UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de navegador para coches basado en Android y OpenStreetMap

Trabajo Final de Grado

Grado en ingeniería informática

Autor: Peris Martinez, David

Director: Tavares Calafate, Carlos

Curso 2013 - 2014

Resumen

Este proyecto esta enfocado en la creación de una interfaz de usuario para un navegador GPS en Android. Esta aplicación permite al usuario seleccionar varios puntos de la ruta que son enviados al servidor el cual devuelve la ruta más rápida que cruza todos los puntos. Además, tiene una función para navegar por esa ruta, mostrando siempre la acción a realizar en cada punto. La aplicación permite, además, el almacenamiento de los lugares y rutas favoritos. Por otra parte, el desarrollo se ha realizado siguiendo la metodología de modelado de prototipos.

This project consists in the development of an user interface for an Android GPS navigator. This app allows the user to select several route points which are sent to the server that returns the fastest route that crosses all points. Futhermore it has a functionallity to navigate by this route, ever showing the action to improve in every point. Moreover, the app allows the storage of the favourite routes and places. For the development of this solution a methodology based on prototypes modeling was adopted.

Palabras clave: interfaz, GPS, navegador, Android, aplicación, modelado de prototipos

Keywords: interface, GPS, navigator, Android, app, prototypes modeling

Índice de contenido

1.Introducción

1.1.Motivación

En el mundo de los *smartphones*, bien es sabido que cualquier sistema operativo que se precie debe contener una serie de aplicaciones esenciales. Una de estas aplicaciones esenciales es sin duda el llamar por teléfono, de ahí que reciban el nombre de *smartphones*. Sin embargo hay otro tipo de aplicación que es igual de importante: Una aplicación de navegación por gps. En el mercado hay una gran cantidad de estas aplicaciones para los tres sistemas operativos móviles más importantes: Windows phone, IOS y Android. De estos tres, mi aplicación esta construida sobre Android y por ello será este el sistema operativo que utilizaré de contexto.

Android es el sistema operativo por excelencia de Google y a diferencia de sus competidores es de código abierto y con licencia GPL, por lo que cualquier persona puede desarrollar aplicaciones para este sistema de forma gratuita[1] [2]. La mayoría de fabricantes incorporan en sus *smartphones* la aplicación de Google maps, que como su nombre indica, es el navegador desarrollado por Google para su sistema operativo. Esta aplicación esta fuertemente ligada a la otra aplicación estrella de Google: Google Navigation, que viene a ser una mejora en el sistema de navegación de Google maps y que incorpora indicaciones por voz.

El motivo por el cual he hecho este navegador se podría resumir en dos ideas principales: OpenStreetMap y el servidor al que hace la petición de que calcule la ruta. El proyecto OpenStreetMap, del cual profundizaré en capítulos posteriores, se podría resumir en que fue ideado con la idea de utilizar datos geográficos, tales como planos de calles, de forma libre. Esto significa que estos mapas, a diferencia de los de Google, no tienen restricciones legales[3]. Por otra parte, el servidor relacionado con este proyecto se encargará de recibir los datos de los usuarios de la aplicación y en función de estos datos cuando se le haga una petición de ruta devolverá la ruta más rápida en función del tiempo que hayan tardado los demás usuarios en pasar por ese tramo.

Si por ejemplo, el usuario de la aplicación se quiere desplazar desde un pueblo a otro, normalmente elegiría para viajar la carretera que conecta esos dos pueblos y mide 1 kilómetro en comparación con la autopista que mide 10. En circunstancias normales el trayecto por la carretera dura apenas 2 minutos con respecto a los 7 minutos que cuesta moverse por la autopista. No obstante, si se produce un atasco en esa carretera el tiempo estimado del viaje sería de 20 minutos con respecto a los 7 minutos que cuesta desplazarse por la autopista. Por desgracia, normalmente los conductores no saben cuando en una carretera o autopista va a haber un atasco y no se dan cuenta hasta que se encuentran metidos en él. Lo interesante de esta aplicación es que servirá para alertar a los conductores de las rutas más fluidas y así hacer más rápido el trayecto y evitar atascos.

1.2.Objetivos

Como objetivos tenemos:

• Crear una interfaz para interactuar con el mapa de la aplicación de tal forma que se pueda desplazar, rotar o hacer zoom. También se debe poder situar marcadores en el mapa para fijar destinos y puntos de ruta.

• Implementar un sistema para guardar los lugares o rutas favoritos para poder usarlos posteriormente. A su vez, estos lugares deben de poder compartirse con otros usuarios de la aplicación.

• Incorporar un buscador de lugares para que, junto con los marcadores, permita predefinir la ruta. Este buscador hace una petición al servidor para obtener la posición del lugar buscado.

•Implementar una función para trazar una ruta sobre el mapa. Se debe de enviar las posiciones de los puntos al servidor, cuya respuesta será la ruta trazada.

• Implementar el modo de navegación de la aplicación, es decir, la funcionalidad que permita al usuario conocer su posición en todo momento a la vez que le indique la acción que deberá tomar en el

siguiente punto de ruta. Al llegar al destino se considerará como finalizada la navegación y este modo se desactivará.

2.OpenStreetMap

Como he comentado antes, el proyecto OpenStreetMap busca la liberación de los datos geográficos, la libertad de poder usar mapas sin miedo a quebrantar restricciones legales. Su objetivo es el de tener un registro de cada localización geográfica del mundo. A pesar de su nombre, OpenStreetMap no solo se encarga del trazado de mapas, sino que también tiene funciones de trazado de rutas, geocodificación (Encontrar las coordenadas para un objeto dado) y análisis espacial. Por otra parte los mapas de la página web son solo ejemplos y un usuario se puede crear sus propios mapas[4].

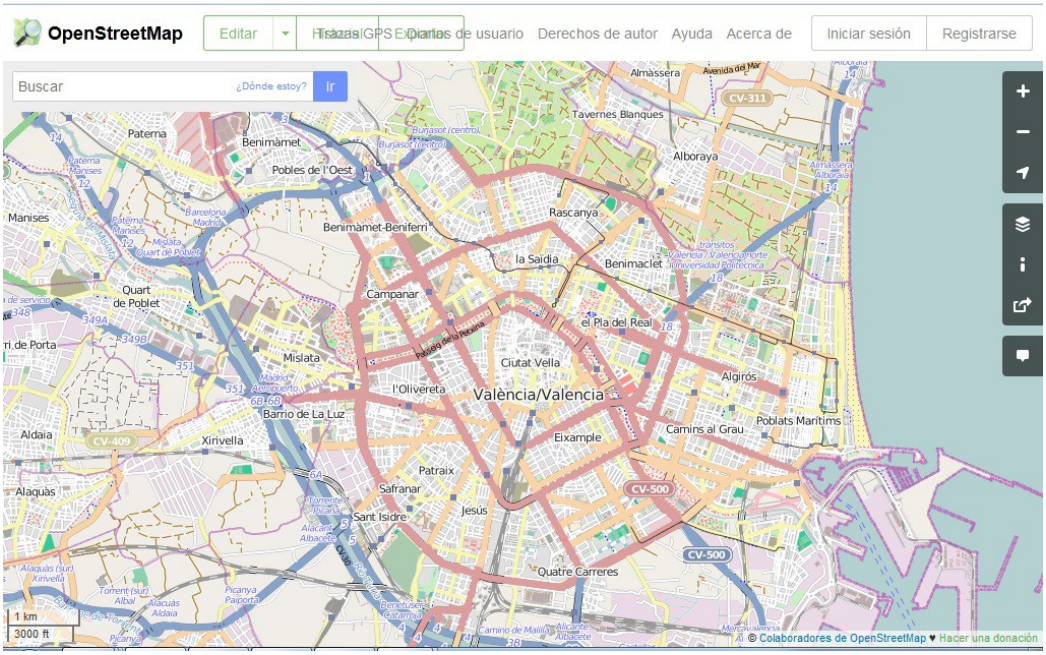

Figura 2. Captura de pantalla de la aplicación web de OpenStreetMap.

Si nos fijamos en la FAQ de la página web del proyecto observaremos toda la información importante explicada por sus desarrolladores en la forma de pregunta-respuesta[5]:

2.1.¿Por qué elegimos OpenStreetMap?

En muchas partes del mundo, como en España, los datos geográficos públicos no son de libre uso. Por lo general, en estos países se ha delegado la tarea de realizar los levantamientos de este tipo de información a diversas instituciones dependientes del gobierno (como el IGN en España) que a su vez venden esta cartografía a personas como usted, obteniendo una ganancia por ello.

En países como EE.UU. los datos cartográficos en bruto (sin tratar) pertenecientes al gobierno, como los ficheros TIGER, son de dominio público, sin embargo los editados y corregidos poseen, por lo general, derechos de autor para poder comerciar con ellos.

2.2.¿Y por qué no utilizo Google Maps para mis datos?

Es un asunto de libertad, no de precio. Para entender el concepto, debes pensar en libre (*free* en inglés) como en "libertad de expresión", no como en "cerveza gratis". La cartografía de Google es por lo tanto "gratuita" pero no "libre"

2.3.¿Por qué en ocasiones los datos son inconsistentes?

Los datos recopilados por OpenStreetMap son enviados por los usuarios, los cuales pueden crear intencionadamente localizaciones erroneas. No obstante esto es algo bueno dado que la mayoría de usuarios se dedican a reparar estos errores y los datos se pueden actualizar de forma más rápida. (En lugar de un grupo de trabajadores, OpenStreetMap funciona gracias a miles de personas altruistas).

2.4.¿Bajo que licencia se distribuyen los datos del proyecto OpenStreetMap?

Actualmente la cartografía de OpenStreetMap se distribuye bajo los términos de la licencia Open Database License (ODbL): los contribuidores al proyecto permiten la copia y distribución de los datos y hacer obras derivadas de éstos siempre que se reconozca a la Fundación OpenStreetMap y a sus contribuidores como sus autores y se comparta bajo una licencia idéntica a esta.

3.Entorno de trabajo utilizado

3.1.Eclipse IDE

He elegido este IDE (integrated development environment o entorno de desarrollo integrado) para el desarrollo del proyecto por su excelente integración con el SDK de Android, por su sencillez de uso y por ser uno de los IDE más completos del mercado.

3.1.a)Descripción

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. La Free Software Foundation ha dicho que ambas licencias son licencias de software libre, pero son incompatibles con Licencia pública general de GNU (GNU GPL).

En cuanto a las aplicaciones clientes, Eclipse provee al programador con frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc[6].

3.1.b)Funcionamiento

La versión que he utilizado es la Juno 4.2. Se trata de la versión más actualizada del IDE en el lapso de tiempo en el que realicé el trabajo.

Los proyectos de Eclipse se pueden estructurar en subpaquetes, dentro de los cuales se hayan las clases y recursos necesarios para la ejecución del proyecto. Esta ejecución se realiza desde el botón del "play" situado en la parte superior desde el cual se pueden elegir los argumentos u opciones de ejecución deseados.

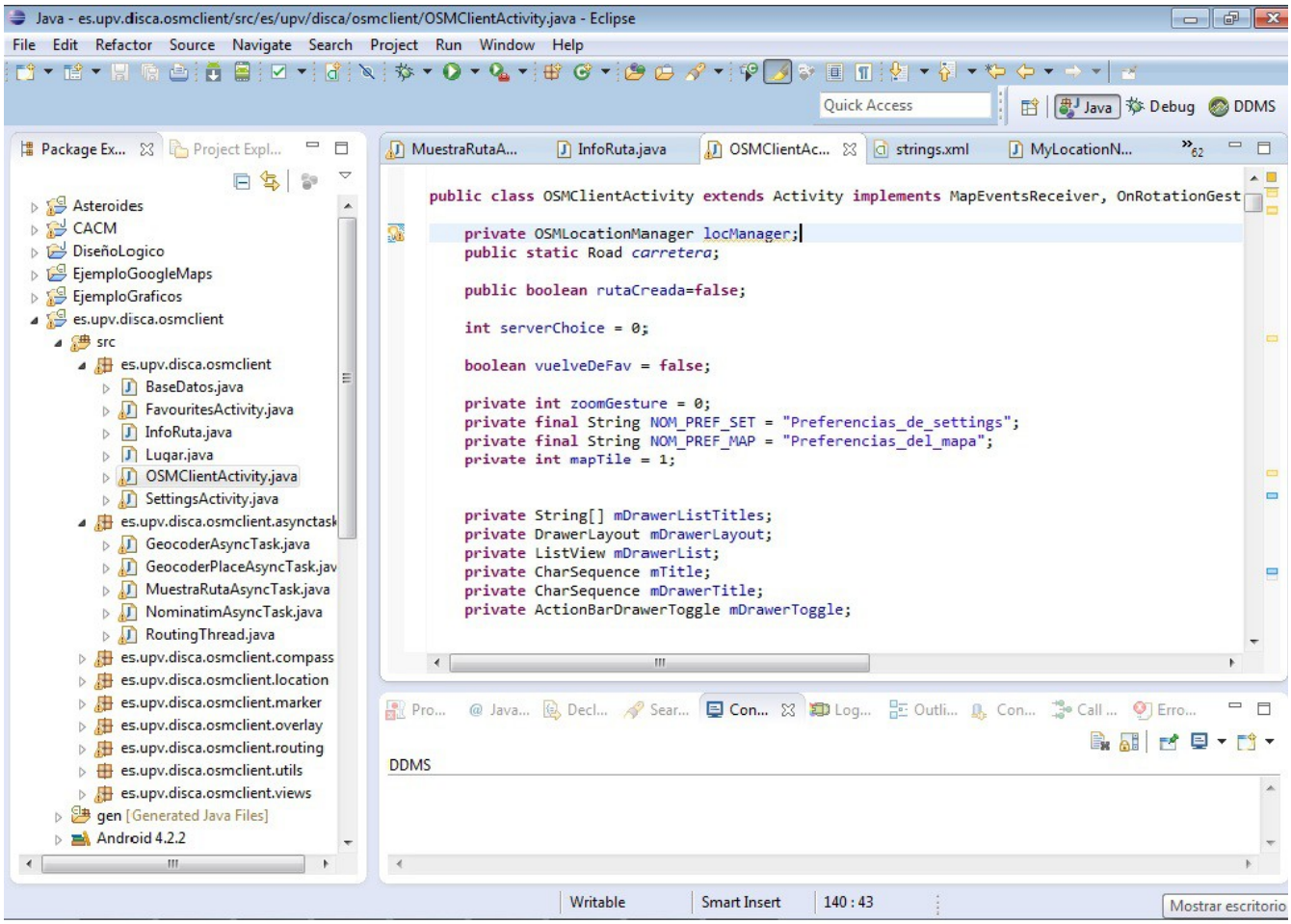

Figura 3.1.b. Captura de pantalla del proyecto realizado en Eclipse.

Al lado de este botón se encuentra el botón de ejecución en modo depuración, para depurar los programas. La depuración admite el uso de

breakpoints o puntos de ruptura en el código (Para consultar el estado de las variables, los hilos y la pila en ese punto del código).

3.2.JAVA

Este es el lenguaje de programación que he utilizado para el proyecto, dado que es el único con el que se puede programar en Android (C++ se utiliza para programar en código nativo). La versión que he utilizado es la JRE 7 update 60.

3.2.a)Descripción

En 1991 James Gosling, Patrick Naughton, Chris Warth, Ed Frank y Mike Sheridan de Sun Microsystems, Inc. concibieron el lenguaje de programación Java. El primer impulso para la creación de Java fue la necesidad de crear un lenguaje independiente de la plataforma, un lenguaje que pudiera utilizarse para crear software para distintos dispositivos. Otro factor que propició su aparición fue la Web, pues a esta se conectan un gran número de dispositivos distintos y por ello había una necesidad imperante de utilizar un lenguaje portable; cuyos programas se pudiesen ejecutar en distintas arquitecturas.

La clave que permite a Java ser un lenguaje portable es que la salida de este no es un compilador ejecutable, sino un *bytecode*. El bytecode es un conjunto de instrucciones altamente optimizado que se ha diseñado para ser ejecutado por el intérprete de Java, Java Virtual Machine (JVM, Máquina Virtual de Java); es decir, el intérprete de Java es un intérprete del *bytecode*.

A pesar de que, en principio, Java se diseño como un lenguaje que había de ser interpretado, es posible la compilar el *bytecode* generado en código nativo. La empresa Sun suministra un compilador JIT (Just In Time) para el *bytecode[7]*.

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo

que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

El lenguaje Java se creó con cinco objetivos principales:

•Debería usar el paradigma de la programación orientada a objetos.

•Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.

•Debería incluir por defecto soporte para trabajo en red.

•Debería diseñarse para ejecutar código en sistemas remotos de forma segura.

•Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++[8].

3.2.b)¿Por qué los desarrolladores de software eligen Java?

Java ha sido probado, ajustado, ampliado y probado por toda una comunidad de desarrolladores, arquitectos de aplicaciones y entusiastas de Java. Java está diseñado para permitir el desarrollo de aplicaciones portátiles de elevado rendimiento para el más amplio rango de plataformas informáticas posible. Al poner a disposición de todo el mundo aplicaciones en entornos heterogéneos, las empresas pueden proporcionar más servicios y mejorar la productividad, las comunicaciones y colaboración del usuario final y reducir drásticamente el costo de propiedad tanto para aplicaciones de usuario como de empresa. Java se ha convertido en un valor impagable para los desarrolladores, ya que les permite:

•Escribir software en una plataforma y ejecutarla virtualmente en otra.

