



Escola d'Enginyeria de Telecomunicació i
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FIN DE CARRERA

TÍTULO DEL TFC: Aplicación Android de movilidad de invidentes

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática

AUTOR: Ingrid Lidó Monzón

DIRECTOR: Sergio Machado Sánchez

FECHA: 30 de Abril de 2011

Título: Aplicación Android de movilidad de invidentes

Autor: Ingrid Lidó Monzón

Director: Sergio Machado Sánchez

Fecha: 30 de Abril de 2011

Resumen

En este proyecto se ha desarrollado parte de una aplicación de movilidad de invidentes para Android. La introducción del destino se realiza por voz y a partir de ahí utilizando diversas herramientas se guía al usuario.

A lo largo del proyecto se realiza dos aplicaciones. En la primera se utiliza los mapas de Google Maps, debido a la facilidad de implementación y a la multitud de funcionalidades. No obstante, el uso de Google Maps tiene el inconveniente de las limitaciones de licencia.

Es por este motivo que más adelante se implementa una nueva aplicación en la que se utilizan los mapas de OSM. OSM tiene la ventaja de que es un proyecto gratuito que nos da más libertad. El lado negativo es que, al menos por ahora, no es tan potente como Google. Aún falta mejorar mucho los mapas, indicar las posiciones de los números en las calles, y el encaminamiento se tiene que realizar mapeando un mapa y usando, en este caso, Dijkstra.

Title: Android application mobility for blind

Author: Ingrid Lidó Monzón

Director: Sergio Machado Sánchez

Date: April, 30th 2011

Overview

This project has developed part of an accessible navigation system for the Android platform. The accessible user interface is based on the natives android voice recognition and voice synthesis systems. The navigation system computes pedestrian routes and in future development it should compute routes using public transport information.

The project is basically the study of what map system is suitable for this development. First, we based our study using Google Maps API because it is natively implemented in the Android Platform, has a relative updated and accurate maps, a routing system which computes routes for pedestrian, and, finally, a reverse geolocation system. Two are the main reasons to not base our solution in Google Maps API: first, the use restrictions imposed by the license agreement and that we can't improve the map including useful information for computing routes.

So, this project considered the use and the development of tools and software libraries to use Open Street Map as the geographic information system. Open Street Map, allows the users to edit the geographic information achieving a map detail level which can overcome Google Maps.

Dedicado a mi familia y amigos.

Gracias al hombre opaco porque sin él esto no habría sido posible.

Gracias a mis amigos porque han sabido despejarme cuando lo necesitaba.

Gracias a mis hermanos porque sólo con una llamada me han animado a seguir adelante.

Gracias al bicho por apoyarme siempre, especialmente en estos últimos meses.

Y sobre todo, gracias a mis padres por estar ahí siempre, pase lo que pase, sea la hora que sea, necesite lo que necesite. GRACIAS.

Índice

INTRODUCCIÓN	1
CAPÍTULO 1. VISIÓN GENERAL	2
1.1. Origen	2
1.2. Objetivos	2
1.3. Terminales	3
1.4. Arquitectura	5
1.5. Proyectos similares.....	6
1.5.1. WalkyTalky e Intersection.....	6
1.5.2. Kapten for Smartphone	7
1.5.3. Blind.....	8
CAPÍTULO 2. DESARROLLO APLICACIONES ANDROID	10
2.1. Instalación del entorno de desarrollo: SDK + Eclipse.....	10
2.2. Estructura de un proyecto android.....	13
2.2.1. Carpeta /src/	13
2.2.2. Carpeta /res/	14
2.2.3. Carpeta /gen/	14
2.2.4. Carpeta /assets/	15
2.2.5. Fichero AndroidManifest.xml	15
2.3. Utilidades usadas	16
2.3.1. Text To Speech	16
2.3.2. Speech To Text	16
CAPÍTULO 3. ESTUDIO DE API DE MAPAS PARA ANDROID.....	17
3.1. Introducción.....	17
3.2. Google Maps	17
3.2.1. Pasos para conseguir la API Key de Google	17
3.2.2. Objetos de mapas	19
3.2.3. Servicios ofrecidos	19
3.2.4. Términos de licencia.....	23
3.3. OSM	24
3.3.1. Formato de datos	24
3.3.2. Cálculo de rutas y navegación	24
3.3.3. Herramientas de edición.....	25
3.3.4. Servicios utilizados	25
3.3.5. Pequeña contribución a OpenStreetMap	29

