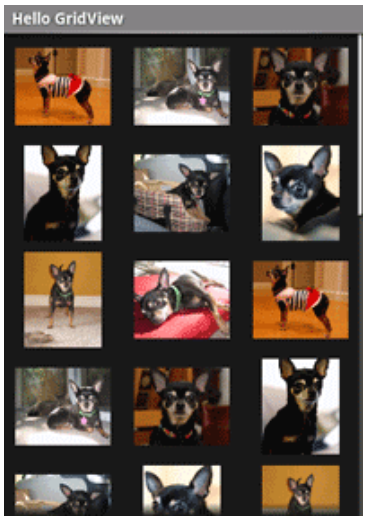


FIGURA 6 - EJEMPLO GRIDVIEW

- **ListView:** Lista con scroll igual que GridView pero de una sola columna.

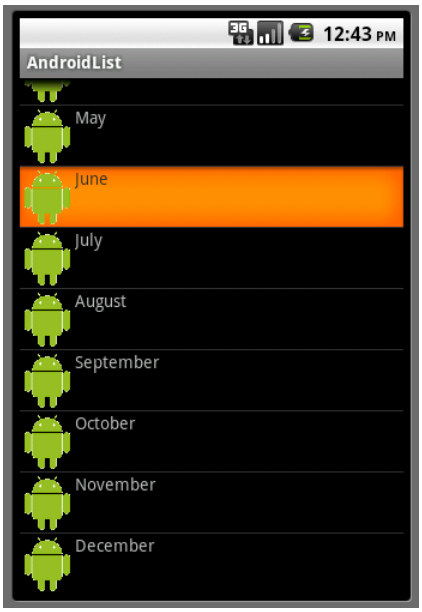


FIGURA 7 - EJEMPLO LISTVIEW

Además de los ViewGroups podemos encontrar simples View que podemos combinar para crear nuestra propia nuestra interfaz personalizada. Encontramos los siguientes como los más usados:

- **Button:** Elemento que dibuja un botón en la pantalla, que al pulsarlo suele ejecutarse alguna acción concreta.
- **Checkbox:** Botón con dos estados, que puede encontrarse marcado o desmarcado.
- **EditText:** Permite mostrar el texto que deseemos en la pantalla.
- **ImageView:** Muestra una imagen en la pantalla, que también se podría utilizar como un botón.
- **VideoView:** Reproduce un archivo de video.
- **WebView:** Muestra una página web.

2.10. ANDRIOD MANIFEST [12]

El archivo Android Manifest es el más importante de cualquier aplicación Android. Se encuentra en la raíz del proyecto, es generado automáticamente al crear un nuevo proyecto y se encuentra escrito en formato XML. En él se recogen datos tan importantes como los que se detallan a continuación:

- El nombre de los paquetes Java de la aplicación, que sirve como identificador único de esta aplicación.

- Describe los componentes que forman parte de la aplicación: Activities, Services y Content Providers, incluyendo datos como el nombre de la clase que ejecuta el componente y características de la clase.

- Indica el componente principal de la aplicación.

- Declara los permisos que necesita la aplicación para acceder a los recursos restringidos y para interactuar con otras aplicaciones.

- Declara los permisos requeridos para otras aplicaciones para acceder a los componentes de esta aplicación.

- Declara el nivel mínimo de versión de Android para poder ejecutar la aplicación.