•Crear programas que se puedan ejecutar en un explorador y acceder a servicios Web disponibles.

•Desarrollar aplicaciones de servidor para foros en línea, almacenes, encuestas, procesamiento de formularios HTML y mucho más.

•Combinar aplicaciones o servicios que utilizan el lenguaje Java para crear aplicaciones o servicios con un gran nivel de personalización.

•Escribir aplicaciones potentes y eficaces para teléfonos móviles, procesadores remotos, microcontroladores, módulos inalámbricos, sensores, gateways, productos de consumo y prácticamente cualquier otro dispositivo electrónico[7].

3.3.Android SDK

El SDK (Software Development Kit) de Android, incluye un conjunto de herramientas de desarrollo: Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. Las plataformas de desarrollo soportadas incluyen Linux, Mac OS X 10.4.9 o posterior, y Windows XP o posterior. El IDE que soporta oficialmente es Eclipse junto con el complemento ADT (Android Development Tools plugin), aunque últimamente están creciendo el número de usuarios del IDE de Android Studio, un IDE diseñado específicamente para Android.

Las Actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad [9].

Un proyecto creado para Android difiere en el resto de proyectos java en el sentido de que se generan automáticamente unas clases adicionales dentro de la carpeta gen del proyecto. Estas clases se llaman BuildConfig y R, la cual crea las referencias de todos los recursos del proyecto. Estos recursos pueden ser de tipo Drawable (Imagenes, vistas personalizadas, etc.), Layout (Encargados de generar la interfaz de usuario de las actividades, fragments y vistas personalizadas del proyecto), String (Como en Java, para almacenar cadenas de texto, lo cual facilita la traducción de las cadenas), array (contenedor de Strings), menus, entre otros.

El SDK de Android tiene un emulador incluido denominado AVD Manager (Android Virtual Device Manager), en el que se pueden emular tanto dispositivos reales seleccionados de una amplia lista, como dispositivos personalizados por el usuario. Este emulador, a pesar de tener un arranque

inicial indeseablemente lento, cuenta con las funciones propias de cada dispositivo, por lo que es de especial utilidad para comprobar como se comportan las aplicaciones desarrolladas en *smartphones* o *tablets* con distintos niveles de resolución y tamaños de pantalla.

En el desarrollo de la aplicación, además de dispositivos físicos he utilizado distintos dispositivos emulados tanto con el emulador nativo del SDK como el emulador Genymotion. Al contrario que el AVD Manager, este emulador solamente es gratuito para particulares y no para empresas. Aún así este emulador tiene mejoras con respecto al anterior en el sentido del rendimiento y en la posibilidad de seleccionar posiciones de GPS directamente desde el mapa que acompaña al emulador (En el AVD se deben enviar por terminal, lo cual resulta más engorroso).

Figura 3.3.i. Captura del emulador AVD Manager en funcionamiento.

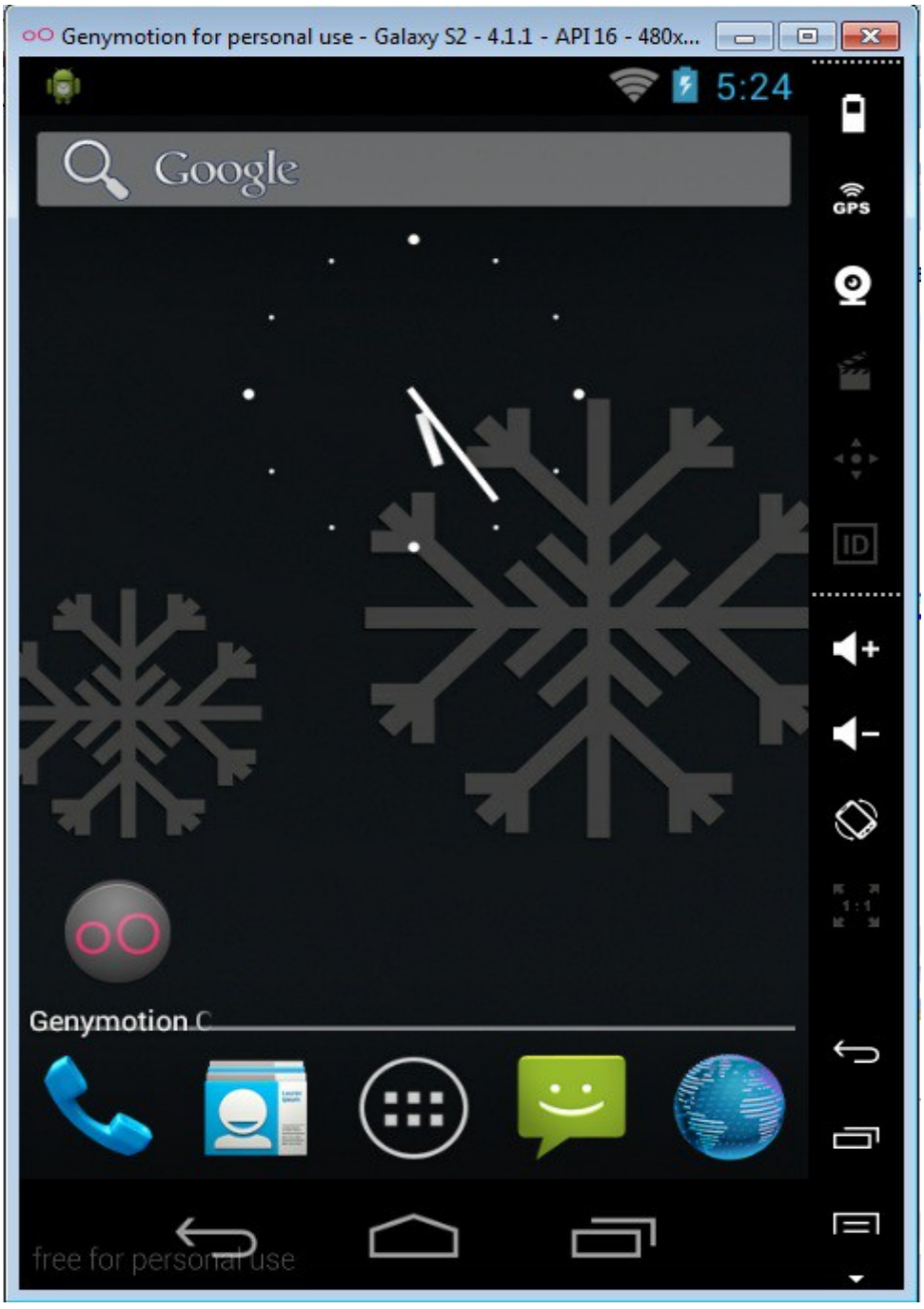

Figura 3.3.ii. Captura del emulador Genymotion en funcionamiento.

4.Librerías utilizadas

Principalmente he utilizado tres librerías (Exceptuando las de JAVA) en el desarrollo del proyecto, las librerías de Android y las del proyecto OSMDroid y su extensión OSMBonusPack. Tan solo explicaré las clases de Android de las que me he hecho servir en el desarrollo del proyecto y las clases que me he obligado a modificar de las demás librerías. Las modificaciones las explicaré en mayor profundidad en el capítulo posterior.

4.1.Android API 17

Antes de comentar las clases pertenecientes a esta librería, debo aclarar algunos conceptos sobre el sistema operativo de Android. Las aplicaciones ejecutadas sobre Android están compuestas visualmente por actividades, fragments y vistas. Las actividades o activities se podrían comparar con las ventanas de otros sistemas operativos como Windows y siempre están asociadas a un layout (Contenedor). Dentro de estas actividades se situarían los fragments y las vistas.

Los fragments se tratan de trozos de la interfaz de usuario que se introdujeron en la version 3.0 de Android, junto con el lanzamiento de las tablets [10]. Una de las razones por las que se crearon fragments fue para solventar los problemas con la reutilización de código y las diferencias entre los tamaños de distintos dispositivos sobre los que se ejecutase la misma aplicación.

Si las actividades y los fragments son el contenedor, las vistas o views serían el contenido. Representan todos los elementos visuales (Botones, cajas de texto...) que se ven en la pantalla de la aplicación.

El ciclo de vida de las aplicaciones en Android es: aplicación activa (visible e interactiva), visible (visible pero no interactiva), pausada (no visible) y destruida. Además de los elementos visuales, las aplicaciones de Android poseen procesos en segundo plano ejecutados sobre hilos y servicios.

4.1.a)Activity

Las actividades descienden todas de la clase Activity y tienen un layout asociado. En este layout, generado por un archivo xml definido previamente o por código java, se introducen todos los elementos visuales deseados por el desarrollador.

El ciclo de vida de las Activities pasa por la ejecución de los siguientes métodos de la clase Activity: onCreate(), onStart(), onResume(), onRestart, onPause(), onStop(), onRestart() y onDestroy(). De estos métodos los únicos que pueden no ejecutarse son los métodos onStop(), onRestart() y onDestroy(), puesto que el sistema operativo puede decidir eliminar la aplicación de memoria para hacer hueco a otras más importantes.

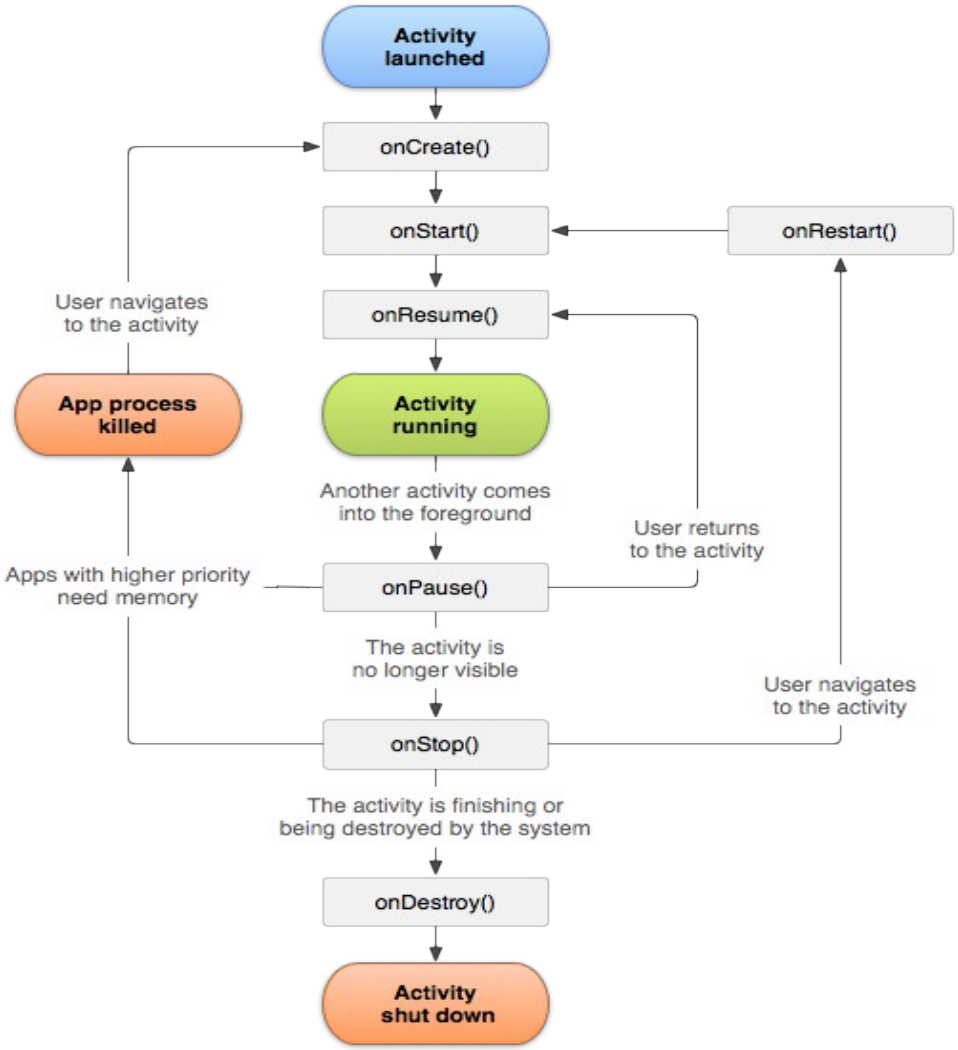

Figura 4.1.a. Ciclo de vida de una actividad en Android [11].

4.1.b)View

Esta clase representa el bloque básico para los componentes de la interfaz de usuario. Una vista ocupa un área rectangular en la pantalla y es la responsable del dibujo y del manejo de eventos. Es la clase básica para los widgets, los cuales son usados para crear componentes interactivos de la IU (interfaz de usuario).

La subclase ViewGroup es la clase básica de los layouts, los cuales son, como ya he mencionado antes, contenedores invisibles que contienen a otras vistas (u otros layouts) y definen sus propiedades de layout (orientación dentro del layout, posición relativa, etc...)[13].

4.1.c)Intent

Un Intent, como su nombre indica, es la intención de llevar una acción a cabo. Puede ser utilizado para lanzar una actividad, un servicio o para comunicarse con un servicio en segundo plano. Es básicamente una estructura de datos pasiva conteniendo una descripción abstracta de una acción a realizar[14].

4.1.d)Toast

Un Toast es una vista que contiene un pequeño y rápido mensaje para el usuario de la aplicación. Esta vista aparece como un mensaje en la parte inferior de la pantalla por defecto y sirve para avisar al usuario de algún problema ocurrido o simplemente informarle de algún evento ocurrido. Por ejemplo se podría programar que si el usuario no esta conectado a la red al apretar el botón para realizar una búsqueda en internet, le aparezca un Toast con el mensaje de "Conectate a la red"[15].

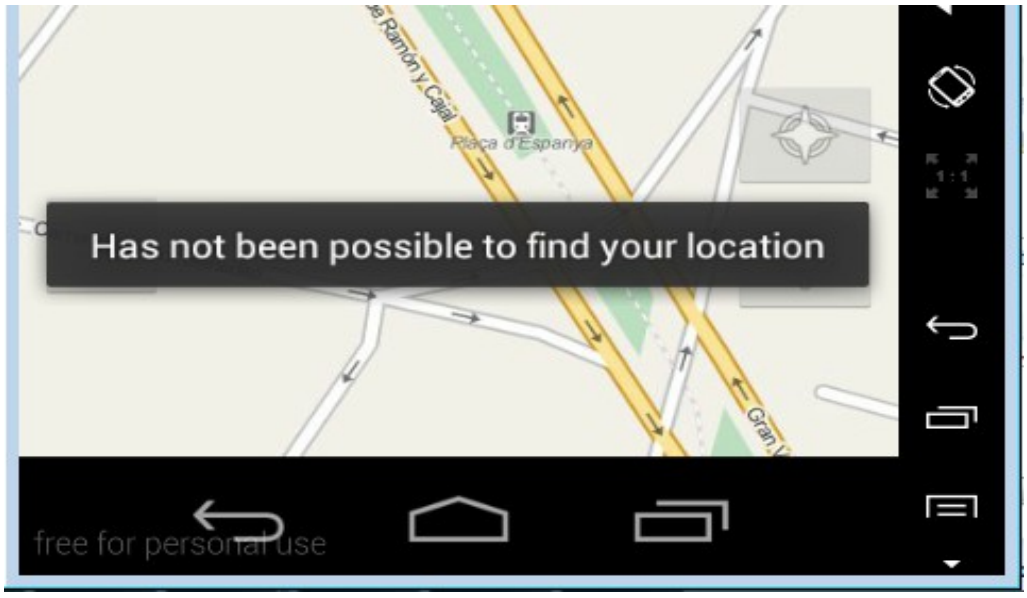

Figura 4.1.d. Ejemplo de Toast.

4.1.e)NotificationManager

Esta clase sirve para notificar al usuario eventos que han ocurrido. Las notificaciones pueden estar formadas por un icono colocado en la status bar (barra de estado) y es accesible a traves del launcher, encendiendo y apagando los LEDs del dispositivo o alertando al usuario mediante sonidos, vibraciones, etc[16].