CAPITULO 4. LAZANDROID: LA APLICACIÓN	31
4.1. Google Maps.....	31
4.2. OSM	34
CAPITULO 5. DEMO DE ENCAMINAMIENTO	40
5.1. Encaminamiento.....	40
5.2. Instalación Servidor OSM.....	41
CAPITULO 6. CONCLUSIONES	42
6.1. Resultados	42
6.2. Mejoras futuras.....	42
6.2.1. Vibración.....	42
6.2.2. Abecedario entrada datos	42
6.2.3. Sonidos.....	43
6.2.4. Opciones predeterminadas	43
6.2.5. Favoritos	43
6.2.6. Brújula.....	43
6.2.7. Puntos de interés.....	43
6.2.8. Ruta más corta contemplando servicio público	43
6.2.9. Bases de datos	44
6.2.10. Accesibilidad en edificios.....	44
6.2.11. Alerta de incidencias.	44
6.2.12. Indicaciones de movilidad por voz	44
6.3. Impacto medioambiental	44
CAPITULO 7. REFERENCIAS	45

INTRODUCCIÓN

Hoy en día los teléfonos móviles no son lo que eran. Ahora no sólo sirven para llamar por teléfono o mandar mensajes. Cuando decidimos comprar un móvil compramos un teléfono, una cámara de fotos, un tomtom, un instrumento para jugar, una conexión a internet...

Pero ya no sólo tenemos teléfonos inteligentes, sino que con la llegada de Android tenemos la posibilidad de tener un “móvil a medida”. Se tiene la posibilidad de descargar o desarrollar tus propias aplicaciones en función de las necesidades o gustos de cada persona, gratuitamente o por un módico precio.

Es por este motivo que se ha pensado en el desarrollo de este proyecto. Con Android se tiene la posibilidad de conseguir un móvil asequible en el que poder ir incorporando aplicaciones a medida que vayan saliendo sin necesidad de cambiar de terminal. También se acaba una época en la que tener que comprar móviles especialmente diseñados para invidentes, con lo que esto conlleva en el precio del mismo.

En el siguiente proyecto se desarrolla una aplicación de movilidad para invidentes sobre el entorno Android, implementando funcionalidades especialmente diseñadas para el fácil manejo del programa para personas con problemas de visión.

En el capítulo 1 se estudia los objetivos del proyecto, así como los terminales sobre los que se desarrollará y se verá algunos proyectos parecidos en este momento.

En el capítulo 2 se explica el entorno de desarrollo, así como lo necesario para su instalación.

En el capítulo 3 se habla de forma general de las API de mapas para android utilizadas en el proyecto: GoogleMaps y OSM.

En el capítulo 4, una vez comentadas las distintas APIs, se explica las aplicaciones realizadas así como los problemas encontrados en cada una de ellas.

En el capítulo 5, se habla de una demo de encaminamiento como solución a algunos problemas encontrados.

Por último en el capítulo 6 se hace una pequeña conclusión del proyecto realizado, y se analiza las posibles mejoras a desarrollar en un futuro.

CAPÍTULO 1. VISIÓN GENERAL

1.1. Origen

La tecnología de los teléfonos móviles cada día avanza más y más. Salen nuevas aplicaciones, nuevos diseños y funcionalidades para todo el mundo, o más bien para casi todo el mundo. Hay un público que no se suele tener en cuenta a la hora de diseñar sistemas operativos para móvil o aplicaciones y son las personas discapacitadas, concretamente los invidentes. Desde hace algunos años se viene desarrollando algunos terminales especialmente diseñados para los invidentes, aunque también cabe destacar que no son fácilmente accesibles debido a su coste. Es por esto que con la llegada de Android se abre una esperanza de conseguir accesibilidad a costes reducidos.