- Lista las librerías externas que usa la aplicación.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.android.softkeyboard">
    <application android:label="@string/ime_name">
        <service android:name="SoftKeyboard"
                 android:permission="android.permission.BIND_INPUT_METHOD">
            <intent-filter>
                <action android:name="android.view.InputMethod" />
            </intent-filter>
            <meta-data android:name="android.view.im"
android:resource="@xml/method" />
        </service>
    </application>
</manifest>
```

Este es un ejemplo de Android Manifest de una aplicación que únicamente la compone un servicio.

3. ASPECTOS TEÓRICOS

3.1. CÁLCULO DE LA ZONA DE FRESNEL

La zona de fresnel es una zona de despeje adicional que hay que tener en cuenta en un radio enlace punto a punto. Las ondas al viajar en un espacio libre sufren un efecto de expansión, lo que permiten equiparar la onda con una elipsoide.

La primera zona de fresnel, región que encierra la elipsoide, es la portadora de la mayor cantidad de energía de la señal. Por ello, nos centraremos en análisis de esta zona.

La onda, a lo largo de su recorrido puede traspasar obstáculos, lo que se traduce en reflexiones y cambios de fase. El resultado es un aumento o disminución del nivel de intensidad de la señal recibida. La obstrucción máxima permitida para poder realizar un radioenlace es del 40%, aunque se recomienda que sea del 20%.

El cálculo de la zona de fresnel se realiza de la siguiente manera: en primer lugar se traza una línea recta entre los dos emplazamientos, a partir de esa línea se obtiene el radio de la primera zona de fresnel para cada punto con la siguiente fórmula.

$$r_n = 548 \sqrt{\frac{d1 \cdot d2}{f \cdot d}}$$

Fórmula para el cálculo de la 1ª zona de fresnel

Donde:

- r_n es el radio del punto enésimo de la 1ª zona de fresnel.
- *d1* distancia del origen al punto.
- *d2* distancia del punto al destino.
- f frecuencia en MHz
- *d* distancia entre los dos emplazamientos

Para el cálculo de la enésima zona de fresnel bastaría con multiplicar lo que se encuentra dentro de la raíz por n.

$$r_n = 548\sqrt{\frac{n \cdot d1 \cdot d2}{f \cdot d}}$$

Fórmula para el cálculo de la enésima zona de fresnel

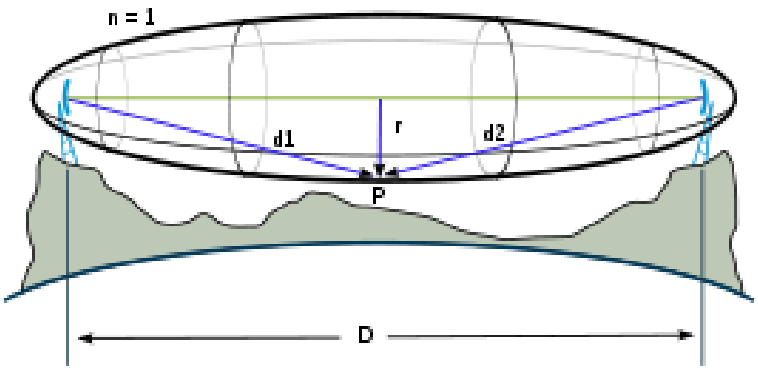


FIGURA 8 - REPRESENTACIÓN DE LA ZONA DE FRESNEL

3.2. CURVATURA DE LA TIERRA

Un factor importante y a tener en cuenta a la hora de obtener la zona de fresnel es la curvatura de la tierra, este factor puede parecer insignificante cuando los emplazamientos se encuentran muy cerca, pero tiene un gran impacto cuando entre ellos hay una gran distancia, ya que reduce la distancia entre la zona de fresnel y la del terreno.

Existe un factor que describe el grado de curvatura de la tierra en una cierta trayectoria conocido como factor K o factor de radio efectivo terrestre. Este factor depende del gradiente de refractividad por kilómetro con respecto a la altura (dN/dh), expresado en N-unidades/km.

$$k = \frac{1}{1 + a \cdot (dN/dh) \cdot 10^{-6}}$$

Fórmula para el cálculo del factor K.

Cuando K toma un valor 0<k<4/3, se dice que la onda es "subrefractada", si k>4/3 la onda es "superrefractada". En condiciones de atmósfera estándar se obtienen unos valores de k=4/3, pero se pueden obtener valores cercanos a 1 en áreas elevadas y secas, y hasta 3 en zonas húmedas o de costa.

Para un valor de k=∞, obtenemos que no existe curvatura de la tierra. En la siguiente imagen se muestra cómo afecta el factor k a la zona de fresnel.

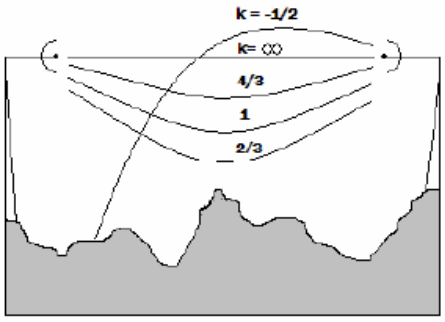


FIGURA 9 - VARIACIÓN DE LA ZONA DE FRESNEL EN FUNCIÓN DEL FACTOR K

Una vez obtenido el factor k, se puede obtener la curvatura de la tierra para cada punto del perfil radioeléctrico mediante la siguiente fórmula:

$$c_n = \frac{4}{51} \cdot \frac{d1 \cdot d2}{k}$$

Donde:

- d1 es la distancia desde el origen al punto.
- *d2* en la distancia del punto al origen.
- *k* es el factor K.

Con esta fórmula podemos obtener la curvatura para cada uno de los puntos. Una vez obtenidos los valores de curvatura, existen dos formas de añadirlos al cálculo de la zona de fresnel. La primera consiste en restarle a cada valor de la zona de fresnel el valor de curvatura de ese punto, y el segundo a cada valor de altitud sumarle el valor de curvatura.

4. RF TERRAIN PROFILE

En este apartado se detallarán los objetivos que se pretendían conseguir con la creación de la aplicación, así como los elementos que forman parte de ella y su proceso de creación.

4.1. DESCRIPCIÓN GENERAL

El objetivo principal de la aplicación RF Terrain Profiles es el de proporcionar una herramienta fiable y móvil para el proceso de cálculo de perfiles radioeléctricos, así cómo obtener la zona de fresnel asociada a los parámetros dados.

Por lo general, un perfil suele crearse ente dos emplazamientos posteriormente creados, por lo que, esta aplicación permite la creación, edición y borrado de emplazamientos. Además proporciona una herramienta para la orientación de las antenas de un perfil, proporcionándonos el azimut y la elevación de una antena respecto a la otra.

Estas son las funciones disponibles en la aplicación:

- Creación/Edición/Borrado de emplazamientos.
- Creación/Borrado de perfiles.
- Consulta de perfiles creados.
- Obtención del perfil radioeléctrico.
- Obtención de la zona de fresnel.
- Orientación de las antenas de un perfil.

FIGURA 10 - ICONO DE LA APLICACIÓN RF TERRAIN PROFILES

4.2. MENÚ PRINCIPAL

Esta es la pantalla inicial de la aplicación, en ella se da la opción al usuario de acceder a los menús de Emplazamiento, Perfil y Orientación.

FIGURA 11 - PANTALLA INICIAL

Este menú se crea mediante una clase especial llamada TabActivty, a la que mediante los métodos añadirTabX(), se le añaden cada una de las pestañas que queremos mostrar.

```
public class Principal extends TabActivity {


      private TabHost mTabHost;
    private Resources mResources;


    private static final String TAG_SCHEDULED = "Scheduled";
    private static final String TAG_CREATE = "Create";
    private static final String PREF_STICKY_TAB = "stickyTab";


    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.tab_principal);



        mTabHost = getTabHost();
        mResources = getResources();


        añadirTab1();
        añadirTab2();
        añadirTab3();


      }
```

Como se puede observar, las 3 pestañas se añaden en el método onCreate(), método que se ejecuta cuando se crea la actividad, en este caso al iniciar la aplicación.