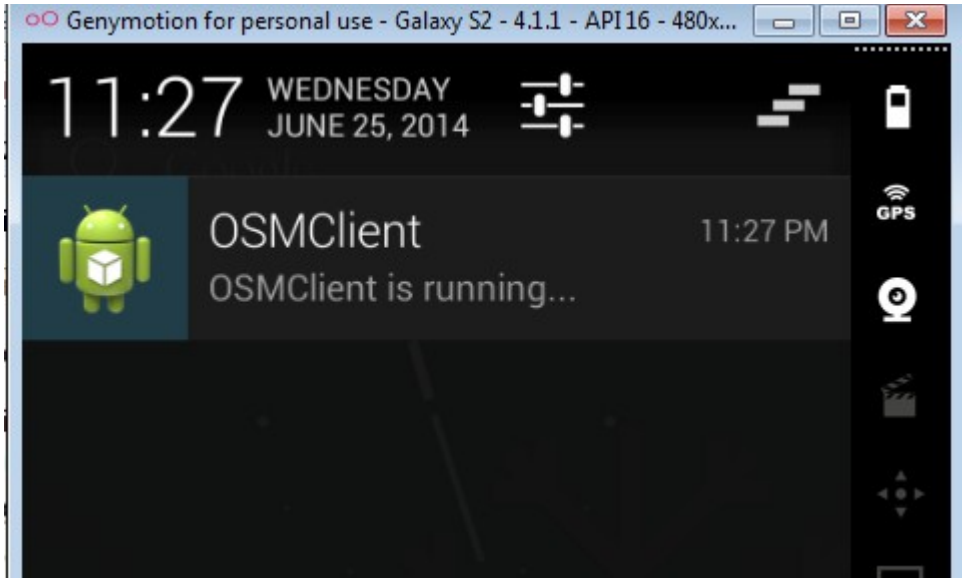

Figura 4.1.e. Ejemplo de notificación.

4.1.f)SharedPreferences

Las SharedPreferences aportan un método para almacenar preferencias y datos de la aplicación de una forma sencilla y segura. Esta clase permite almacenar los datos en forma de par clave-valor, y admite los tipos primitivos de Java. Es de gran utilidad a la hora de guardar las preferencias de configuración de la aplicación o cualquier valor que no necesite el uso de una base de datos. Las preferencias deben guardarse en el método onPause() de las Activities para asegurar su almacenamiento y la recuperación debe realizarse en el método onResume() pues es el que se ejecuta a continuación del anterior.

4.1.g)AsyncTask

Las AsyncTask o tareas asíncronas son tareas realizadas en segundo plano. Se caracterizan por tener tres métodos que se deben implementar: onPreExecute(), doInBackground(parametros) y onPostExecute(). En el método doInBackground es donde se realiza el trabajo en segundo plano y puede retornar un valor para el siguiente método, onPostExecute(). los métodos onPreExecute() (Se ejecuta antes que el doInBackground) y onPostExecute() se ejecutan en el hilo principal, por lo que los cambios que afecten a la IU se deben hacer en estos métodos.

Dado que el único hilo que puede alterar la IU es el hilo principal, los propios desarrolladores de Android recomiendan usar esta clase para los trabajos en segundo plano siempre que sea posible y mientras la tarea a realizar no sea temporalmente costosa. Las clases que extiendan de esta deben implementarse de este modo[17]:

public class EjemploAsyncTask extends AsyncTask<'tipo de los parametros', 'tipo de las actualizaciones en el hilo principal' (usado para las barras de progreso), 'tipo del valor devuelto por el doInBackground'>{

public EjemploAsyncTask(...){}

protected 'tipo del valor devuelto' doInBackground('tipo de los parámetros'){}

}

4.1.h)SQLiteOpenHelper

Clase de ayuda para la creación de bases de datos en Android. Esta clase incorpora funciones para actualizar la versión de la base de datos y abrir la base de datos en modo lectura (Para realizar consultas) y escritura (Para insertar, modificar o eliminar filas)[18].

4.1.i)Dialog

Los diálogos son pequeñas ventanas que sirven para alertar al usuario de una acción que debe tomar, como introducir información adicional o seleccionar una opción. A diferencia de las actividades, los diálogos tan solo cubren una porción de la pantalla[19].

4.2.OSMDroid versión 4.2

El proyecto OSMDroid se desarrolló para crear una alternativa gratuita a la clase MapView (API 1) de Android. El defecto de esta clase es que tan solo puede usar los mapas de Google, al igual que solo puede hacer peticiones de rutas al servidor de Google. Además, Google no ofrece este servicio de forma gratuita ni libre (Como expliqué en el capítulo 1.3).

La ventaja de OSMDroid es que no solo solventa estos problemas, sino que además nos permite trabajar con mapas offline, es decir, en lugar de ir descargando los mapas mientras se navega, se pueden recuperar de una copia almacenada en el dispositivo (Hay que tener cuidado con esto pues algunos proveedores de mapas lo consideran como una infracción de copyright)[20].

4.2.a)MapView

Aunque no he modificado esta clase propia de la librería debo explicarla puesto que es la encargada de crear la vista del mapa. Esta clase permite seleccionar el proveedor de mapas deseado (Con el método

setTileSource()), obtener las capas u overlays del mapa (polilineas dibujadas, marcadores introducidos, listeners (escuchadores de eventos) entre otros).

También posee un controlador de tipo MapController para realizar operaciones como cambiar el zoom, situar el centro del mapa en un punto específico, desplazar el mapa hacia una posición en concreto, etc.

Por otra parte, la implementación incluida en la librería de esta clase modifica el listener de deslizar el dedo sobre el mapa para desplazarlo en la dirección opuesta, al igual que modifica la acción del "doble click" del dedo sobre el mapa para ampliarlo cuando este evento ocurra.

4.2.b)MyMarker (modificada a partir de Marker)

Clase modificada a partir de la clase Marker. Esta clase permite situar marcadores en el mapa y sus funciones se amplían con la clase InfoWindow de la librería OSMBonusPack.

4.2.c)GeoPoint

Este objeto esta compuesto por una latitud y una longitud. Representa unas coordenadas reales sobre el mapa, por lo que es el objeto utilizado en el momento de desplazarse hacia lugares concretos o situar marcadores y otros elementos en el mapa.

4.2.d)MyLocationNewOverlay

Esta clase es la encargada de obtener tanto la ubicación actual del usuario como su representación en forma de persona o flecha sobre el mapa. Si el dispositivo posee una orientación (ya sea mediante la brújula interna o mediante el trazado de sucesivas posiciones consecutivas) aparecerá una flecha en la dirección que se halle el dispositivo respecto del mapa. Si por el contrario no hay modo de saber esta orientación, simplemente aparecerá el dibujo de una persona para señalizar la posición.

En esta clase también se lleva a cabo el cambio de orientación del mapa (mapa, que no flecha) en el modo de navegación, el cual explicaré en el capitulo de "funciones de la aplicación". Además de la orientación, esta clase nos permite saber la velocidad a la que nos estamos desplazando, la precisión de nuestra localización actual y métodos para activar o desactivar el seguimiento automático de nuestra posición (Que el mapa se desplace a nuestro paso).

4.3.OSMBonusPack versión 4.5

Esta clase es una extensión de la anterior y utiliza muchas de las clases de la librería OSMDroid por lo que no se puede utilizar sin una versión actualizada de esta. Las características más importantes de esta librería son que incorpora clases y métodos para el trazado de rutas, además de mejoras en el sistema de marcadores de OSMDroid[21].

4.3.a)MyMarkerInfoWindow(modificada a partir de MarkerInfoWindow)

Clase creada para ampliar las características de MyMarker. Al pulsar en el marcador abre una pequeña ventana informativa que varía dependiendo del layout pasado como referencia. Esta clase hereda de InfoWindow por lo que debe implementar el método onOpen(), el cual realiza las acciones necesarias cuando se abre la ventana. Del mismo modo se debe implementar el método onClose() que se activa cuando se cierra la ventana.

4.3.b)GeocoderNominatim

Esta clase es equivalente a la clase Geocoder de Android. En lugar de utilizar el servidor de Google, utiliza el servidor nominatim de OpenStreetMap u otro especificado por el desarrollador. Nominatim (Del latín, 'por nombre') es una herramienta para buscar datos de un servidor por nombre y dirección y para generar direcciones sintéticas de puntos OSM (OpenStreetMap) (Geocoding reverso). En otras palabras, se trata de una herramienta para hallar un punto geográfico a partir de un nombre de calle, pueblo, país, lugar, etc. [22]

4.3.c)OSRMRoadManager

Esta clase es la encargada de obtener la ruta del servidor de OSM o del servidor al que esta orientado la aplicación. La ruta es devuelta en forma de indicaciones previamente definidas como "ve a la calle 'x' o gira a la

derecha". Para definir la ruta utiliza objetos de tipo Road, RoadNode y RoadLeg, los cuales se unen para construir la ruta.

5.Análisis

5.1.Requisitos

La aplicación debe permitir realizar las siguientes acciones:

-Ampliar, alejar, rotar y desplazarse por el mapa.

-Situar marcadores sobre el mapa y cambiar el tipo de estos.

-Mostrar la información de un lugar especifico en el mapa (Por medio de los marcadores).

-Trazar una ruta entre un origen y un destino.

-Utilizar la función de búsqueda para encontrar un lugar mediante su nombre.

-Activar el modo de navegación para que la aplicación muestre al usuario automáticamente la acción a realizar en cada punto de la ruta.

-Almacenar los lugares y rutas favoritos para su posterior utilización.

De estas funciones hay una que es crítica para el proyecto y que debe implementarse a toda costa: La función para trazar una ruta y poder navegar por ella.

5.2.Diagramas de flujo

En los siguientes diagramas de flujo se explica conceptualmente como se pueden llevar a cabo las funciones citadas anteriormente. El circulo negro simboliza el estado inicial, las flechas representan transiciones, las elipses representan estados, los rombos representan una decisión a tomar y un circulo negro dentro de otro circulo simboliza un estado final.

5.2.a)Situar marcadores en el mapa, modificarlos y añadir lugares a favoritos

El usuario al hacer una pulsación larga sobre el mapa crea un marcador vacio sobre el lugar pulsado. Si acto seguido hace un clic sobre este marcador, abre la ventana de información del marcador, la cual le permite cambiar el tipo de marcador por el de destino, origen, o punto de ruta. Si por el contrario pulsa en el botón de acciones extra, se abre el dialogo de acciones extra para situar el marcador como "hogar" o añadirlo a favoritos.

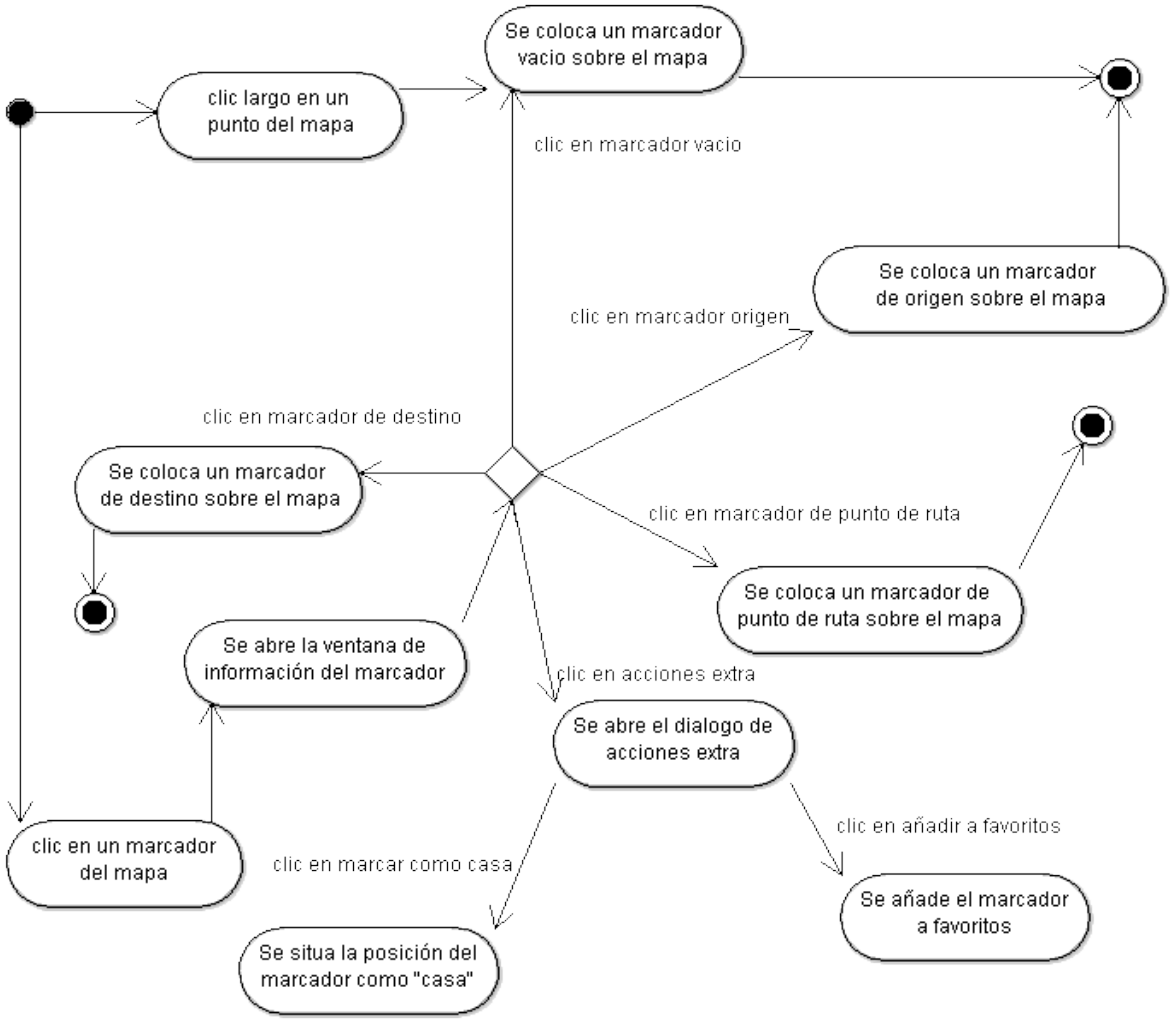

Figura 5.2.a. Diagrama de flujo para añadir marcadores, modificarlos y añadir lugares a favoritos.

5.2.b)Funciones de zoom y rotación

Si el usuario desliza el dedo por el mapa, este se desplaza; si hace un gesto de pinza acercando los dedos este se aleja y del modo opuesto se acerca. Si por el contrario rota los dedos por el mapa, este se girará en el sentido de la rotación.

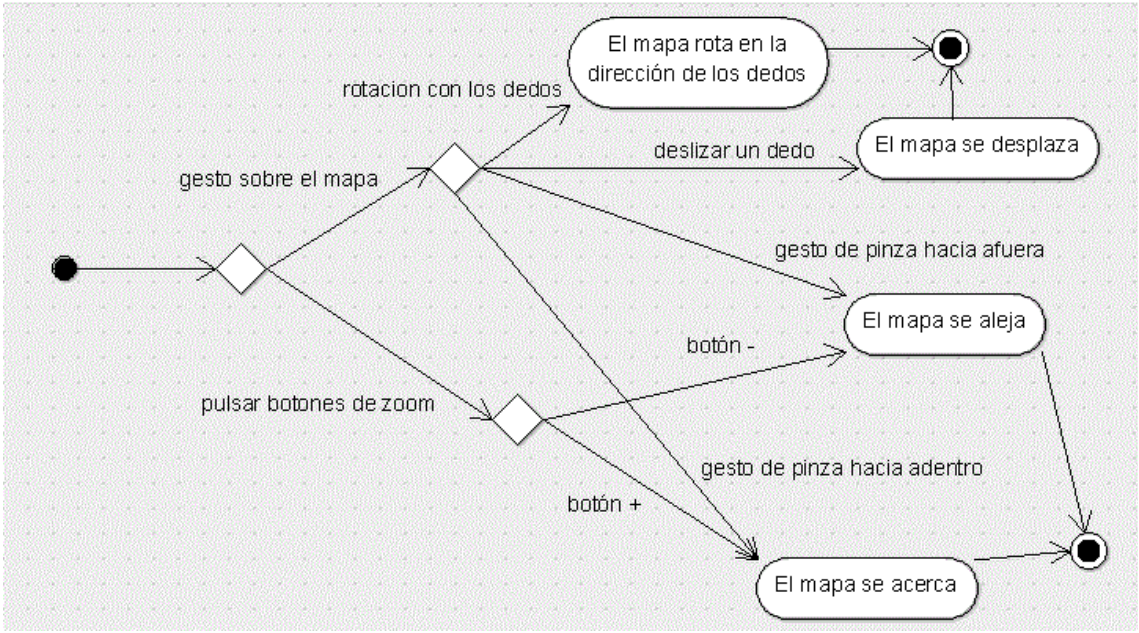

Figura 5.2.b. Diagrama de flujo de las funciones de zoom y rotación del mapa.

5.2.c)Trazar/eliminar ruta

Si no hay una ruta trazada sobre el mapa, se puede hacer clic en el botón de trazar ruta para crearla, a menos de que no se disponga de conexión a Internet o no se pueda trazar la ruta. Si se traza satisfactoriamente la ruta, se generarán una serie de nodos unidos por una linea. Si se hace clic sobre estos nodos se abrirá su ventana de información. Por el contrario, si se pulsa en el botón de borrar ruta, esta se eliminará del mapa.