Con la llegada al mercado de los terminales con GPS y conexión permanente a Internet se plantea la posibilidad de tener en el propio móvil un programa que nos ayude a llegar de la forma más rápida a un destino concreto. Esto es una ayuda para cualquiera, pero más aún para personas invidentes.

La idea es crear una aplicación que facilite dicha necesidad añadiendo ciertas funcionalidades especialmente necesarias. Nos basamos en el sistema Android, ya que es una forma fácil de añadir aplicaciones nuevas a terminales, sin necesidad de unos requerimientos especiales, simplemente basta con el sistema operativo.

1.2. Objetivos

Los objetivos de este proyecto son los siguientes:

- Facilidad de manejar (Vibración).
- Facilidad introducción/salida de textos (SpeechToText, TextToSpeech).
- Encaminamiento paso a paso.
- Instrucciones orales.
- Señalización (Pasos de cebra, semáforos adaptados, entradas a edificios).
- Actualización.
- Alerta de incidencias.

1.3. Terminales

El Smartphone es un término comercial para denominar a un teléfono móvil que ofrece más funciones que un teléfono común. Una característica importante de casi todos los teléfonos inteligentes es que permiten la instalación de programas que se adecuen a las necesidades de cada usuario. Estas aplicaciones pueden ser desarrolladas por el fabricante del dispositivo, por el operador o por un tercero.

Algunos ejemplos de teléfonos denominados inteligentes son: BlackBerry, iPhone, Palm y todos los que tienen el sistema operativo Android, S60 ó Windows Mobile.

El Kernel de un sistema operativo es el núcleo del mismo, el software responsable de facilitar a los programas acceso seguro al ordenador, el encargado de gestionar recursos. Por ello es importante conocer qué núcleo utiliza cada uno de los sistemas operativos de los terminales.

Tanto Android como Palm están basados en Linux. BlackBerry está basado en un kernel propietario. iPhone se basa en OS X. S60 se basa en Symbian y Windows Mobile en Windows CE. La principal diferencia entre un kernel de propietario y uno de libre distribución se basa en que estos últimos cuentan con una amplia comunidad de desarrolladores, gracias a los cuales se encuentran rápidamente fallos, y se realizan mejoras, tanto para solucionar problemas como para adaptarse a los nuevos tiempos. En los sistemas propietarios es más costoso encontrar errores ya que deben ser los propios desarrolladores del sistema los que detecten y realicen mejoras, debiendo dedicar más recursos a investigación en estos sistemas, con el consiguiente aumento del coste.

Otro aspecto importante es la adaptabilidad de la plataforma, la facilidad para poder adaptarlo a diferentes terminales. En este sentido Android es el que mayor adaptabilidad presenta, ya que cada vez se está utilizando en más dispositivos, no sólo teléfonos móviles.

También cabe destacar la conectividad de los sistemas, ya que es importante para poder sacarle el mayor partido a las funcionalidades que se ofrecen. Es importante el hecho de que cuenten con acceso WiFi a Internet, así como conectividad 3G.

El Kernel es importante, sin embargo el usuario lo que más aprecia es la interfaz. En este sentido iPhone ha marcado estilo con sus iconos y su fácil acceso, además de la pantalla táctil que no requiere de un puntero, ya que cuenta con gran precisión al utilizarla con los dedos. Android también tiene una interfaz sencilla e intuitiva con una buena precisión. Sin embargo Windows Mobile siempre ha requerido el uso de punteros.

Otro aspecto importante es el grado de personalización de las interfaces de usuario. Android permite personalizar completamente el escritorio. Tiene varios

escritorios completamente personalizables para poder organizar mejor las aplicaciones, donde se pueden añadir accesos directos a los programas más utilizados. Windows Mobile también permite personalizar los iconos que aparecen en el escritorio. BlackBerry permite únicamente personalizar el tipo y tamaño de letra, y ocultar iconos de acceso directo a aplicaciones que no se utilizan.