```
private void añadirTab1() {


      Intent intent = new Intent(this,
MenuEmplazamiento.class);


        TabSpec spec = mTabHost.newTabSpec(TAG_SCHEDULED);



spec.setIndicator(mResources.getString(R.string.emplazamiento
s), mResources

                .getDrawable(R.drawable.emp));


        spec.setContent(intent);


        mTabHost.addTab(spec);


          }
```

Éste es un ejemplo de cómo se añade cada una de las pestañas. Como se puede observar, se crea un objeto TabSpec al que luego se la pasa un String (R.string.emplazamientos) y una imagen (R.drawable.emp) que serán los que luego se muestren en la pestaña. Finalmente, se añade la pestaña con la sentencia mTabHost.addTab(spec).

Dentro del menú de emplazamientos se ofrecen dos botones: el primero para crear, y el segundo para editar y borrar emplazamientos.

FIGURA 12 - MENÚ EMPLAZAMIENTOS

4.3.1. Creación de Emplazamientos

Tras pulsar en el botón correspondiente a la creación de emplazamientos, aparece el siguiente formulario.

FIGURA 13 - CREAR EMPLAZAMIENTO

Como se puede observar, los datos que definen un emplazamiento son el nombre y las coordenadas, latitud y longitud, en grados. Ambos son obligatorios, ya que no es posible crear un emplazamiento dejando uno de los dos campos en blanco.

En cuanto a las coordenadas, existen 3 formas de introducirlas:

- La primera es manualmente, introduciendo los valores de latitud y longitud directamente en los EditText. Ambos valores deben estar en grados decimales, permitiéndose valores negativos.

  En cuanto al campo latitud, un valor negativo indica que el emplazamiento a crear se encuentra en el hemisferio sur, mientras que un valor negativo en el campo longitud indica que el emplazamiento se encuentra al Oeste del meridiano de Greenwich.

- Mediante el uso del GPS: Con el botón “Mi Ubicación” podemos acceder a nuestra posición mediante el GPS del móvil. Si el GPS no se encuentra activado, nos mostrará un mensaje y la acción no se llevará a cabo.

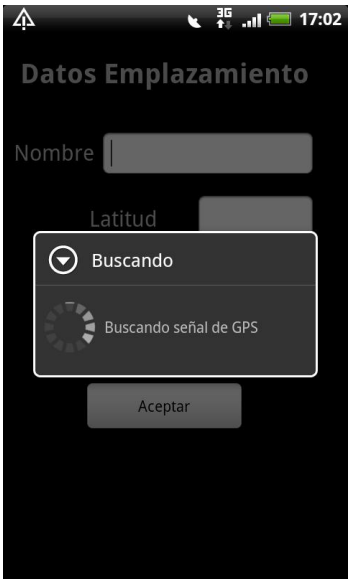


FIGURA 14 - OBTENCIÓN DE COORDENADAS POR GPS

- Girando el móvil podemos acceder a un MapView, mediante el cual podemos seleccionar la ubicación del emplazamiento directamente en el mapa, manteniendo pulsado un instante de tiempo sobre la posición donde queremos crearlo. Además, aparecerá un icono informándonos de la posición de este.

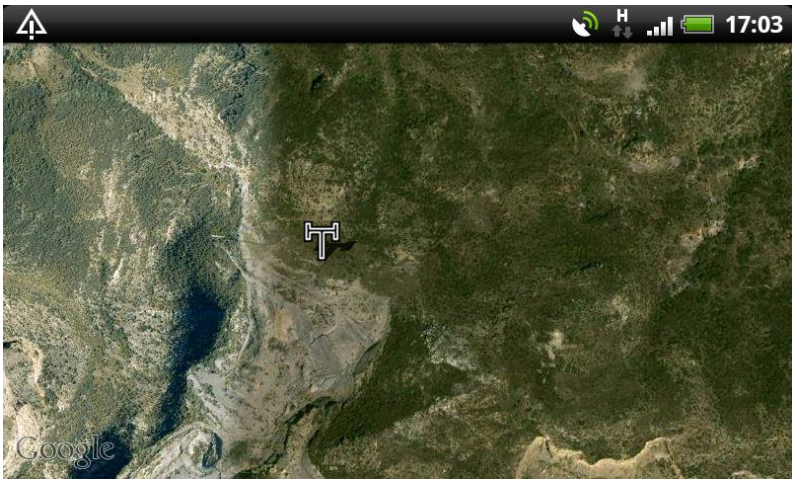


FIGURA 15 - OBTENCIÓN DE COORDENADAS A TRAVÉS DEL MAPA

Una vez tenemos introducidos todos los parámetros del emplazamiento, pulsamos el botón Aceptar para crearlo, y si todo ha ido bien deberá aparecer un mensaje positivo en pantalla.

Los Emplazamientos creados se guardan en una carpeta llamada Sites, dentro de RF Terrain profiles. Allí se guarda cada emplazamiento en un archivo XML.