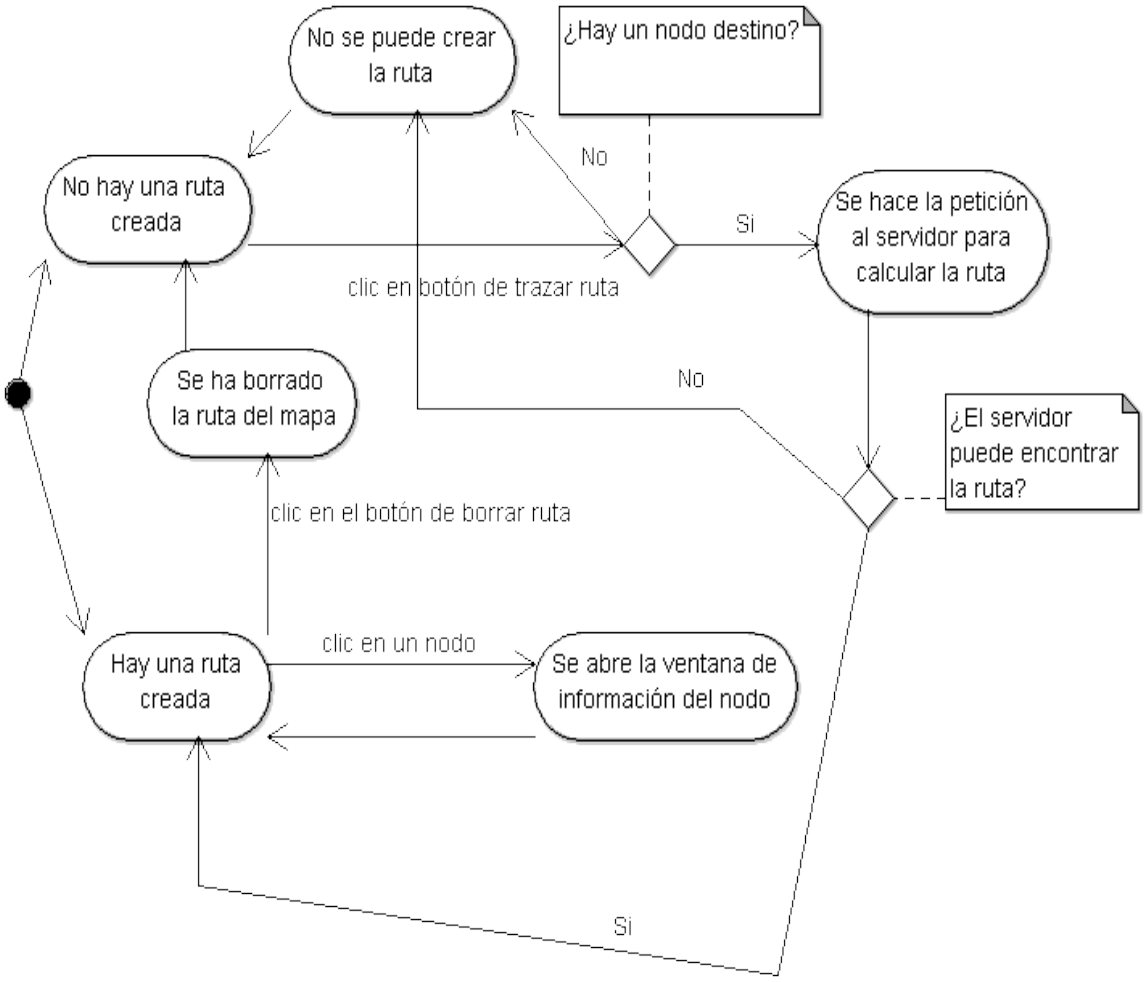

Figura 5.2.c. Diagrama de flujo de las funciones de creación y eliminación de la ruta.

5.2.d)Navegación y lista de rutas

Si hay una ruta creada, se puede hacer clic en el botón del modo de navegación para comenzarla. En el caso de que el gps no funcione, se haya llegado al destino o el usuario interaccione con el mapa (Tocándolo, desplazándolo...) el modo de navegación terminará. Si se hace clic en el botón de borrar ruta, esta se eliminará y el usuario abandonará el modo de navegación. Por otra parte si el usuario hace clic en el layout de información de ruta, se abrirá un dialogo con la lista de instrucciones, un campo de texto y un botón para añadir la ruta a favoritos.

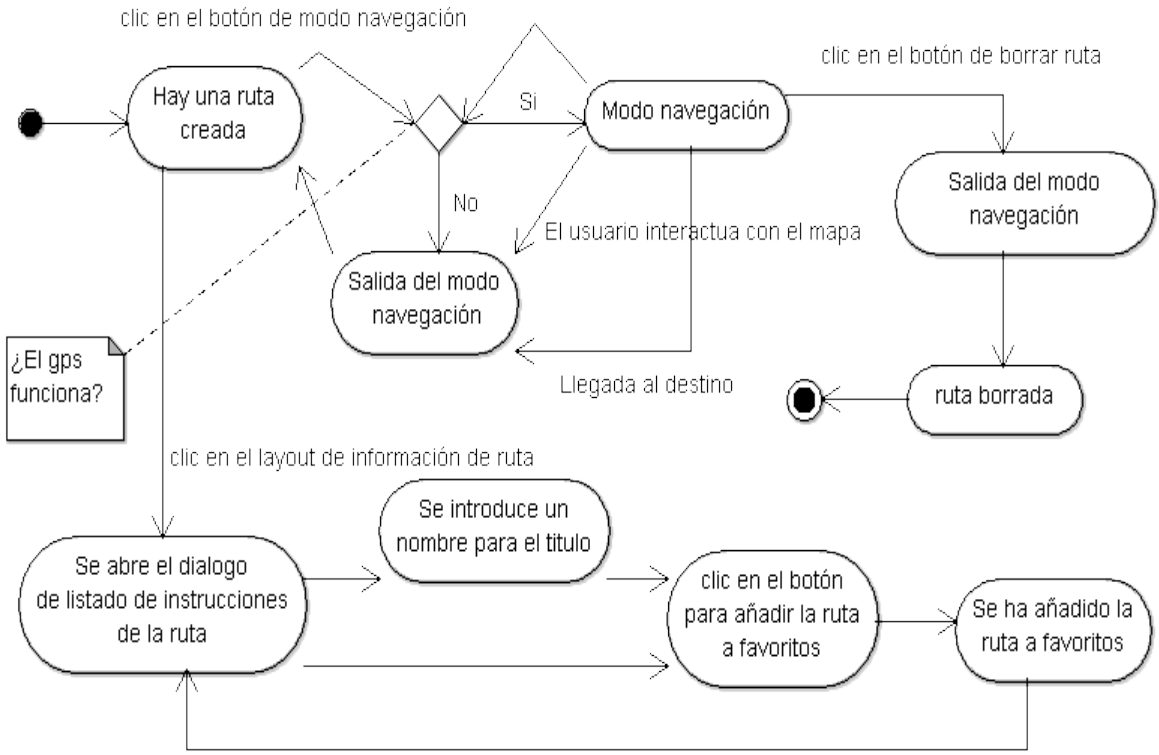

Figura 5.2.d. Diagrama de flujo del modo de navegación y de como se añaden las rutas a favoritos.

5.2.e)Búsqueda de un lugar

Al hacer clic en el botón de búsqueda se abré un campo de texto para introducir el lugar a buscar. Al pulsar en el botón de 'Ok', si el usuario no esta conectado a Internet se muestra un mensaje. En caso contrario se inicia la búsqueda y si se encuentra un resultado se abre un dialogo con la lista de resultados (En caso contrario se muestra un mensaje). Si se hace clic en uno de estos resultados, el sistema creará un marcador en el lugar seleccionado.

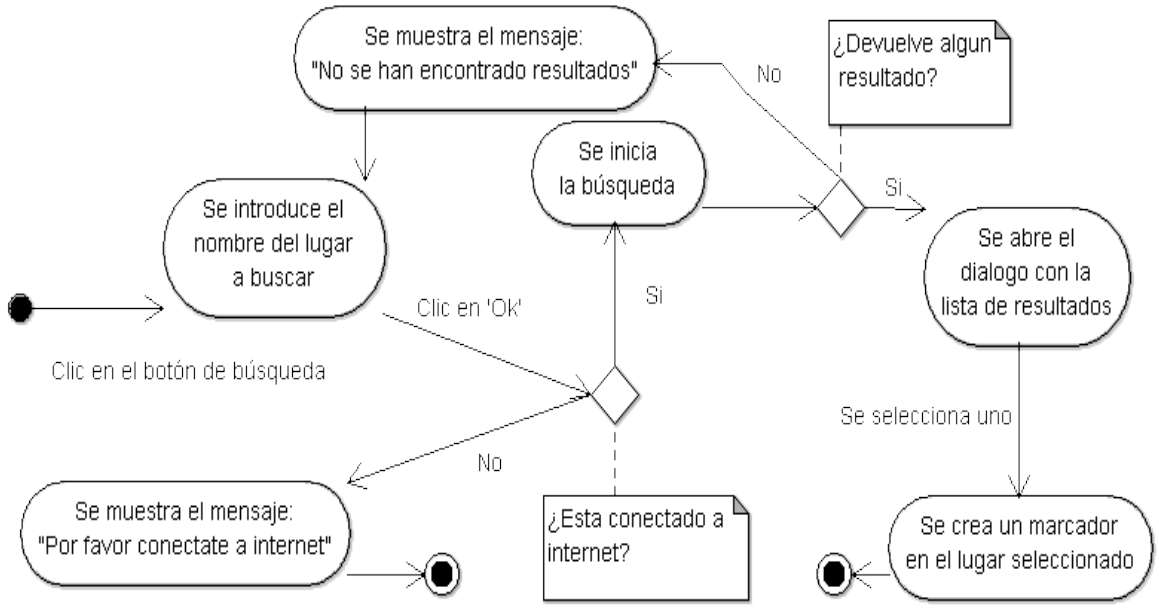

Figura 5.2.e. Diagrama de flujo de la función de búsqueda.

5.3.Diseño de la base de datos

La base de datos debe permitir almacenar y recuperar los lugares y rutas favoritos. Por ello he creado tres tablas: Lugar, Ruta y Waypoint. La tabla Lugar almacena los datos de un punto geográfico específico. La tabla Ruta almacena el nombre, titulo e identificador de la ruta; este identificador es usado como clave ajena por la tabla Waypoint, donde se almacena la localización geográfica de los distintos puntos de ruta para la ruta con la que comparte el identificador.

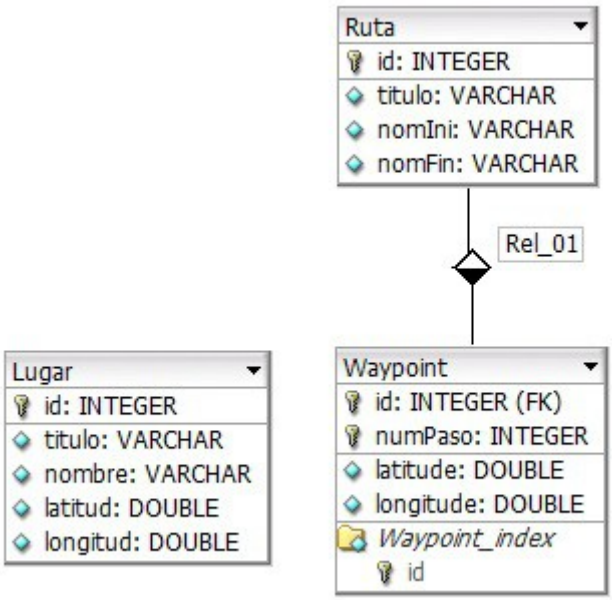

Figura 5.3. Modelo conceptual de la base de datos.

6.Estructura y codificación

La estructura del proyecto se puede observar en la siguiente figura:

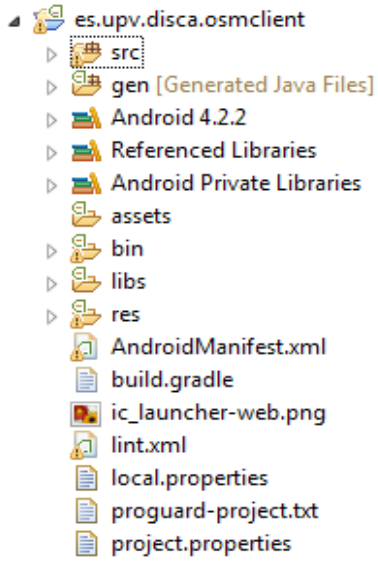

Figura 6. Estructura del proyecto.

6.1.Paquetes de Java

La ruta que he seguido para los archivos de código fuente es: es.upv.disca.osmclient, como se puede apreciar en la siguiente figura:

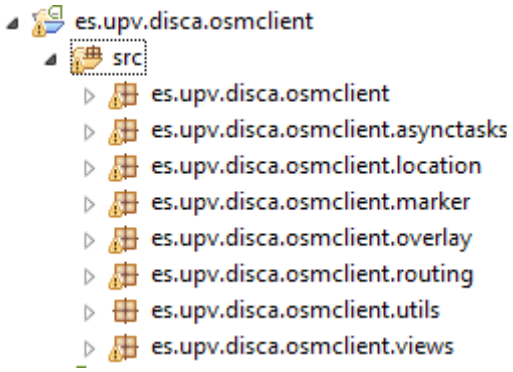

Figura 6.1. Distribución de los paquetes de java.

6.1.a)es.upv.disca.osmclient

Aqui guardo las clases principales como las que descienden de Activity y otras auxiliares.

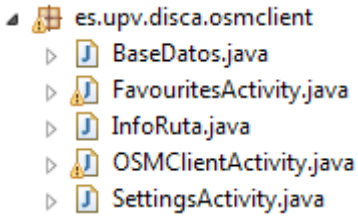

Figura 6.1.a. Clases principales del proyecto.

•*BaseDatos*

clase que extiende de SQLiteOpenHelper y me permite crear la base de datos. La base de datos se denomina "Favoritos.db" y las tablas creadas son: Ruta, Lugar y Waypoint.

Name	Type
Tables (4)	
Lugar	
id	INTEGERAUTO_INCREMENT
titulo	TEXT
nombre	TEXT
latitud	DOUBLE
longitud	DOUBLE
Ruta	
id	INTEGER
titulo	TEXT
nomIni	TEXT
nomFin	TEXT
Waypoint	
id	INTEGER
numPaso	INTEGER
latitude	DOUBLE
longitude	DOUBLE
android_metadata	
locale	TEXT

Figura 6.1.a.i. Estructura de la base de datos.

La tabla android_metadata sirve para almacenar el lenguaje y región geográfica del dispositivo.

•*FavoritosActivity*

Esta clase tiene asociado el layout "favourites_layout". Posee dos tablas para mostrar los lugares y rutas guardados en la base de datos.

•*InfoRuta*

Tupla para almacenar el titulo, snippet, subdescripción yla imagen de las instrucciones de ruta.

•OSMClientActivity

Esta es la actividad principal del proyecto y la primera en iniciarse junto con la aplicación. Su layout asociado es "main" e implementa las clases MapEventsReceiver y OnRotationGestureListener. En el método onCreate() se inicializan las vistas del layout incluida la del mapa. También se inicializa el controlador del mapa, el proveedor de localización, el Navigation Drawer (Menu deslizable), el NotificationManager y la ActionBar (barra de acción).

En el método onDestroy() se cancela la instancia del NotificationManager. En el onResume() se recuperan las preferencias almacenadas, se centra el mapa en la posición que estaba antes de ponerse la actividad en pausa y se vuelve a activar la localización. En el onPause() por el contrario, se guardan las preferencias y se desactiva la localización y en el onStop() se lanza una notificación.

El método recuperarPreferencias() recupera las preferencias almacenadas en los ficheros "preferencias_de_settings" y "preferencias_del_mapa". Del primero se recuperan las opciones predefinidas por el usuario como si desea activar o desactivar el zoom, mientras que del segundo se rescatan las preferencias referentes al tipo de mapa en uso o el punto actual del mapa en el que se esta situado.

El método onCreateOptionsmenu(Menu menu) se extiende de Activity y se activa cuando se crea el menú de la aplicación. En este caso carga el menú "osmclient". por otra parte el onPrepareOptionsMenu(Menu menu) se activa cuando se realiza la llamada a invalidateOptionsMenu(). A su vez, el onOptionsItemSelected(MenuItem item) realiza una acción en función del ítem seleccionado en el menu.

El método selectItem(int position) es parecido al anterior pero aplicado al menú del Navigation Drawer. El onActivityResult(int request, int result, Intent data) se ejecuta cuando otra Activity retorna un valor a esta (Útil para la comunicacion entre actividades).

El método onTouchEvent(MotionEvent event) implementa el listener encargado de realizar la función de zoom del mapa cuando

se hace pinza con los dedos sobre la pantalla. Los métodos singleTapConfirmedHelper(GeoPoint arg0) y longPressHelper(GeoPoint loc) se encargan de implementar la interfaz de MapEventsReceiver para los gestos de hacer clic sobre el mapa y de mantener presionado en el mapa respectivamente. Para rotar el mapa con los dedos, se utiliza la implementación de la interfaz onRotationGestureListener de la clase RotationGestureDetector.

El método de showDialogButtonClick(final List<Address> address) se encarga de mostrar un dialogo que contiene una lista con las direcciones (Address) pasadas como argumento y permite seleccionar una de ellas para realizar una geolocalización reversa (Nominatim). El método generateNotification(Context context, String message) sirve para generar una notificación con el mensaje pasado como argumento.