	Android	BlackBerry OS 4.7	iPhone OS 3.0	S60 5th Edition	Palm WebOS	Windows Mobile 6.5
Interfaz intuitiva	Sí	Sí	Sí	Menos	Sí	Sí
Instalación de nuevas aplicaciones	Sencilla (Android Market)	Sencilla	Sencilla (App Store)	Compleja	Sencilla	Costosa
Notificación	Bandeja	Pop-up, fondo	Pop-up	Pop-up	Bandeja	Bandeja, pop-up
Administración de contactos	Google. Posibilidad de sincronización con otros servicios.	BES, BIS	Exchange, ActiveSync, Mac OS Address Book	Exchange, Domino, BlackBerry, iSync	Synergy	Exchange, Domino, BlackBerry, ActiveSync
Multitaskin	Sí	Sí	No	Sí	Sí	Sí
Copiar/Pegar	Sí	Sí	Sí	Sí	Sí	Sí
Ecosistema/Soporte multimedia	Amazon	iTunes sin DRM	iTunes	Ovi	Amazon	Windows Media Player
Actualización del firmware	OTA	Tethered, OTA	Tethered	Tethered, OTA	¿?	Tethered, OTA
Motor navegador	Webkit	Propietario	Webkit	Webkit	Webkit	Internet Explorer
Thethering	Sí	Sí	Sí	Sí	Sí	Sí
Bluetooth estéreo	Sí	Sí	Sí	Sí	Sí	Sí

Fig. 1.1 Tabla comparativa sistemas operativos móviles.

Pero sin duda uno de los puntos fuertes de los teléfonos inteligentes es poder incluir nuevas funcionalidades y nuevas aplicaciones. Para ello es importante que la plataforma admita desarrollo de terceros. En este sentido todas las plataformas ponen a disposición de los desarrolladores el SDK que permite desarrollar aplicaciones para la plataforma en cuestión. Todas estas aplicaciones desarrolladas por terceros deben estar disponibles en algún lugar de la red. Los terminales iPhone disponen de AppStore. Android utiliza Android Market. Windows y Palm han seguido los mismos pasos y ya han lanzado mercados similares.

	Android	BlackBerry OS 4.7	iPhone OS 3.0	S60 5th Edition	Palm WebOS	Windows Mobile 6.5
Disponibilidad del SDK	Sí	Sí	Sí	Sí	Sí	Sí
Tienda de aplicaciones	Sí	Próximamente	Sí	Próximamente	Sí	Sí
Disponibilidad de aplicaciones	Alta	Mediana	Alta	Mediana	Baja	Alta
Aplicaciones nativas	Sí	No	Sí	Sí	No	Sí
Administración local de aplicaciones	Excelente	Buena	Excelente	Buena	Excelente	Buena

Fig. 1.2 Tabla comparativa desarrollo en SO móviles

Debido a la gran demanda de los teléfonos Android, así como la facilidad de poner a disposición de los demás las aplicaciones y el hecho de que no sea un sistema operativo que dependa de un hardware concreto, nos centraremos en este sistema operativo para realizar el proyecto

1.4. Arquitectura

El proyecto final tiene tres partes bien diferenciadas, tal y como se puede ver en la Fig. 1.3.



Fig. 1.3 Arquitectura del proyecto

La parte principal consiste en la aplicación Android en sí. Manejo de entrada y salida de datos. Básicamente es la forma que tiene el usuario de interactuar con el programa, con distintos menús y opciones.

La parte de OSM consiste en el uso del servidor de mapas. Se utiliza un servicio de localización llamado Nominatim, que permite geolocalización inversa, y un servicio de descarga de mapas denominado XAPI, que permite la descarga de mapas sobre los que implementar el encaminamiento.

El servicio de encaminamiento se realiza sobre el mapa que nos devuelve la XAPI. En esta parte se extraen todos los puntos de interés del mapa. Una vez se tienen todos los nodos se calcula, mediante el algoritmo de Dijkstra, el camino más corto entre dos nodos concretos. Se construye un camino con los distintos nodos que se recorren y se muestran la ruta en el mapa.

1.5. Proyectos similares

1.5.1. WalkyTalky e Intersection

Google ha presentado dos aplicaciones para Android diseñadas para ayudar a los ciegos a caminar siguiendo las indicaciones de direcciones de Google Maps con una tecnología de navegación GPS. Las aplicaciones, WalkyTalky e Intersection Explorer, utilizan indicaciones por voz de recorridos a pie de Google Maps dando a los ciegos la oportunidad de explorar el diseño de las calles antes de caminar por el mundo físico.