```
<xml>
	<nombre>Cartagena</nombre>
	<descripcion/>
	<tipo>null</tipo>
	<xcoord>-0.9824</xcoord>
	<ycoord>37.599671</ycoord>
</xml>
```

En este ejemplo se puede ver un emplazamiento creado, donde xcoord es la longitud e ycoord la latitud.

4.3.2. Edición de emplazamientos

Otra de las opciones disponibles en el menú es la de edición de emplazamientos, una vez pulsado el botón aparece un ListView con los emplazamientos anteriormente creados y ordenados alfabéticamente.

FIGURA 16 - LISTA DE EMPLAZAMIENTOS CREADOS

En la figura de arriba se puede ver una serie de emplazamientos creados. Pulsando sobre uno de ellos nos lleva directamente a la pantalla de creación de emplazamientos. Una vez en ella podemos cambiar los datos que deseemos del emplazamiento y guardarlo pulsando el botón Aceptar. El XML se actualizará con los nuevos datos.

Si la edición ha sido correcta se mostrará un mensaje en pantalla informándonos de ello, en caso negativo también.

4.3.3. Borrado de Emplazamientos

El proceso de borrado de emplazamientos es similar al de edición. Pulsamos el botón de Editar en el menú emplazamientos y nos vuelve a salir la lista anteriormente comentada. Una vez encontrado el emplazamiento que deseamos borrar mantenemos pulsado sobre él hasta que nos aparezca el mensaje de confirmación.

Si estamos seguros pulsamos en Sí y el XML del emplazamiento se borrará. En caso negativo podemos pulsar en No o la tecla retroceso para cancelar la operación.

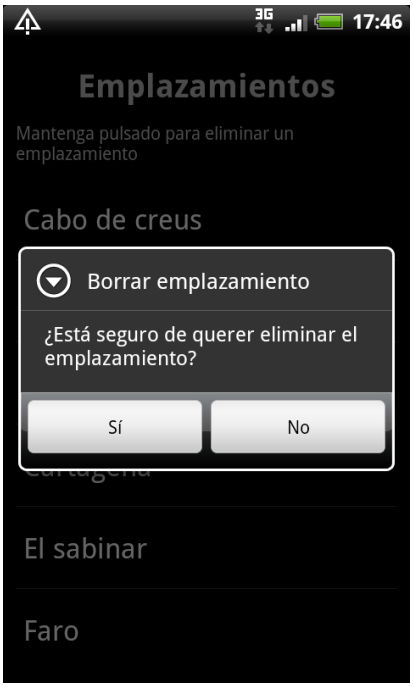


FIGURA 17 - BORRADO DE UN EMPLAZAMIENTO

Al igual que en el caso de los Emplazamientos, si pulsamos en la pestaña de Perfil nos aparecen dos opciones, en primer lugar un botón para crear un nuevo perfil a través de dos emplazamientos, y un segundo botón donde podemos consultar los perfiles anteriormente creados.

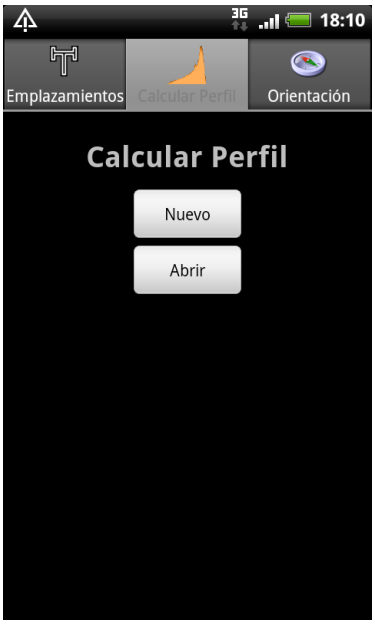


FIGURA 18 - MENÚ PERFILES

4.4.1. Crear nuevo perfil

Una vez pulsado el botón de nuevo perfil aparece una pantalla parecida a la de creación de un nuevo emplazamiento. A simple vista se pueden observar algunas diferencias, por ejemplo el botón de Seleccionar Emplazamiento, la casilla de altura del emplazamiento y un checkbox para guardarlo.

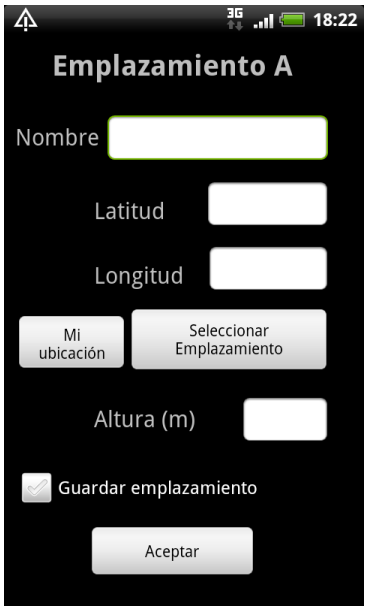


FIGURA 19 - EMPLAZAMIENTO PERFIL

El botón Seleccionar Emplazamiento permite seleccionar un emplazamiento que hayamos creado anteriormente, rellenando automáticamente los campos Nombre, Latitud y Longitud. En cambio, el campo altura será algo que tenemos que rellenar manualmente con los metros a los que se encuentra la antena del suelo.

Los datos del emplazamiento se pueden modificar libremente (como vimos en la sección 4.1.1. Creación de Emplazamientos) aunque se haya seleccionado la opción de seleccionar un emplazamiento que ya está creado.

Finalmente, el checkbox Guardar emplazamiento, sirve para, además de utilizar el emplazamiento para calcular el perfil, se guarda para poder utilizarlo para posteriores perfiles.

Una vez introducidos los datos del emplazamiento A, nos volvería a salir una pantalla similar para introducir los datos del Emplazamiento B. Una vez concluido este formulario, pasaríamos a la siguiente pantalla. En ésta se vuelve a mostrar un formulario, pero en este caso distinto.

FIGURA 20 - DATOS PERFIL

Gracias a este formulario se pueden recoger datos tan importantes a la hora de crear un perfil radio eléctrico como son: el nombre con el que se guardará el perfil, la frecuencia para el cálculo de la zona de fresnel (3.2. Cálculo de la zona de fresnel) y el factor k, necesario para el cálculo de la curvatura de la tierra (3.3. Curvatura de la tierra).

Además, se incorpora un elemento Spinner para facilitar la introducción de la frecuencia, permite seleccionar las unidades en las que se va a introducir.

FIGURA 21 - SELECCIÓN DE UNIDADES DE FRECUENCIA

Una vez completado este formulario, los datos del perfil se guardan en un archivo XML dentro de una carpeta con el nombre del perfil, en la ruta /RF Terrain profiles/Profiles/. Además del archivo con los datos del perfil, se guarda otro archivo llamado res.xml. Este archivo es el resultado de la petición al servidor de la Elevation API de Google y contiene las elevaciones de los puntos intermedios entre los dos perfiles.

Llegados a este punto ya tenemos todos los datos de los emplazamientos y del perfil, por lo que únicamente queda representarlo. Dependiendo de la posición en la que se encuentre el móvil tendremos una representación u otra.

Si el móvil se encuentra en posición vertical (portrait) obtendremos la vista de un MapView centrado en el punto medio de los dos emplazamientos, y en la posición de cada emplazamiento una imagen y el nombre que le hayamos dado a cada uno de ellos, además de una línea azul uniendo los dos emplazamientos.

FIGURA 22 - REPRESENTACIÓN DEL PERFIL EN EL MAPA

En el caso en el que el dispositivo se encuentre en posición horizontal (landscape), obtendremos la imagen del perfil radioeléctrico.

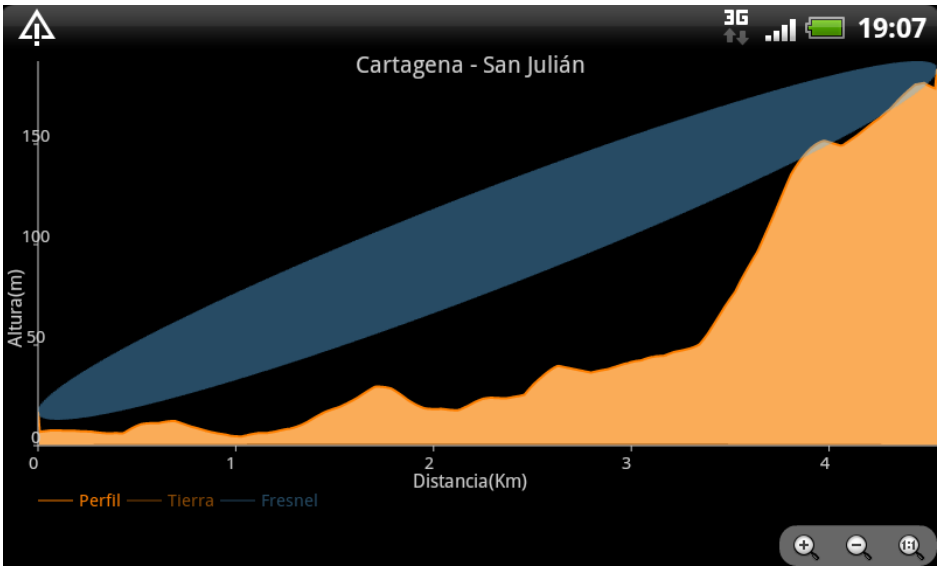


FIGURA 23 - *PERFIL RADIOELÉCTRICO*

Como podemos observar se muestra el perfil radioeléctrico. En él, se pueden diferenciar tres componentes, por un lado y en azul tenemos la 1º zona de fresnel, que va desde el emplazamiento A hasta el B.

En naranja encontramos el perfil terrestre que existe desde el punto A al B. En marrón se diferencia la curvatura de la tierra, aunque en este ejemplo es casi inexistente dada la cercanía de los dos emplazamientos.

Como comentábamos anteriormente, los datos del perfil se guardan en un archivo XML, concretamente el archivo tiene esta organización:

```
<xml>
    <nombre>Cartagena - San Julián</nombre>
    <latOrigen>37.609192</latOrigen>
    <lonOrigen>-0.979999</lonOrigen>
    <alturaOrigen>10.0</alturaOrigen>
    <latDestino>37.592054</latDestino>
    <lonDestino>-0.942918</lonDestino>
    <alturaDestino>10.0</alturaDestino>
    <frecuencia>5.0E8</frecuencia>
    <udFrecuencia>3</udFrecuencia>
    <factorK>1.33</factorK>
    <distancia>4545.513141635092</distancia>
    <resolucion>0</resolucion>
    <nombreOrigen>Cartagena</nombreOrigen>
    <nombreDestino>San Julián</nombreDestino>
</xml>
```

Como se puede observar los datos del perfil se guardan ordenados por emplazamientos, es decir, primero los datos del emplazamiento origen, después lo del emplazamiento destino y después los del perfil, a excepción del nombre.

4.4.2. Abrir Perfil

La aplicación también ofrece la posibilidad de consultar perfiles que hayamos creado. Para esto basta con pulsar en el botón Abrir dentro del menú Perfil. Tras pulsar el botón nos aparecerá una pantalla similar a la de selección de emplazamiento, con la diferencia de que esta nos muestra los perfiles creados.

FIGURA 24 - LISTA DE PERFILES CREADOS

Al pulsar sobre uno de ellos, directamente aparecerán las figuras 15 ó 16, dependiendo de la posición del móvil. El proceso de obtención de los datos del perfil del XML es sencillo, se utiliza un XML Parser para obtenerlos. A continuación se muestra un ejemplo:

```
DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();