El método limpiarMarcasMapa() borra todos los marcadores de tipo "Ninguno" del mapa, mientras que el método colocarNodo(GeoPoint loc) coloca un nodo/marcador en la posición "loc". Por otra parte, el método limpiarRutasMapa() borra tanto los marcadores como las rutas. El método encontrarWaypoints() devuelve una lista con los nodos de tipo "Waypoint" que hay sobre el mapa, los cuales se pueden usar en el método trazarRuta() para obtener la ruta entre un nodo origen y un nodo destino.

Por último el método calcularNuevaRuta() se utiliza en la navegación y sirve para calcular nuevas rutas en el caso de que el usuario se salga de la ruta establecida. El resto de métodos que no he comentado son o bien auxiliares, o bien getters y setters o métodos ligados a la función onclick() de los botones incluidos en el layout.

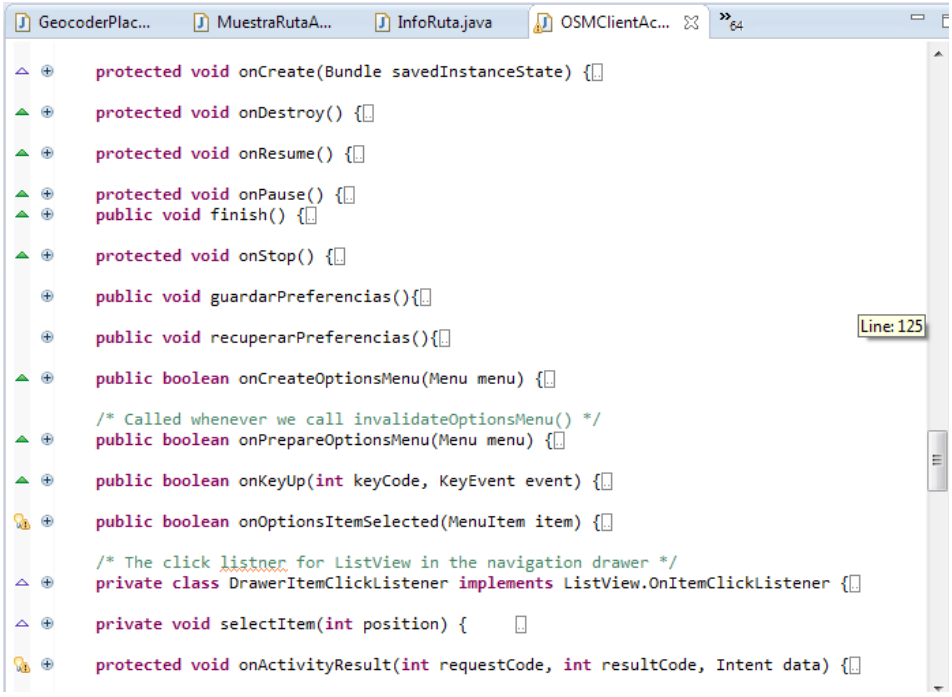

Figura 6.1.a.ii.1. Estructura de los métodos heredados de Activity en la clase OSMClient.

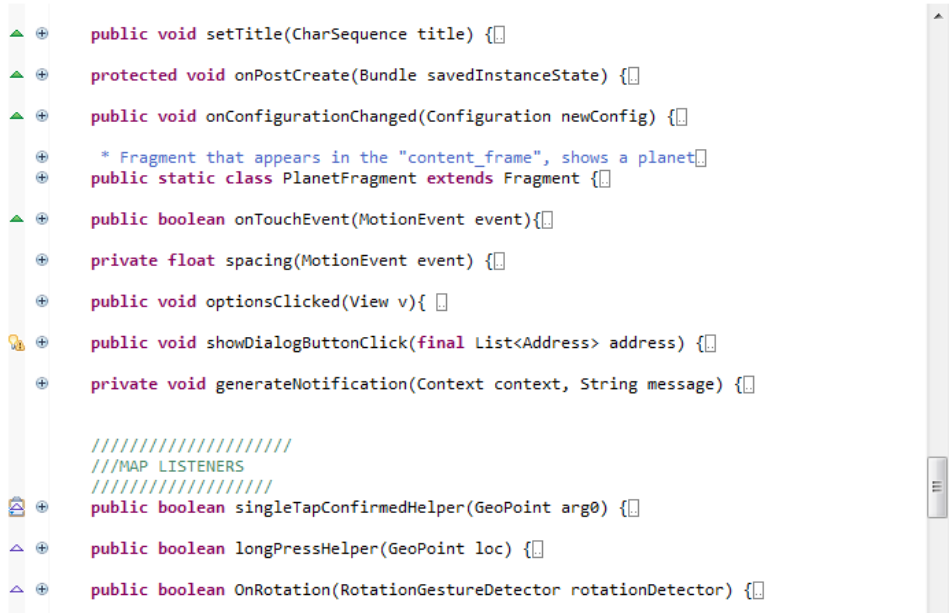

Figura 6.1.a.ii.2. Estructura de los métodos que implementan el Navigation Drawer y los *Listeners* del mapa de la clase OSMClient.

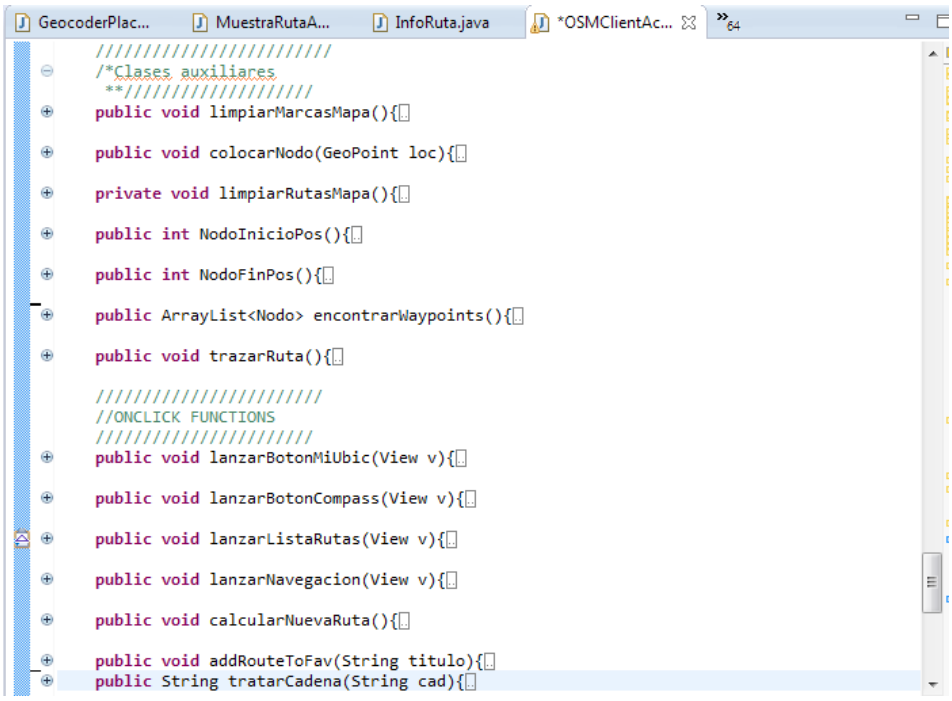

Figura 6.1.a.ii.3. Estructura de los métodos auxiliares y lanzadores de funciones de la clase OSMClient.

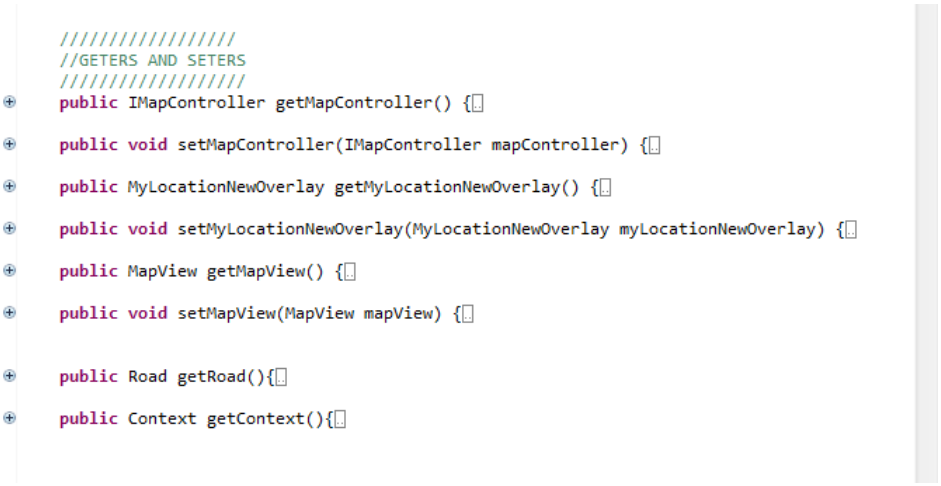

Figura 6.1.a.ii.4. Estructura de los métodos *getters* y *setters* de OSMClientActivity.

•*SettingsActivity*

Esta actividad esta asociada al layout "settings_lay" y es la encargada de almacenar las preferencias referentes a las acciones

de zoom y rotación del mapa. Las preferencias se almacenan en el método onPause() en el archivo "preferencias_de_settings" y se recuperan en el método onResume().

6.1.b)es.upv.disca.osmclient.asyncTasks

Este paquete almacena todas las clases que descienden de AsyncTask y de Thread. Es decir, son las clases que se encargan de ejecutar las tareas en segundo plano.

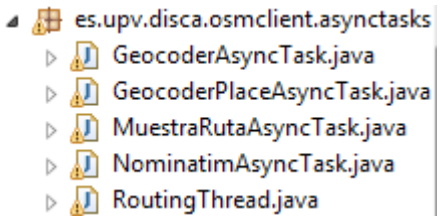

Figura 6.1.b. Clases del paquete asynctasks.

•*GeocoderAsyncTask*

Esta tarea se encarga de obtener la lista de direcciones en base al nombre de un lugar. Esto se realiza gracias al método getFromlocationName() de la clase GeocoderNominatim.

•*GeocoderPlaceAsyncTask*

Esta tarea se encarga de obtener el nombre de un lugar en base a su localización en el mapa. Se utiliza para conocer el nombre del lugar donde esta situado un marcador introducido por el usuario.

•*MuestraRutaAsyncTask*

Esta clase se encarga de crear una ruta sobre el mapa que pasa por los distintos waypoints o puntos de ruta elegidos por el usuario. La ruta creada esta formada por nodos que representan las distintas

instrucciones a realizar en la navegación como girar a la derecha, hacer un cambio de sentido, etc. La tarea se encarga de poner estos nodos sobre el mapa con sus respectivas ventanas de información donde aparece el dibujo de la acción a realizar, el número de la instrucción, la descripción en texto de la acción y la distancia a la que se encuentra la siguiente instrucción.

Además esta clase manda una señal a la actividad principal para que muestre un layout previamente oculto el cual da información sobre la ruta como el nombre del destino y la distancia desde el origen hasta este. En el modo de navegación aparece la información de la acción a tomar para el siguiente waypoint.

•NominatimAsyncTask

Tarea que se encarga de poner en el mapa un nodo con la información contenida en el objeto de tipo Address pasado como parámetro.

•RoutingThread

Esta es, junto con la actividad principal, una de las clases críticas del proyecto puesto que se encarga de la navegación por la ruta creada previamente. Al contrario que las anteriores, esta clase extiende de la clase Thread de Java y no de AsyncTask. Esto es debido a que las tareas asíncronas se utilizan para operaciones sencillas y cortas, mientras que los hilos se usan para las acciones complejas y duraderas.

La ejecución en segundo plano se ejecuta sobre el método run(). Al principio de la ejecución se impide que la pantalla del dispositivo se suspenda (Para que no se apague durante la navegación), acto seguido hay un bucle "while" el cual se ejecuta permanentemente a menos que se cumplan una serie de condiciones las cuales son: Que el usuario se aleje de la ruta establecida, que llegue a su destino, que siga en el modo navegación y que no haya alcanzado un nodo posterior al actual.

Dentro del while se actualiza la velocidad mostrada a la que se desplaza el dispositivo y se efectúa el algoritmo de navegación. El algoritmo funciona del siguiente modo: Primero comprueba si el indice de instrucciones actual no es igual al indice del nodo de destino de la ruta (El último de la lista de nodos), si es igual se considera que se ha llegado al destino y se termina la navegación, en caso contrario se comprueba si se ha llegado al nodo actual.

Si todavía no se ha llegado al nodo no ocurre nada, en caso contrario, si se esta alejando de este nodo y se esta acercando al siguiente se incrementa el indice de la lista de nodos para convertir al siguiente nodo en el actual y en caso contrario se traza una nueva ruta desde la posición actual hasta el destino.

Fuera del bucle while se hacen comprobaciones para saber cúal es la causa de que se halla salido del bucle. Si se ha salido de la ruta o se esta alejando del nodo actual, se calcula una nueva ruta, sino se cancela el modo de navegación.

Dado que en Android no se puede alterar la actividad que se esta ejecutando en primer plano desde un hilo que no sea el principal, se ha utilizado el método runOnUiThread(Runnable run) de la clase Android para actualizar las instrucciones en pantalla. El método seEstaAlejando(int index, double anterior) coge la información de la distancia a la que estamos del nodo actual y la distancia a la que estábamos en el punto anterior y si estamos a más distancia devuelve verdadero. El método crearSegmentos() sirve para crear los segmentos entre los distintos puntos de la ruta para utilizarlos posteriormente para saber si nos hemos alejado de la ruta.

6.1.c)es.upv.disca.osmclient.location

En este paquete se encuentran las clases referentes a la localización y la representación de esta sobre el mapa.

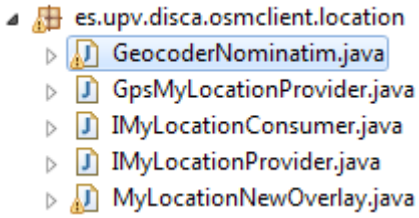

Figura 6.1.c. Estructura del paquete location.

•*GeocoderNominatim*

Esta clase la he explicado en el capítulo anterior y no hay nada más que añadir.

•*GPSMyLocationProvider*

Esta clase es idéntica a la que se encuentra en la librería osmdroid y se encarga de obtener actualizaciones de la posición mediante el GPS gracias a los métodos de la interfaz IMyLocationProvider.

•*MyLocationNewOverlay*

Esta es la clase más importante en cuanto a la localización se refiere, pues como he explicado antes, a partir de sus métodos se obtiene la mayoría de la información relacionada con la localización actual. La clase extiende de Overlay e implementa los métodos de IMyLocationConsumer, IOverlayMenuProvider y Snappable.

He tenido que modificarla para adaptarlo al modo de navegación, durante el cual la flecha no gira, sino que se mantiene siempre mirando arriba y es el mapa el que va girando. Para ello he tenido que modificar el método drawMyLocation(final Canvas canvas, final MapView mapView, final Location lastFix) que es donde se realiza el re-dibujo en función del cambio de localización.

6.1.d)es.upv.disca.osmclient.marker

En este paquete están las clases personalizadas de Marker y MarkerInfoWindow para distinguir entre distintos tipos de marcadores.

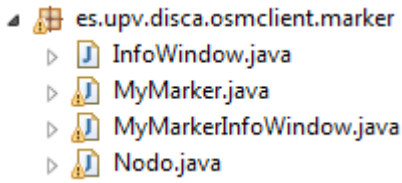

Figura 6.1.d. Estructura del paquete marker.

•*InfoWindow*

Clase abstracta para la creación de la ventana de información y sus métodos de apertura y clausura.

•*MyMarker*

Esta clase esta modificada a partir de la clase Marker y constituye un marcador con una posición, una imagen, un titulo, una descripción y subdescripción asociados. Además tiene un campo de tipo que señala el tipo de marcador. Los tipos son "ninguno", "comienzo", "waypoint" y "destino".

•*MyMarkerInfoWindow*

Clase modificada a partir de MarkerInfoWindow, la cual extiende de InfoWindow. Representa la ventana de información del marcador y tiene asociado un objeto de tipo MyMarker.

•*Nodo*

Esta clase es una especialización de MyMarker y sirve para distinguir los marcadores de los puntos generados al trazar la ruta. Implementa la interfaz Comparable para ordenar posteriormente los puntos de ruta.

6.1.e) es.upv.disca.osmclient.overlay

Este paquete tan solo posee una clase, por lo que no pondré una imagen de su estructura.

- *RotationGestureDetector*

  Esta clase se encarga de manejar la rotación del mapa. La clase recoge la información de los gestos introducidos en el mapa y si se trata de un gesto de rotación (Dos dedos girando alrededor de la pantalla) obtiene el angulo formado por los dedos entre la posición inicial y final devuelve ese valor como rotación del mapa.

6.1.f)es.upv.disca.osmclient.routing

Este paquete contiene las clases que permiten generar la ruta.

- *OSRMRoadManager*

  Esta clase la he explicado antes y la unica modificación que le he hecho con respecto a la incluida en la librería es que las instrucciones se obtienen de los recursos de Android (Para facilitar la traducción) y ahora puede hacer la petición de ruta tanto al servidor de OpenStreetMap como al Servidor del proyecto.