WalkyTalky es una aplicación de indicaciones de direcciones por voz, mientras que Intersection Explorer proporciona exploración táctil.

La primera aplicación, WalkyTalky, proporciona acceso directo al componente de direcciones a pie de Google Maps. A diferencia de la aplicación de navegación normal de Android, WalkyTalky pronuncia en alto los nombres de las calles e intersecciones mientras caminamos por y a través de ellas.

La segunda aplicación, Intersection Explorer, es una aplicación de exploración táctil. La aplicación se inicia en nuestra ubicación inicial (se puede cambiar el lugar de partida introduciendo una dirección diferente). A continuación, se puede tocar la pantalla y mover el dedo por la aplicación para “explorar” la zona.

A medida que “exploramos”, la aplicación pronuncia las direcciones y las calles sobre las que va pasando nuestro dedo. Si nos perdemos, también podemos mover el dedo en círculo por la pantalla y la aplicación nos dirá el nombre de cada calle sobre la que vaya pasando nuestro dedo, junto con su dirección de brújula correspondiente.



Fig.1.4 Captura programa Intersection.

1.5.2. Kapten for Smartphone

Kapten for Smartphone de Kapsys es una solución completa de ayuda a la movilidad, disponible para smartphones con GPS. Ofrece a las personas en situación de minusvalía visual una valiosa ayuda, permitiéndoles ubicarse y desplazarse con mayor facilidad. Totalmente embarcada y, por tanto, sin necesidad de acceso de pago a la red, la aplicación ofrece Servicios integrados de localización y navegación peatonal vocales 100% accesibles.

Kapten for Smartphone integra navegación vocal inteligente, navegación “libre” y lectura de mapas urbanos. Integra una base Premium de puntos de interés que incluye miles de direcciones pregrabadas (farmacias, comisarías, hospitales, restaurantes, etc.).

Disponible en el transcurso del segundo trimestre de 2011 en plataforma Windows Mobile, también podrán adquirirse, a finales de 2011, en versión Android.



Fig. 1.5 Kaptén

1.5.3. Blind

Este proyecto no es un proyecto de movilidad en sí, pero es interesante ver las funcionalidades que ayudan al uso del terminal a personas invidentes.

Este programa utiliza la función TextToSpeech para poder gestionar el uso del terminal. La primera vez que se accede a Blind te solicita instalar las librerías TextToSpeech, y se muestra una pantalla en negro con el logotipo de la aplicación.

Se utiliza el teléfono de forma apaisada, para poder usar el TrackBall, y moviendo de izquierda a derecha se va pasando por las opciones del programa. Si se pulsa en la esquina superior izquierda de la pantalla se puede volver al menú. Si se pulsa la esquina superior derecha se repite la opción en la que se encuentra.

Con Blind se puede acceder a la lista de contactos, los cuales dicta a medida que se vaya moviendo el TrackBall de arriba abajo. Para realizar una llamada basta con hacer click cuando te encuentras en el contacto deseado.

Otra posibilidad es la de enviar mensajes de texto, seleccionando el contacto y dictando el mensaje. El mensaje se escribe utilizando también el TrackBall, el cual irá dictando las distintas letras del abecedario.

Estas son algunas entre muchas de las posibilidades que te brinda Blind (gestionar canciones, comprobar localización GPS, conocer el estado de la batería...).

En la figura 1.6 se puede ver la única pantalla de la aplicación. Puede parecer chocante, pero al fin y al cabo consiste en una aplicación diseñada para escuchar, y por raro que parezca lo visual queda en un segundo plano.



Fig. 1.6 Aplicación Blind.

Esta aplicación no es una aplicación de movilidad en sí, pero se ha considerado interesante comentarla por lo ingenioso que resulta el manejo de la misma.

CAPÍTULO 2. DESARROLLO APLICACIONES ANDROID

2.1. Instalación del entorno de desarrollo: SDK + Eclipse

Para poder trabajar con Android se tiene que instalar el entorno de desarrollo Eclipse y el SDK de Android.