DocumentBuilder db = dbf.newDocumentBuilder();

Document doc = db.parse(new
File(ruta+"/Profiles/"+nombre+"/"+nombre+".xml"));




Element rootElement = doc.getDocumentElement();
```

En la variable de tipo String nombrePerfil, tendríamos el valor del XML en ese campo, siguiendo el ejemplo anterior, tomaría el valor "Cartagena – San Julián". El proceso es equivalente para la obtención de los demás datos, una vez se tienen todos, se pasa a representar el perfil.

4.4.3. Borrar Perfil

Como indica la frase que hay debajo de Perfiles, tenemos la opción de borrar perfiles. Para esto únicamente tenemos que mantener pulsado sobre el perfil deseado hasta que aparezca el cuadro de confirmación, al igual que ocurría en el apartado 4.1.3. Borrado de Emplazamientos,

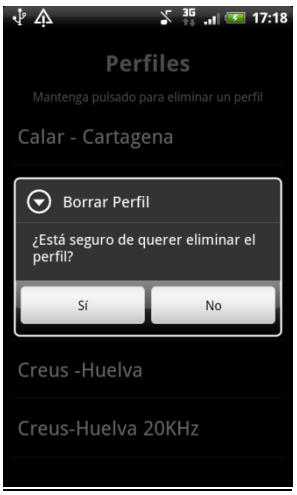


FIGURA 25 - BORRADO DE PERFILES

4.5. ORIENTACIÓN

La última pestaña de las que disponemos en la aplicación es la de Orientación. Al pulsar sobre ella nos aparece el menú de orientación. En este caso sólo con una única opción, la de seleccionar perfiles. Esta parte de la aplicación provee al usuario de una herramienta para la orientación de las antenas de un perfil, devolviéndole datos tan importantes como el azimut o la elevación.

FIGURA 26 - MENÚ ORIENTACIÓN

Pulsando sobre el botón aparece una lista similar a la del apartado 4.3.2. Abrir Perfil, con la diferencia de que en este caso no es posible el borrado de perfiles, únicamente su consulta. Tras pulsar en un perfil se accede a una pantalla titulada "Seleccionar Dirección", que ofrece al usuario la opción de orientar la antena que va desde el emplazamiento A hasta el B, o por el contrario la que va desde el punto B al A.

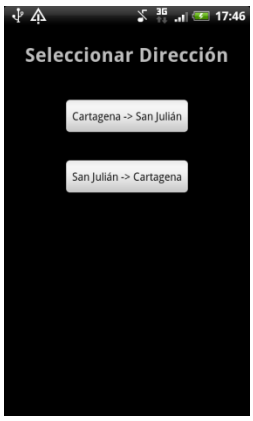


Figura 27 - Seleccionar dirección

Una vez el usuario haya seleccionado la opción deseada, aparecerá en la pantalla la información deseada, además de una brújula informándonos de la ubicación del emplazamiento.

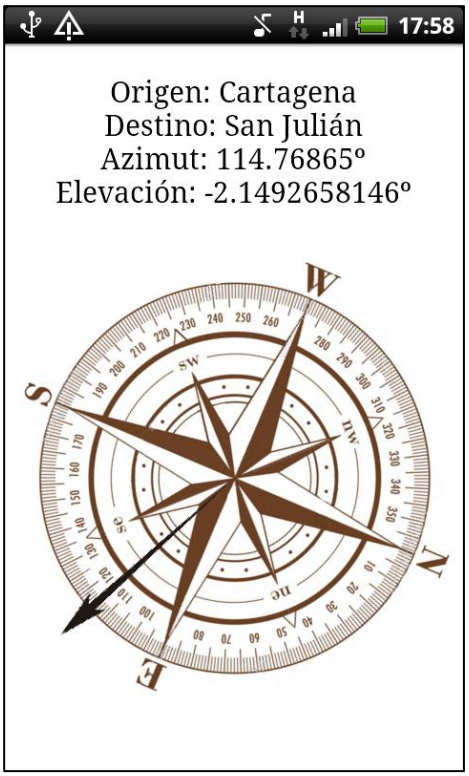


FIGURA 28 - BRÚJULA

Como se observa en la imagen se dispone de una brújula, la cual apunta siempre al norte, y además, una flecha que nos informa de la dirección de un emplazamiento respecto al otro, es decir, si estamos en el emplazamiento A apuntaría hacia el B siempre que se seleccione en la pantalla de "Seleccionar Dirección".

Además de la brújula, se proporciona información de un emplazamiento respecto al otro, para la orientación de la antena, en este caso concreto se indica que la antena situada en el emplazamiento de Cartagena debería girarse 114.76865º con respecto al Norte en el sentido de las agujas del reloj, y que

debería bajarse con respecto a la vertical -2.1492658146º, o lo que es lo mismo, subirla 2.1492658146º.

A continuación se muestra un diagrama explicativo para el uso de la elevación y el azimut en la orientación de antenas.

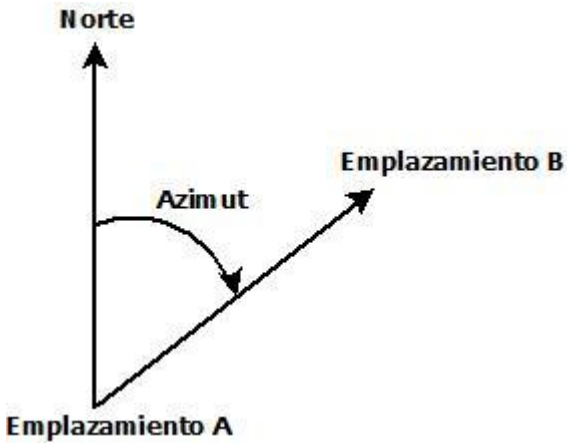


FIGURA 29 - DIAGRAMA AZIMUT

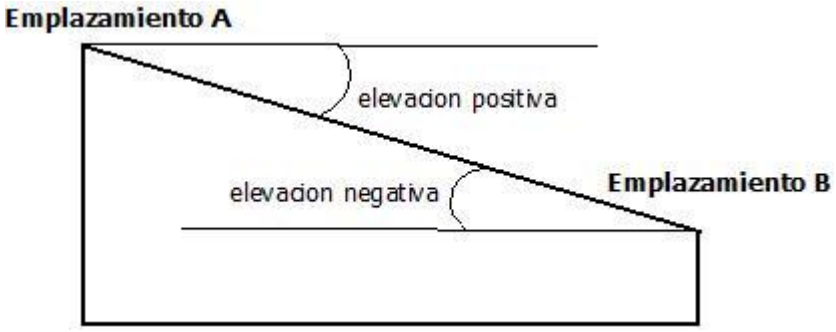
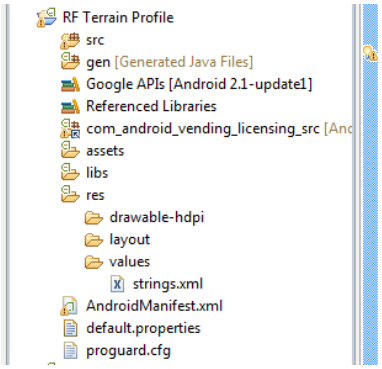


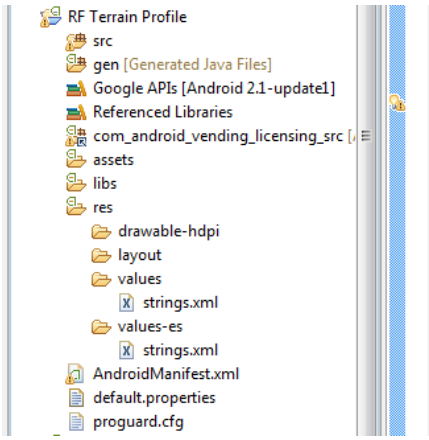