- *RoadManager*

  Clase de la que hereda la anterior. De nuevo, solo le he añadido un nuevo método: isOutOfRoute(ArrayList<GeoPoint[]> lsegmento, Road road, GeoPoint currentPosition) el cual devuelve verdadero si el punto "currentPosition" esta alejado más de 50 metros de la carretera "road" y los segmentos de esta "lsegmento".

6.1.g)es.upv.disca.osmclient.utils

En este paquete estan contenidas las clases que pueden ser de utilidad en el proyecto.

- *DistanceCalculator*

  Esta clase es especialmente útil dado que posee métodos tanto para calcular la distancia entre dos puntos Geográficos como la distancia entre un punto y un segmento formado por otros dos puntos. La distancia devuelta esta representada en kilómetros.

6.1.h)es.upv.disca.osmclient.views

Finalmente, este paquete incorpora las clases referentes a vistas personalizadas que he tenido que utilizar sobretodo para el momento de generar vistas dinámicamente, por código.

- *ElemFavView*

  Esta vista hereda de LinearLayout y su layout asociado es "elem_fav_view". Posee dos TextView y dos ImageView y se utiliza para representar los lugares y rutas almacenados en favoritos.

- *ElemRutaView*

  Esta vista hereda de RelativeLayout y su layout asociado es "elemento_ruta". Posee un ImageView y cuatro TextView y se utiliza para representar las instrucciones cuando se abre el dialogo con la lista de instrucciones.

- *OSMClientSearchView*

  Esta vista hereda de SearchView e implementa la clase OnQueryTextListener. Un SearchView es una vista con un campo para escribir un texto y un botón de búsqueda. En el caso de esta vista, el texto introducido en la búsqueda se utiliza como parámetro para una instancia de la clase GeocoderAsyncTask, por lo que sirve para encontrar el punto geográfico de un lugar en concreto.

6.2.Paquetes de Android

Cuando se crea un proyecto de Android en eclipse (O en cualquier otro IDE) se genera un archivo llamado AndroidManifest.xml y dos carpetas, gen y res.

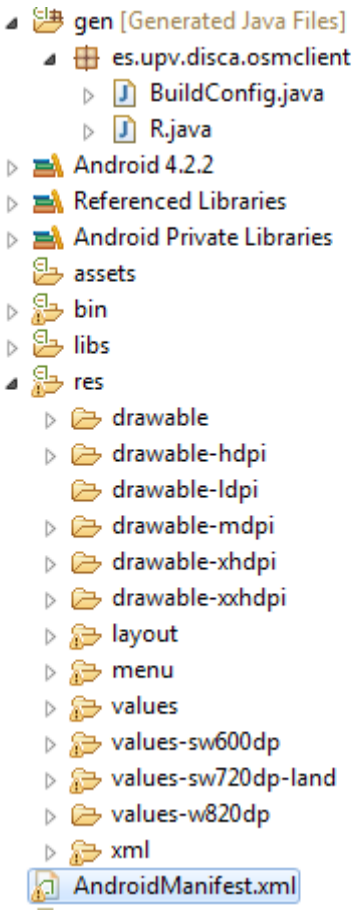

Figura 6.2. Estructura de los paquetes de Android

6.2.a)AndroidManifest.xml

Cada aplicación debe tener este archivo. El manifiesto contiene información esencial de la aplicación para el sistema operativo de Android, información que debe tener antes de ejecutar la aplicación[23].

Mi aplicación tiene los permisos de localización, comprobación del estado del wifi y del internet, escritura en almacenamiento externo y el permiso para poder depurar la aplicación. Como meta-datos posee una clave Bing para acceder a los mapas en vista satelite, además las actividades del proyecto se declaran aqui para que Android las tenga en cuenta. La actividad OSMClientActivity esta definida como principal y como lanzadora de la aplicación.

6.2.b)Carpeta de recursos (res)

•*drawable*

Según la pagina de Android developers, un recurso drawable es un concepto genérico para un gráfico que puede ser dibujado en pantalla y que puedes obtener por código mediante el método getDrawable(int). También se refiere a otros recursos XML con atributos como android:drawable y android:icon. Hay distintos tipos de drawables como archivos Bitmap, listas de capas, listas de estados, etc[24]. Aparte de drawable, estan las carpetas drawable-ldpi, drawable-mdpi, drawable-hdpi, drawable-xhdpi, etc. Estas carpetas diferencian a los distintos drawables por su resolución de menor a mayor, para que se puedan adaptar a las resoluciones de pantalla de distintos dispositivos.

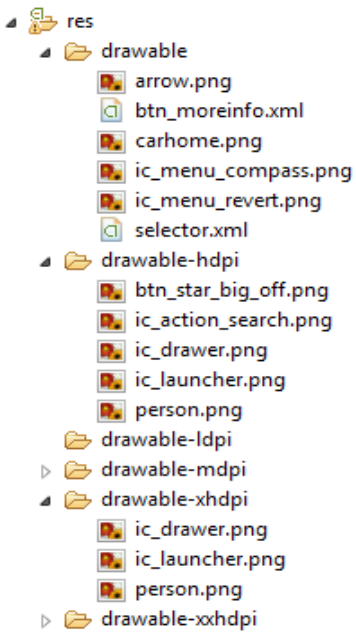

Figura 6.2.c. Carpetas del recurso drawable.

- *layout*

  En esta carpeta se encuentran los layouts asociados a las actividades o a vistas personalizadas.

  - **bonuspack_bubble.xml**

    Este es el layout por defecto que se usa para crear las InfoWindow.

  - **dialog_fav_title.xml**

    Layout con un TextView y un botón. Constituye al última vista del dialogo creado para ver la lista de instrucciones de la ruta.

  - **drawer_list_item.xml**

    Layout auxiliar del Navigation Drawer, representa el texto de cada elemento del menú.

  - **elem_fav_view.xml**

    Layout asociado a una vista y que muestra la información de los favoritos guardados en la base de datos. Posee dos TextView y dos botones, uno para editar el titulo del favorito y otro para eliminarlo.

  - **elemento_ruta.xml**

    Layout de los elementos de la lista de instrucciones. Tiene una ImageView y cuatro TextView en los que se muestra la instrucción a tomar en ese punto.

  - **favourites_layout.xml**

    El layout asociado a la actividad de favoritos. Cuenta con un TabHost o vista de pestañas con las listas de los lugares y rutas almacenados en favoritos. dentro de cada pestaña hay un LinearLayout que contiene vistas del tipo "elem_fav_view".

  - **fragment_planet.xml**

    Layout (Sin uso) con un ImageView para representar una posible imagen en el Navigation Drawer.

•info_window.xml

Ventana de información personalizada con un TextView para el nombre del marcador y cinco botones para crear un marcador de destino, origen, punto de ruta o para eliminarlo y un botón para abrir el dialogo con las acciones extra.

Figura 6.2.c.i. Visual del layout info_window.

•lista_ruta.xml

Layout constituido por un LinearLayout y un ScrollView para contener las vistas de tipo "elemento_ruta".

•main.xml

El layout de la actividad principal. Es el layout más complejo con diferencia, esta compuesto a simple vista por tres botones para accionar los métodos lanzarBotonCompass(), lanzarBotonMiUbic() y optionsClicked() de la clase OSMClientActivity los cuales respectivamente orienta el mapa hacia el norte, mueve el mapa hasta la posición actual y abre el Navigation Drawer.

También posee un MapView que ocupa toda la pantalla y el Navigation Drawer que permanece escondido hasta su apertura. También hay un layout invisible que tan solo aparece cuando se crea la ruta y contiene la información referente a esta y un botón para comenzar el modo de navegación (llama al método lanzarNavegación()).

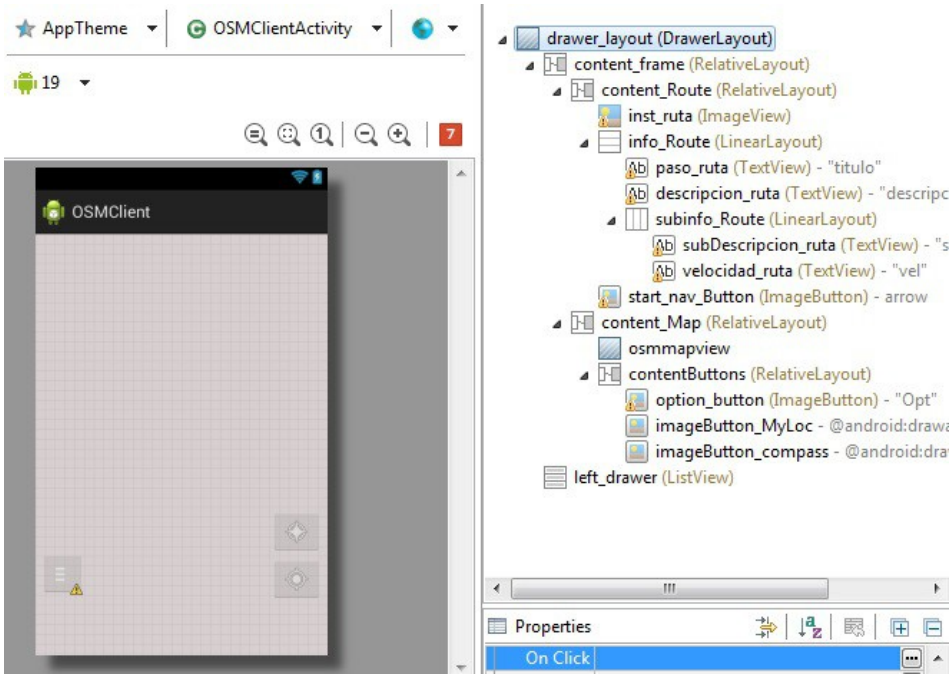

Figura 6.2.c.ii. Estructura de las vistas del layout main y su representación en pantalla.

•settings_lay.xml

Este es el layout asociado a la actividad de Settings y tiene colocados tres botones sobre un fondo negro. Cuando se pulsa encima de los botones, estos cambian de color gracias al drawable "selector.xml" asociado. Los botones accionan los métodos lanzarZoomGestures(), lanzarZoomButtons() y lanzarRotationGesture() de la clase SettingsActivity.

•titleedittext.xml

Vista sencilla con un TextEdit para cambiar el titulo de los lugares y las rutas almacenados en favoritos (Se incorpora en el dialogo que se abre al pulsar sobre el botón de editar).

•menu

Este paquete contiene todos los menús creados para la aplicación. Un menú puede ser simplemente una lista de acciones a realizar, aunque estas acciones pueden aparecer en la ActionBar como iconos.

•osmclient.xml

En este menú todas las acciones aparecen en la ActionBar. Por un lado esta la SearchView permanentemente visible y por el otro hay dos iconos que se alternan entre ellos, el icono de trazar la ruta y el de borrarla.

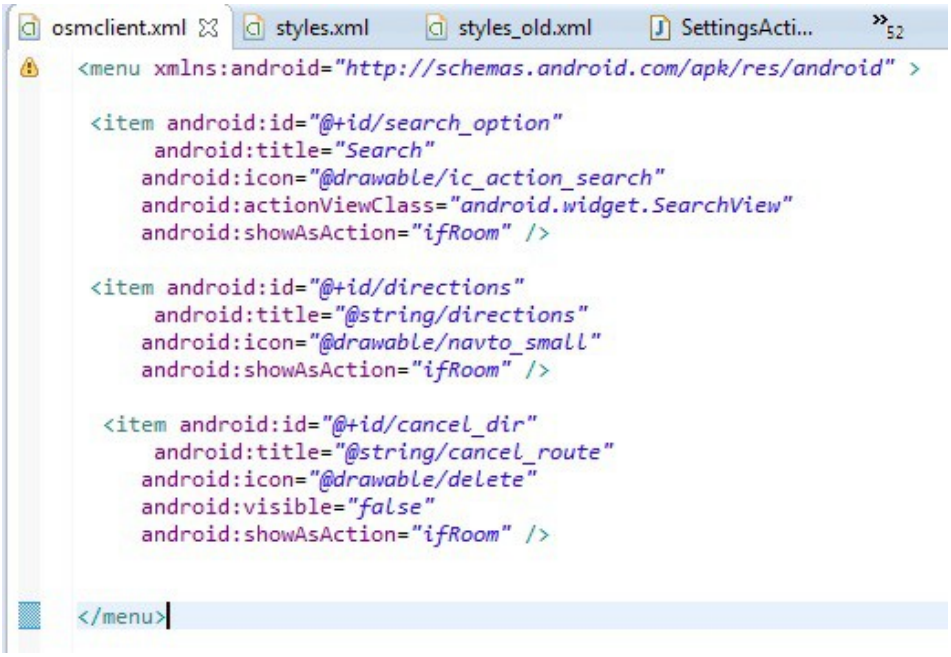

Figura 6.2.c.iii. Vista del menú principal y único de la aplicación.

•values

En esta carpeta están todos los tipos de valores que no pertenecen a las anteriores categorias y cuyas instancias se pueden definir en un único fichero (No como los layouts, por ejemplo, que tenemos un archivo para cada vista).

• arrays.xml

En este archivo están definidos todos los arrays de strings de la aplicación los cuales se usan en las listas de los dialogos, del Navigation Drawer, etc.

•color.xml

Como su nombre indica, aqui se almacenan los colores definidos por el usuario (en valor decimal o hexadecimal) y se les identifica por un nombre, por ejemplo "Yellow".

•strings.xml

Aquí se almacenan los strings, los textos que aparecen en la aplicación. Aquí es donde se observa la gran importancia de obtenerlos desde recursos, pues si creásemos la carpeta values-es (ES de España) podríamos copiar todos los strings de values y pegarlos en esta para traducirlos al español conservando el mismo identificador (No hace falta crear variables adicionales en las clases de la aplicación) con lo cual se simplifica enormemente la traducción de la aplicación.

•styles.xml

Finalmente, aquí se almacenan los estilos usados en la aplicación para poder reutilizarlos más fácilmente. Se pueden crear estilos nuevos a partir de otros e incluso hacer que toda la aplicación herede de un mismo estilo.

7.Elementos Visuales

7.1.Actividad principal

7.1.a)Ventana

La actividad principal esta compuesta por una vista de tipo MapView que ocupa toda la pantalla (A excepción de la Action Bar), un botón en la esquina inferior izquierda para abrir el Navigation Drawer y dos botones en la esquina inferior derecha. Uno de ellos, el que tiene el símbolo del gps, sirve para centrar el mapa en la ubicación actual y el otro botón, que tiene el dibujo de una brújula, sirve para situar la orientación del mapa hacia el norte.

Justo debajo de la Action Bar hay un Layout que solo se hace visible cuando el usuario tiene una ruta creada en el mapa, es decir cuando se ha pulsado previamente al botón de "crear ruta". Este Layout contiene información de la ruta creada como el destino, la distancia total del recorrido y su duración (Esto solo aparece cuando se ha creado la ruta. Cuando comience la navegación estos textos se sustituirán por la información del paso actual de la ruta). Al lado de esta información esta situado el botón para comenzar la navegación el cual tiene el dibujo de una flecha azul en él.

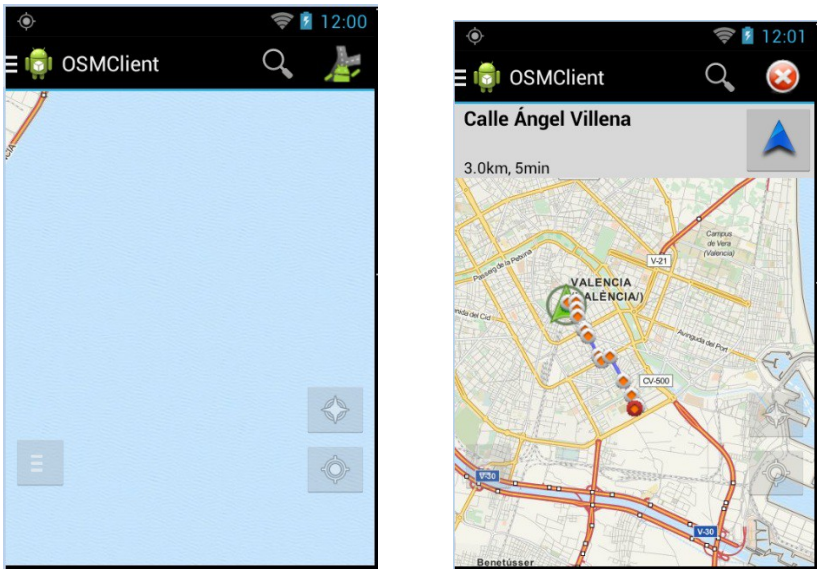

Figura 7.1.a.i. Actividad principal sin y con la ruta creada.

Si se pulsa en el layout que estaba previamente oculto se abre un dialogo con una lista con las indicaciones para la ruta generada. Si se presiona el botón de añadir ruta a favoritos, esta se guardará con el titulo escrito en el campo de texto.