FIGURA 30 - DIAGRAMA ELEVACIÓN

4.6. TRADUCCIÓN DE LA APLICACIÓN

En programación Android tenemos una carpeta destinada a los recursos que puede acceder la aplicación: imágenes, plantillas, strings, etc… Las imágenes se guardan en una carpeta llamada “drawables”. Además de ésa tenemos la carpeta “values” y dentro de ella un archivo en el que se guardan todos los Strings utilizados en el programa.



La carpeta “values” actúa como carpeta por defecto. En el ejemplo de arriba únicamente se dispone de esta carpeta, por lo que siempre se utilizarán los valores de los String que en ella se contienen.



Por el contrario, en el ejemplo de arriba se puede observar que existen las carpetas “values” y “values-es”. La extensión “-es” indica que se utilizan valores diferentes a los de la carpeta “values” cuando el idioma que utiliza el dispositivo es el español. La carpeta “values” contiene los Strings en inglés, mientras que la carpeta “values-es” lo hace en español. Por consiguiente, cuando el móvil se encuentre configurado en español, la aplicación aparecerá en español, y para cualquier otro idioma aparecerá en inglés.

4.7. OTROS ELEMENTOS UTILIZADOS

En esta sección se detallarán otros elementos utilizados para la creación de la aplicación, como son las peticiones de la elevación del terreno al servidor de Google o la representación de los valores gracias a la librería AChartEngine.

4.7.1. AChartEngine [13]

AChartEngine es una librería de libre uso para representación de gráficos en dispositivos Android. Este framework proporciona una herramienta potente para realizar todo tipo de gráficos a nuestro gusto.

Permite realizar gráficos de puntos, líneas, diagramas de barras, gráficos de dispersión y circulares. En el caso concreto de RF Terrain Profiles, se han utilizado gráficos de puntos para la representación de la zona de fresnel, el perfil del terreno y la curvatura de la Tierra.

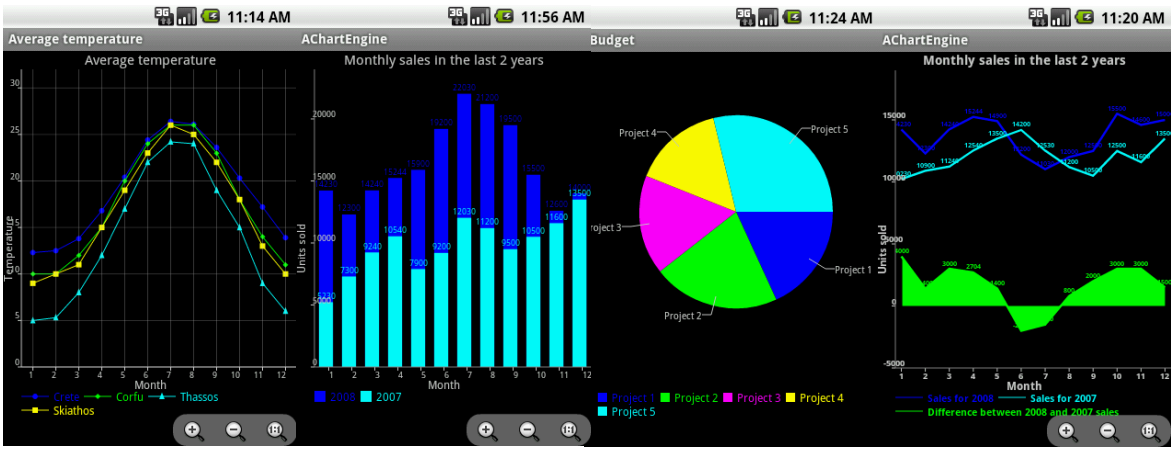


FIGURA 31 - EJEMPLOS ACHARTENGINE

4.7.2. Google Elevation API [14]

La Elevation API de Google proporciona una interfaz sencilla para consultar la elevación de cualquier punto de la Tierra. Permite la consulta de la elevación de un único punto, o si lo deseamos, puede devolvernos la elevación de todos los puntos de una ruta. La API dispone de valores para cualquier punto de la Tierra, tanto positivos (puntos sobre el nivel del mar) como negativos (puntos bajo el nivel del mar), en caso de que no se disponga de la elevación para el punto exacto solicitado, el servicio calculará la media de los 4 puntos más cercanos.

Límites de uso

La Google Elevation API tiene asociada una restricción en cuanto al número de peticiones de elevación, el número máximo de peticiones es de 25.000 al día. Una vez sobrepasado este límite en un día, el servicio dejará de funcionar temporalmente. Además se impone una restricción más, y es que el número de peticiones de elevación en una solicitud, no puede ser mayor de 512.

Solicitudes de elevación

Las solicitudes a la API se realizan mediante una petición http de la siguiente sintaxis.

```
http://maps.google.com/maps/api/elevation/outputFormat?parameters
```

Donde outputFormat indica el formato de salida en el que deseamos la respuesta, puede ser "/json" para obtener la salida en JSON o "/xml" para obtener la respuesta en XML.

En cuanto a la partícula parameters dependerá su formato del tipo de petición a realizar, existen dos tipos de peticiones:

- Locations: Con este tipo de petición se pretende conocer la elevación de uno o más puntos, pero que no guardan relación entre sí. En parameters se debe añadir la variable "locations", seguida de las coordenadas (latitud, longitud) del punto sobre el que queremos realizar la consulta

```
http://maps.google.com/maps/api/elevation/json?locations=
39.7391536,-104.9847034&sensor=false
```

- Path: Este tipo de petición informa de la elevación de dos puntos que nosotros le pasemos como parámetros, y además, devuelve la elevación de un número de puntos intermedios, a la petición se le deben añadir los siguientes parámetros, "path" seguido de las coordenadas de dos puntos separados por "|" y "samples" seguido del número de puntos intermedios.

```
http://maps.google.com/maps/api/elevation/json?path=36.5785
81,-118.291994|36.23998,-116.83171&samples=3&sensor=false
```

La variable "sensor" indica si se está utilizando o no un sensor de posicionamiento, como puede ser un GPS.

5. CONCLUSIONES

5.1. CONCLUSIONES

Con este proyecto:

- Se ha puesto de manifiesto las posibilidades que ofrece el sistema operativo Android para el desarrollo de aplicaciones orientadas a la planificación de sistemas de radiocomunicaciones. En concreto se ha implementado una aplicación de cálculo de perfiles radioeléctricos para dispositivos móviles.

- Se ha desarrollado una aplicación que permite gestionar emplazamientos creados por el usuario, representar un perfil junto con la primera zona de Fresnel en cualquier parte del mundo mediante un terminal móvil, así como facilitar la orientación de las antenas para cada emplazamiento del radio enlace.

Como futuras líneas de trabajo para la aplicación se podría estudiar la posibilidad de portar esta aplicación a otros sistemas operativos para móviles, como pueden iOs o Windows Phone, y comercializarla para estas plataformas, al igual que se ha hecho con Android.

Además existe la posibilidad de añadirle nuevas funcionalidades a la aplicación: posibilidad de tomar fotos de un emplazamiento y almacenarlas junto al emplazamiento, interaccionar con otras aplicaciones como RadioEarth desarrollada por el Grupo de Sistemas de Comunicaciones Móviles (SiCoMo) de la Universidad Politécnica de Cartagena.

En líneas generales, se podría decir que la realización de este proyecto ha sido satisfactoria. Bajo mi punto de vista, como desarrollador, me ha supuesto un acercamiento al S. O. Android, una motivación extra al tratarse de lenguaje Java y poder aplicar todo lo aprendido durante estos 3 años de carrera, aunque también ha supuesto algunas dificultades comenzar a programar en esta plataforma.

6. BIBLIOGRAFÍA

[1] Web de la Open Handset Alliance

http://www.openhandsetalliance.com/

[2] Licencia Apache

http://www.apache.org/licenses/

[3]Posición de Android en el mercado

http://www.canalys.com/newsroom/android-takes-almost-50-share-worldwide-smart-phone-market

[4] Definición Android

http://developer.android.com/guide/basics/what-is-android.html

[5] Componente Activity

http://developer.android.com/guide/topics/fundamentals/activities.html

[6] Componente Service

http://developer.android.com/guide/topics/fundamentals/services.html

[7] Componente Intent

http://developer.android.com/guide/topics/intents/intents-filters.html

[8] Componente Content Providers

http://developer.android.com/guide/topics/providers/content-providers.html

[9] Almacenamiento en Android

http://developer.android.com/guide/topics/data/data-storage.html

[10] Seguridad en Android

http://developer.android.com/guide/topics/security/security.html

[11] Interfaz de Usuario

http://developer.android.com/guide/topics/ui/index.html

[12] Android Manifest

http://developer.android.com/guide/topics/manifest/manifest-intro.html

[13] Librería AChartEngine

http://www.achartengine.org/

[14] Google Elevation API

http://code.google.com/intl/es-ES/apis/maps/documentation/elevation/