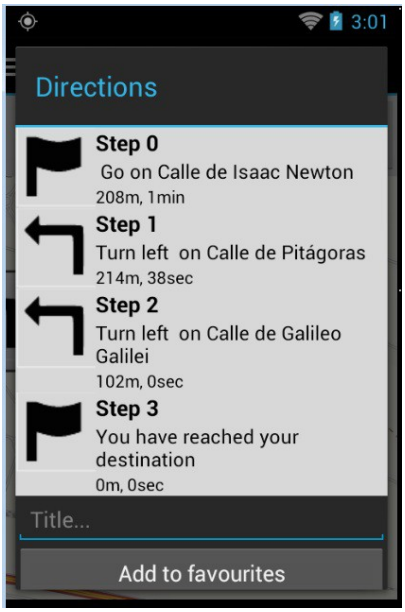

Figura 7.1.a.ii. Ejemplo de la vista para listar las indicaciones de la ruta.

7.1.b)Action Bar

La Action Bar o barra de acción, situada arriba de la ventana, es una característica de la ventana que provee al usuario de funciones útiles como son el mostrar el nombre de la actividad actual, en la que se encuentra el usuario, permite acceder a funciones importantes como la de búsqueda y el trazado de ruta además de añadir un botón para acceder al menú o en el caso de esta aplicación, para abrir el Navigation Drawer[25].

En la actividad principal, la Action Bar muestra el titulo de "OSMClient" o, si se ha elegido una opción del Navigatión Drawer, muestra la opción elegida. A la izquierda del titulo hay un botón para abrir el Navigation Drawer y a la derecha están los botones para buscar un lugar (Con el dibujo de una lupa, figura 7.1.b) y para trazar y borrar una ruta (Con los dibujos de ruta y cancelar respectivamente como se puede observar en la figura 7.1.a.i).

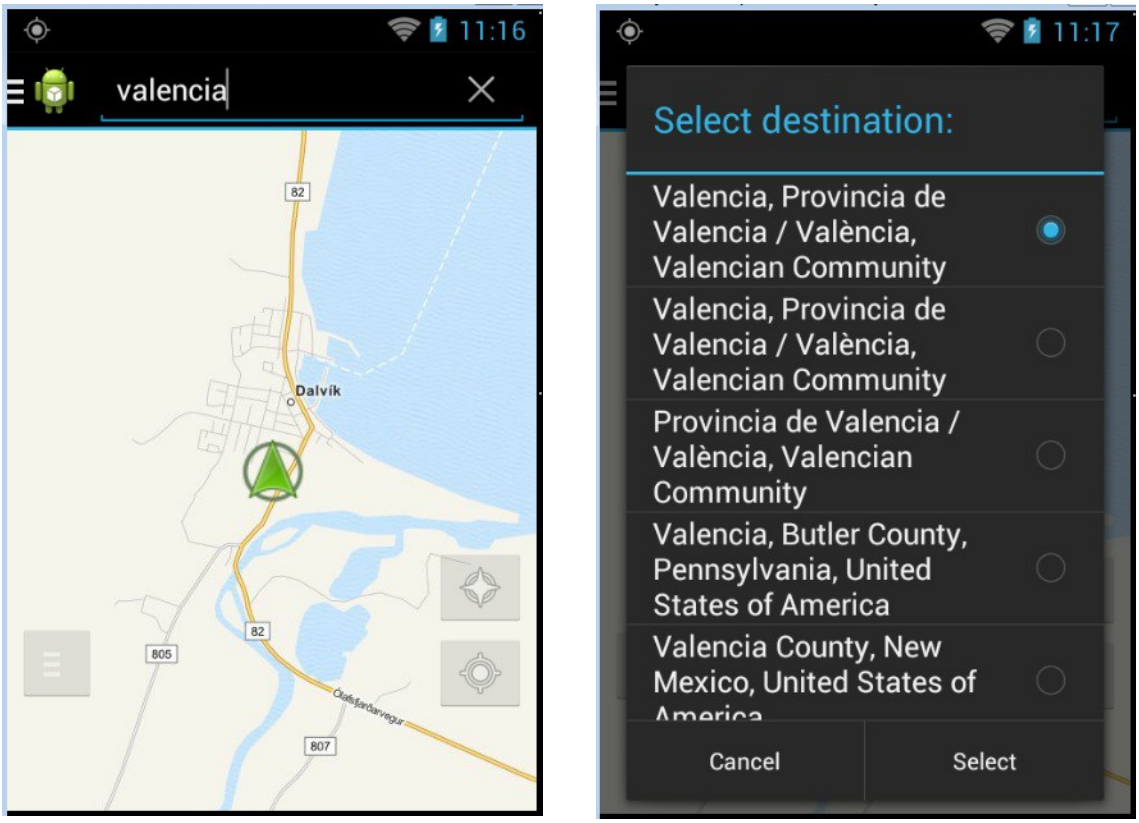

Figura 7.1.b. Vistas de la función de búsqueda antes y después de buscar la localización introducida.

7.1.c)Navigation Drawer

El Navigation Drawer es un panel deslizable que aparece cuando se desliza el dedo desde el lado izquierdo de la pantalla hacia el derecho. Actúa como un menú vertical y tiene como opciones la función de cambiar el tipo de ruta (Más rápida o más corta, si se elige la más rápida hace la petición de ruta al servidor de la aplicación, en caso contrario al de OpenStreetMap), cambiar el tipo de capa (vista de satélite, calle o híbrida), crear y eliminar la ruta, acceder a los lugares guardados en favoritos además del lugar designado como casa y finalmente la posibilidad de cambiar las configuraciones de la aplicación (Ventana de opciones).

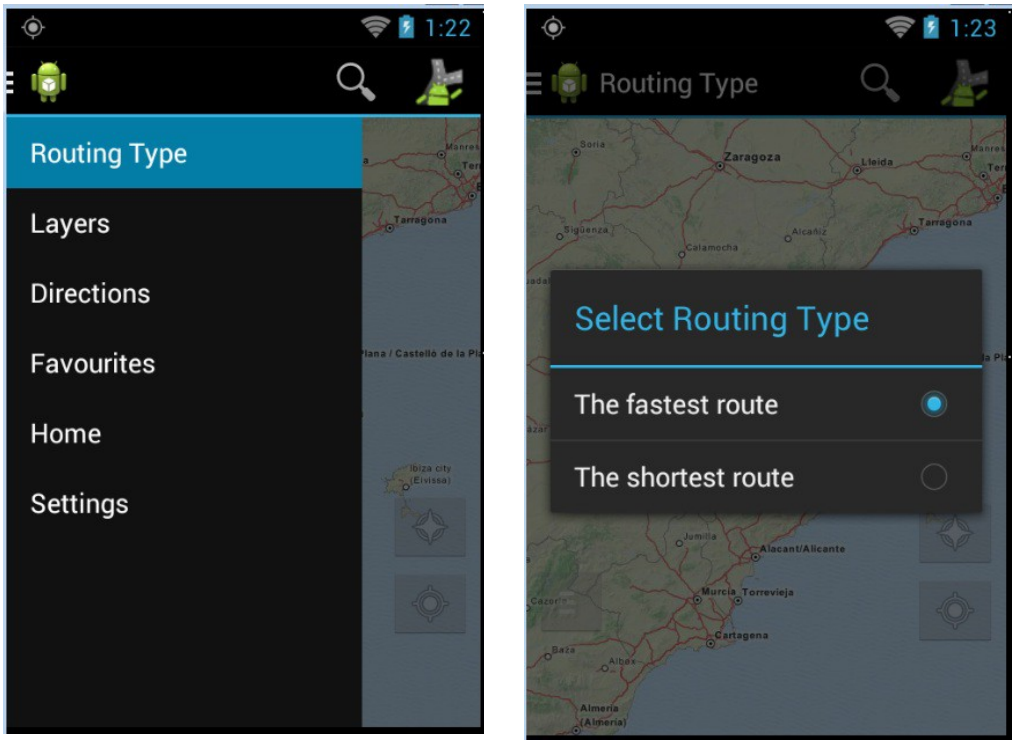

Figura 7.1.c.i. Vistas del Navigation Drawer y del dialogo para elegir el tipo de ruta.

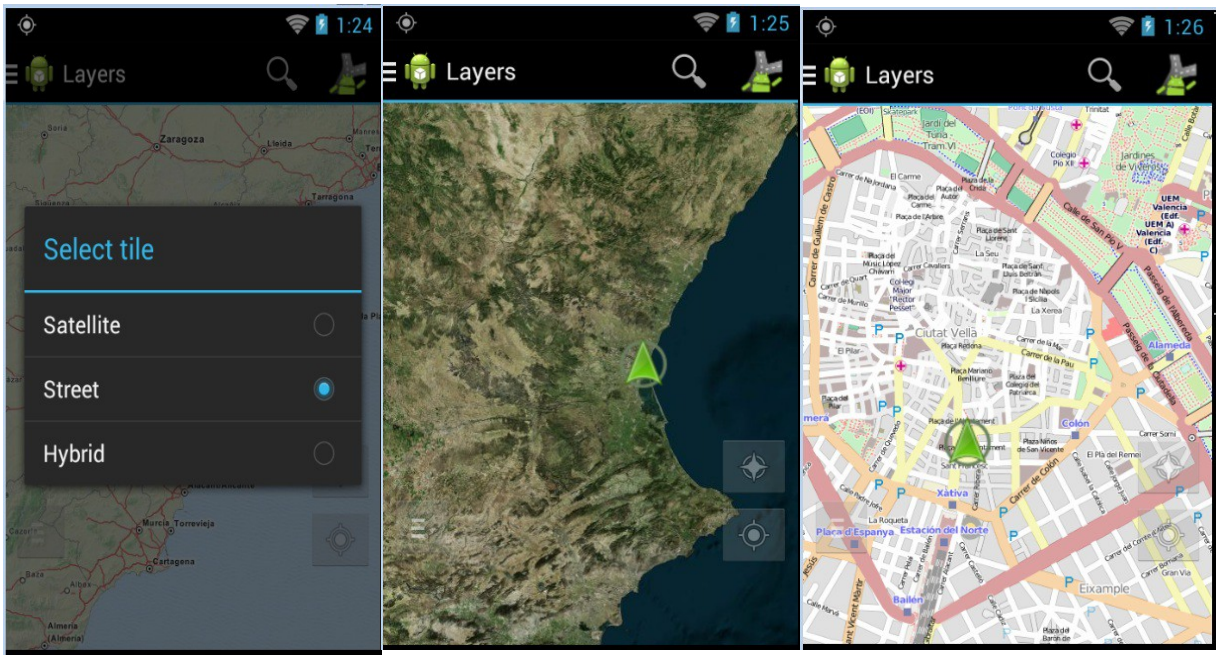

Figura 7.1.c.ii. Vistas del dialogo para elegir el tipo de capa y ejemplos de las vistas de satélite e híbrida.

7.1.d)MapView

En el capitulo de codificación he explicado algunos de los métodos que afectan al MapView por lo que ahora voy a mostrar el aspecto visual que poseen los distintos tipos de marcadores y sus respectivas ventanas de información:

Figura 7.1.d.i. De izquierda a derecha, marcador de posición (Sin tipo), de destino de ruta, de comienzo de ruta y de punto de ruta.

Figura 7.1.d.ii. Ejemplo de una ventana de información de un marcador.

El texto en negrita es el nombre de la calle o lugar, si no se pudiese obtener este nombre aparecería simplemente como "marcador". Aparte de estos marcadores, hay unos nodos que se generan al trazar la ruta y que contienen las indicaciones a seguir durante la ruta:

Figura 7.1.d.iii. Ejemplo de una ruta trazada, la ruta a seguir esta marcada por la línea de color azul y los nodos. El nodo verde representa el origen, el rojo el destino y los blancos son los puntos intermedios.

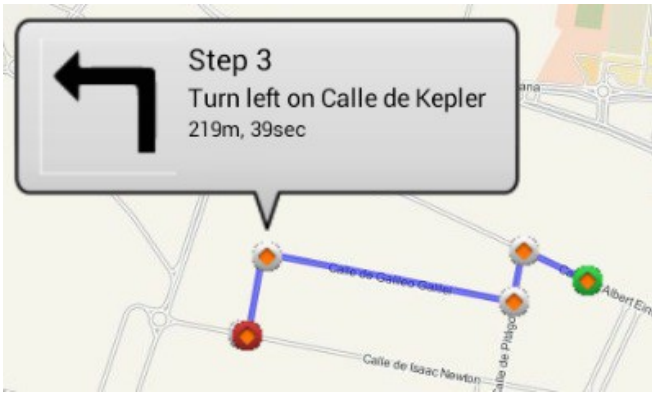

Figura 7.1.d.iv. Ejemplo de ventana de información de un nodo.

7.2.Actividad de favoritos

La actividad de favoritos tan solo cuenta con dos pestañas, una para listar los lugares guardados en la base de datos y la otra para las rutas. Al contrario que la actividad anterior, no tiene ni Action Bar ni Navigation Drawer:

Figura 7.2.i. Ejemplos de lugares y rutas mostrados por la actividad de favoritos.

Los botones situados a la derecha del titulo y la descripción del favorito sirven para editar el titulo del favorito y borrar el propio favorito almacenado.

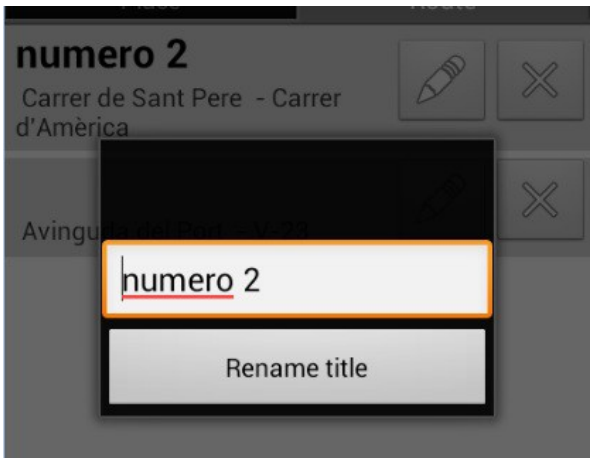

Figura 7.2.ii. Dialogo para editar el titulo del favorito.

7.3.Actividad de opciones

La actividad de opciones cuenta con tres botones para modificar el gesto de hacer zoom, mostrar o quitar los botones de zoom y modificar el gesto de rotación del mapa:

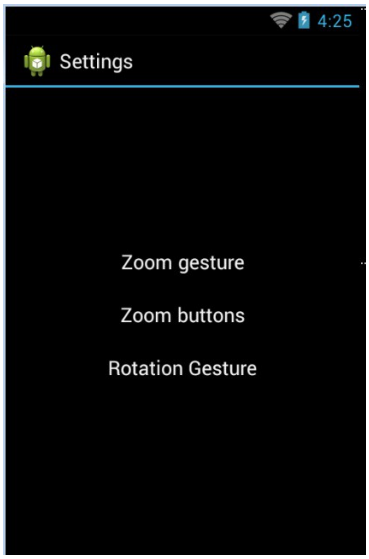

Figura 7.3.i. Actividad de opciones.

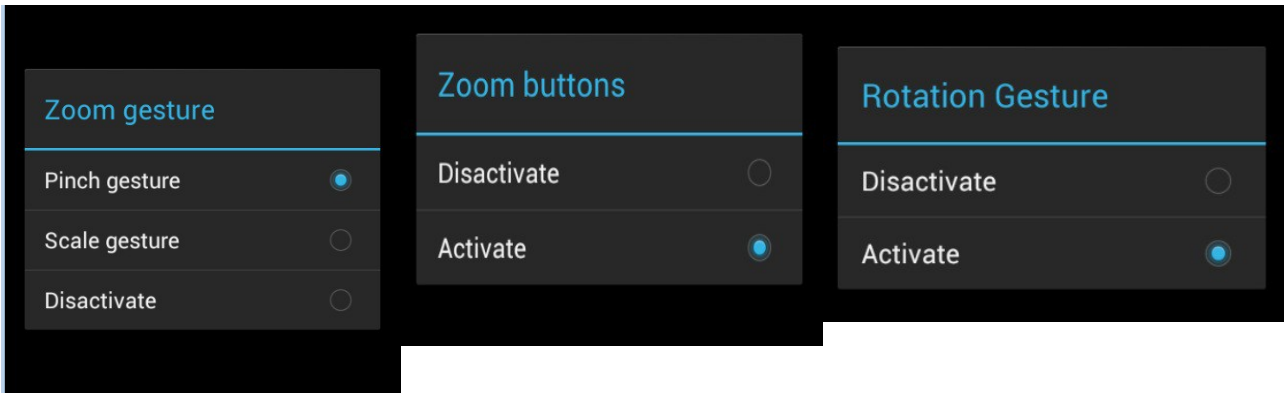

Figura 7.3.ii. Diálogos de los botones de opciones con sus valores predeterminados.

8.Funciones de la aplicación

En este capítulo explico en mayor detalle las funciones de la aplicación mostradas en el capítulo 5 y aquellas que no halla mencionado antes.

8.1.Mapa

El mapa de la ventana principal posee las siguientes funcionalidades:

8.1.a)Hacer zoom

En el mapa se puede hacer zoom tanto con los botones de zoom como con un gesto de "pinza" realizado con dos dedos sobre la pantalla táctil. Si los dedos se acercan se alejará el zoom en un nivel y si los dedos se alejan el zoom se ampliará. La capacidad de hacer zoom se puede desactivar desde las opciones.

8.1.b)Rotar el mapa

El mapa se puede rotar haciendo un gesto circular con dos dedos. Dependiendo del sentido en el que giren los dedos el mapa rotará hacia un sentido u otro. Si se presiona el botón de la brújula el mapa se orientará automáticamente en dirección al norte.

8.1.c)Desplazarse por el mapa

Si se desliza un dedo sobre el mapa, este se moverá en esa misma dirección.

8.1.d)Situar un marcador en el mapa

Si se mantiene presionado en un punto del mapa durante unos segundos, en ese punto aparecerá un marcador de color verde. Si se fuera a crear otro marcador verde se borraría el anterior.

8.1.e)Abrir un marcador del mapa

Los marcadores se abren al hacer clic en ellos, mostrando su pantalla de acciones.

8.2.Navegación

Esta es sin duda la función más compleja de la aplicación. En el apartado de "RoutingThread" del capítulo 6.1.b he explicado como funciona la navegación a nivel de código y en la figura 5.2.d lo he representado tan solo como "modo de navegación".

La navegación comienza cuando se pulsa el botón de navegación. Acto seguido el mapa se centra en la posición del usuario, la flecha que marca la orientación del usuario con respecto del mapa se queda fijada señalando hacia arriba y el mapa se orienta hacia la dirección en la que se desplaza el usuario. El layout que contiene la información de ruta cambia su titulo en negrita por el de "paso 1", su descripción por la instrucción a seguir, se genera una imagen acorde con esta instrucción y en la subdescripción se marca la duración y longitud del trayecto entre el punto de ruta anterior y el actual, además de mostrar la velocidad a la que se desplaza el usuario en m/s.

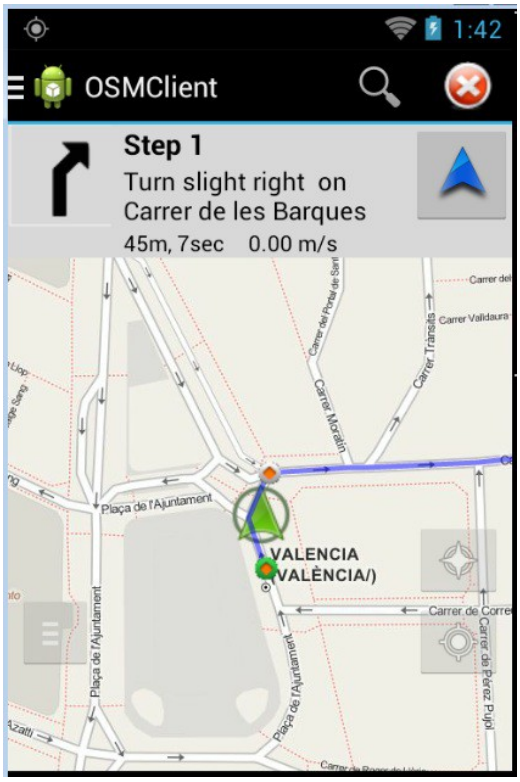

Figura 8.2.i. Ejemplo de inicio del modo navegación.

La navegación continua del siguiente modo, cuando el usuario llega al punto de ruta actual (Esta a menos de 15 metros), espera a ver si se aleja de este punto y si se aleja y a la vez se acerca al siguiente punto de ruta la aplicación considera que ha pasado el punto actual y actualiza la información del layout (Carga la información del siguiente punto) (Figura 8.2.ii). En el caso de que el usuario se aleje de el punto actual y no se acerque al siguiente punto, el sistema trazará una nueva ruta y la navegación continuará a partir de esta ruta.

Si el usuario se aleja más de 50 metros de la ruta actual, el sistema deduce que se ha perdido y que debe calcular una ruta hacia el destino. Por otra parte, si antes de llegar al punto de ruta actual el sistema observa que el usuario se esta alejando de este, el sistema calcula una nueva ruta para asegurarse de que el usuario no se ha pasado el punto de ruta y de que se muestran las indicaciones correctas para el punto de ruta actual.

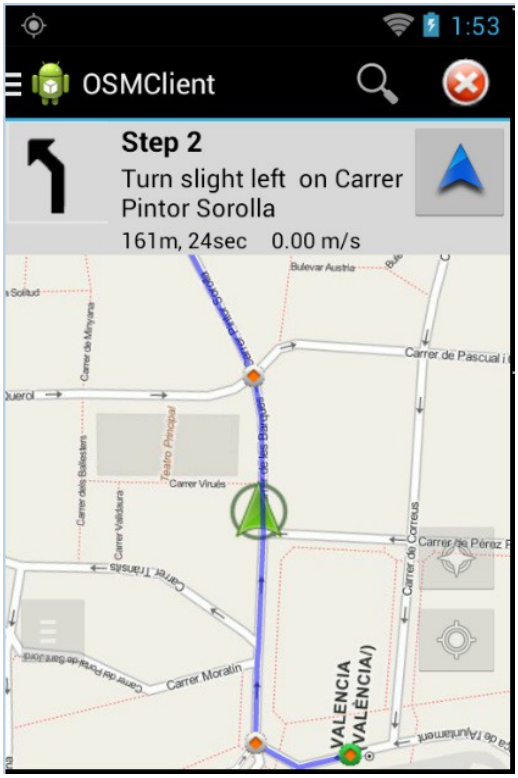

Figura 8.2.ii. La navegación ha llegado al punto actual y se dirige al siguiente punto de la ruta.

8.3.Búsqueda de localizaciones

La búsqueda de lugares se realiza desde el widget (Elemento gráfico) de búsqueda de la Action Bar. Una vez se ha introducido el nombre de la localización a buscar, el sistema comprueba si se esta conectado a Internet (Para poder acceder al servidor) y si la respuesta es afirmativa y el sistema ha encontrado el lugar muestra una lista de hasta 10 direcciones ordenadas por su distancia hasta el punto actual del usuario (Se debe tener esto en cuenta para seleccionar un resultado más o menos relevante). El usuario entonces puede elegir una de las direcciones mostradas y acto seguido se generará un marcador en el mapa en la posición de la localización seleccionada.

9.Pruebas y desarrollo del proyecto

Las pruebas se han realizado en diversos dispositivos:

- Samsung Galaxy S2 con versión de Android 4.1.1 y API 16. Emulado por Genymotion.

- Dispositivo personalizado con versión de Android 4.2.2 y API 17. Emulado por el AVD Manager.

- Lenovo A850 con versión de Android 4.2.2 y API 17. Dispositivo físico con capacidad para conectarse a Internet y utilizar el gps, pero sin opción de brújula (Detecta la orientación en base al desplazamiento).

Para el desarrollo del proyecto he seguido una metodología de desarrollo basado en prototipos. He ido cumpliendo los requisitos del proyecto uno por uno y dando mayor prioridad a los que cumplían el objetivo crítico que son el trazado de las rutas y la navegación por estas (¿Como si no sería un navegador gps?).

Primero he desarrollado los elementos visuales que componen la actividad principal (Para tener la estructura creada), ello incluye el mapa, la Action Bar y el Navigation Drawer para que actuara de menú. Una vez hube comprobado que la estructura funcionaba en distintas resoluciones, desarrollé los marcadores y sus respectivas ventanas de información para que me ayudasen con el trazado de las rutas. Con los marcadores tuve un problema a la hora de insertarlos en el mapa, pues parecía que si intentaba alterar el método que reconoce los gestos (onTouch()) del MapView desde su método setOnTouchListener() la vista del mapa se bloqueaba. Esto lo resolví usando unas clases propias de la librería que utilizaba y que estaban diseñadas para este proposito.

Una vez resuelto el tema de los marcadores seguí los tutoriales que los creadores de la librería OSMBonusPack habían publicado en su pagina web y adapté su método de creación de ruta al de mi proyecto. Tras haber comprobado que la ruta se trazaba correctamente, continué con la implementación de la navegación.

La ventaja del desarrollo de prototipos es que al actuar yo mismo como cliente, puedo suplir más fácilmente las carencias de la aplicación y dar mayor flexibilidad al desarrollo del proyecto[26]. Puesto que estaba trabajando con

una librería desconocida, esta era la mejor opción por si surgían problemas inesperados.

En la implementación de la navegación es cuando más se ve la importancia del desarrollo de prototipos. Cuando creé la primera versión de la navegación esta apenas funcionaba. Me di cuenta de que los márgenes relacionados con la distancia a la cual se daba por sentado que el usuario había llegado al punto de ruta, eran demasiado altos (De centenares de metros). Al rectificar esto comprobé que a veces no calculaba bien cuando había sobrepasado el punto de ruta actual y avanzaba al siguiente. Por otra parte, a veces re-calculaba la ruta como si se hubiese alejado de esta cuando en realidad el usuario estaba a menos de un milímetro de esta.

Ambos problemas los corregí con un método que me indicaba si el usuario se estaba alejando de un punto en concreto y cambié el método que me avisaba si el usuario se había salido de la ruta por uno que media la distancia del usuario al segmento más próximo formado entre dos puntos de la ruta (El anterior media la distancia del usuario respecto al punto de ruta más próximo, con lo cual en trayectos con puntos de ruta muy separados entre si devolvía un valor incorrecto). Además, para que la interfaz de usuario no se bloquease hice que el hilo que computaba la navegación se durmiese unas décimas de segundo en cada iteración.

Estas pruebas las hice tanto en el dispositivo emulado por Genymotion como en el dispositivo físico, aunque en este me vi limitado por el hecho de que era más costoso. Una vez tuve acabada la parte esencial, implementé las funciones referentes a la búsqueda de lugares y al almacenamiento de lugares y rutas en favoritos, con los cuales tuve algunos problemas menores.

10.Conclusión

A lo largo del desarrollo del proyecto me he encontrado con problemas que me han retrasado más de lo que tenía previsto. No obstante, he conseguido cumplir casi todos los objetivos que me había marcado.

El único objetivo que no he podido conseguir es el de compartir una localización o ruta con otro usuario de la aplicación. El planteamiento inicial era añadir una acción extra a los marcadores y a la lista de rutas para compartir el lugar y la ruta en cuestión. Esta acción estaría representada por un botón que al pulsarlo mostraría una lista de programas recomendados (Whatsapp, SMS, Line, ...) con los cuales se enviaría un mensaje con la localización del lugar o ruta al compañero seleccionado. Esta funcionalidad no he podido implementarla por falta de tiempo y sobretodo de experiencia en el sistema operativo Android.

Mientras indagaba en la materia del proyecto me he dado cuenta de las posibilidades que ofrece Android. Además, el desarrollo de este proyecto me ha servido para obtener experiencia en una aplicación profesional y para realizar deducciones autónomas:

-No solo he mejorado mis conocimientos en el desarrollo de interfaces para este sistema operativo, sino que además he podido aprender a hacer un navegador gps desde cero (A partir de las funciones que me ofrecían las librerías).

-He mejorado mis conocimientos en el desarrollo y perfeccionamiento de algoritmos de una forma no solo teórica sino también práctica.

-He comprobado lo necesario que es para un proyecto de gran envergadura el disponer de un código limpio y bien estructurado para encontrar rápidamente los fallos y aplicar más fácilmente las mejoras oportunas.

-Del mismo modo, me he dado cuenta de lo importante que es el disponer de un modelo inicial desde el cual partir; de tener desde el principio una idea clara de lo que se quiere hacer.

Como lineas futuras, podría implementar la funcionalidad que he explicado anteriormente. Por otra parte, al estar la aplicación tan solo en inglés sería interesante utilizar la funcionalidad que ofrece Android con los recursos para traducir la aplicación al castellano o a otros idiomas. Además de una mejora en la traducción, se podría añadir una opción para cambiar el tema global de la

aplicación (Hay personas que les gustan más las características de un tema específico) y mejorar la forma en que se ven los elementos visuales (Sobretodo el layout que muestra la información de ruta).

Una de las desventajas de la aplicación creada es que no esta capacitada para tramitar demasiadas peticiones al servidor de OpenStreetMap pues este es un servidor público y en sus condiciones de uso restringe el envío masivo de datos[27]. No obstante este problema se puede solventar con la creación de un servidor privado basado en una instancia de OpenStreetMap[28].

En conclusión, con este proyecto he podido instruirme en el desarrollo de una aplicación profesional y darme cuenta de los recursos, tanto temporales como económicos, que precisa.

11.Bibliografía

1. Android[Internet]. Wikipedia. 16 junio 2014 [cita 18 junio 2014]. Disponible en: http://es.wikipedia.org/wiki/Android

2. GNU General Public License[Internet]. Wikipedia. 2 abril 2014 [cita 18 junio 2014]. Disponible en: http://es.wikipedia.org/wiki/GNU_General_Public_License

3.OpenStreetMap[Internet]. openstreetmap.org, 19 nov 2013 [cita 19 junio 2014]. Disponible en: http://wiki.openstreetmap.org/wiki/ES:Main_Page

4.Jonathan Bennet. 2010. *OpenStreetMap*[en línea]. Birmingham: Packt Publishing Ltd. [cita 2 julio 2014]. ISBN: 978-1-847197-50-4. Disponible en: http://books.google.es/books?id=SZfqRcPXApoC&printsec=frontcover&dq=openstreetmap&hl=es&sa=X&ei=dmu0U5zxCIrG0QWutIAw&ved=0CCsQ6AEwAA#v=onepage&q=openstreetmap&f=false

5.¿Por que OpenStreetMap?[Internet]. openstreetMap.org, 31 mayo 2014 [cita 19 jun 2014]. Disponible en: http://wiki.openstreetmap.org/wiki/ES:FAQ

6.Eclipse(Software)[Internet]. Wikipedia. 11 mar 2014 [cita 22 junio 2014]. Disponible en: http://es.wikipedia.org/wiki/Eclipse_%28software%29

7.Schildt H. 2005. *Java 2 v5.0*. The McGraw-Hill Companies. ISBN: 84-415-1865-3.

8.Java (lenguaje de programación)[Internet]. Wikipedia. 18 junio 2014 [cita 22 junio 2014]. Disponible en: http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29

9.Conozca más sobre la tecnología java[Internet]. www.Java.com. [cita 24 junio 2014]. Disponible en: https://www.java.com/es/about/

10.Desarrollo de programas para Android[Internet]. Wikipedia. 23 marzo 2014 [cita 24 junio 2014]. Disponible en: http://es.wikipedia.org/wiki/Desarrollo_de_programas_para_Android

11.Fragment[Internet]. developer.android.com. [cita 25 junio 2014]. Disponible en: http://developer.android.com/reference/android/app/Fragment.html

12.Activity[Internet]. developer.android.com. [cita 25 junio 2014]. Disponible en: http://developer.android.com/reference/android/app/Activity.html

13.View[Internet]. developer.android.com. [cita 25 junio 2014]. Disponible en: http://developer.android.com/reference/android/view/View.html

14.Intent[Internet]. developer.android.com. [cita 25 junio 2014]. Disponible en: http://developer.android.com/reference/android/content/Intent.html

15.Toast[Internet]. developer.android.com. [cita 25 junio 2014]. Disponible en: http://developer.android.com/reference/android/widget/Toast.html

16.NotificationManager[Internet]. developer.android.com. [cita 25 junio 2014]. Disponible en: http://developer.android.com/reference/android/app/NotificationManager.html

17.AsyncTask[Internet]. developer.android.com. [cita 25 junio 2014]. Disponible en: http://developer.android.com/reference/android/os/AsyncTask.html

18.SQLiteOpenHelper[Internet]. developer.android.com. [cita 25 junio 2014]. Disponible en: http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html

19.Dialog[Internet]. developer.android.com. [cita 25 junio 2014]. Disponible en: http://developer.android.com/guide/topics/ui/dialogs.html

20.osmdroid[Internet]. github.com. [cita 26 junio 2014]. Disponible en: https://github.com/osmdroid/osmdroid

21.osmbonuspack[Internet]. code.google.com. [cita 26 junio 2014]. Disponible en: https://code.google.com/p/osmbonuspack/wiki/Features

22.Nominatim[Internet]. wiki.openstreetmap.org. 29 may 2014 [cita 26 junio 2014]. Disponible en: http://wiki.openstreetmap.org/wiki/Nominatim

23App Manifest[Internet]. developer.android.com. [cita 27 junio 2014]. Disponible en: http://developer.android.com/guide/topics/manifest/manifest-intro.html

24.Drawable Resources[Internet]. developer.android.com. [cita 27 junio 2014]. Disponible en: http://developer.android.com/guide/topics/resources/drawable-resource.html

25.Action Bar[Internet]. developer.android.com. [cita 30 junio 2014]. Disponible en: http://developer.android.com/guide/topics/ui/actionbar.html#

26.Modelo de prototipos[Internet]. Wikipedia. 20 may 2014 [cita 1 julio 2014]. Disponible en: http://es.wikipedia.org/wiki/Modelo_de_prototipos

27.Nominatim usage policy[Internet]. wiki.openstreetmap.org. 11 julio 2013 [cita 2 julio 2014]. Disponible en: http://wiki.openstreetmap.org/wiki/Nominatim_usage_policy

28.Nominatim/Installation[Internet]. wiki.openstreetmap.org. 27 febrero 2014 [cita 2 julio 2014]. Disponible en: http://wiki.openstreetmap.org/wiki/Nominatim/Installation