Para poder configurar el SDK de Android se debe instalar el paquete SDK de arranque, que se puede encontrar en la página de <http://developer.android.com/sdk/index.html>. Una vez instalado el paquete, se instala el plug-in para Eclipse ADT.

Antes de comenzar con el SDK, se debe confirmar que se tiene instalado el JDK. Una vez comprobado, se descarga el motor de arranque paquete SDK de la página de descarga del SDK. Se descomprime el archivo .zip en un lugar del equipo. Se anota el nombre y la ubicación del directorio de SDK en el sistema, ya que se tendrá que hacer referencia al directorio SDK más tarde.

Android ofrece un plugin para el IDE de Eclipse, ADT, así se puede configurar rápidamente nuevos proyectos de Android, crear una aplicación de interfaz de usuario, depurar sus aplicaciones utilizando las herramientas SDK de Android.

Para descargar el ADT, hay que ir a StartEclipse, Help -> Install New Software. Se da click en Add, y en el diálogo que aparece se pone ADT Plugin en el Nombre y en Location la URL <https://dl-ssl.google.com/android/eclipse/>.

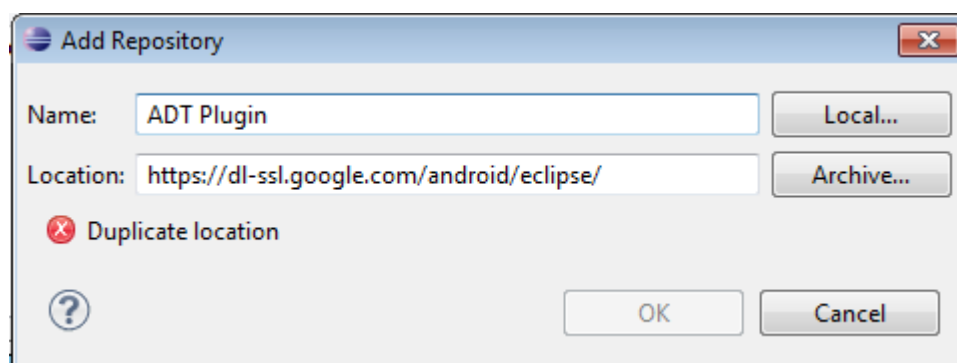


Fig. 2.1 Ventana descarga ADT.

En el diálogo que aparece, se hace click en el checkbox y se pasa a la siguiente ventana, donde se mostrará una lista con las herramientas que se descargarán. (Fig. 2.2 y 2.3).

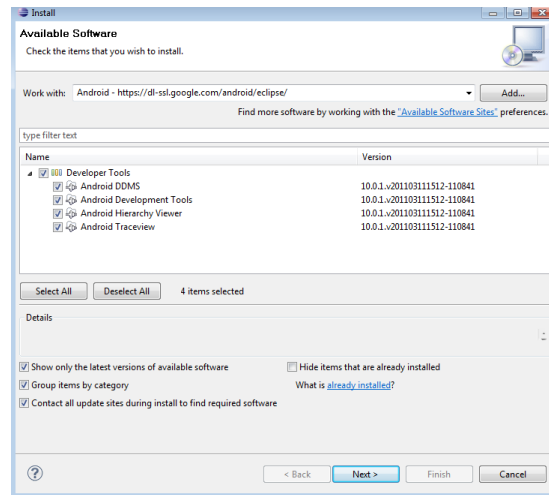


Fig. 2.2 Instalación ADT.

Name	Version	Id
<input checked="" type="checkbox"/> Android DDMS	10.0.1.v201103111...	com.android.ide.eclipse.ddms.feature.gr...
<input checked="" type="checkbox"/> Android Development Tools	10.0.1.v201103111...	com.android.ide.eclipse.adt.feature.group
<input checked="" type="checkbox"/> Android Hierarchy Viewer	10.0.1.v201103111...	com.android.ide.eclipse.hierarchyviewer...
<input checked="" type="checkbox"/> Android Traceview	10.0.1.v201103111...	com.android.ide.eclipse.traceview.featur...

Fig.2.3 Instalación ADT

En la siguiente ventana se acepta la licencia y se finaliza.

Reabrimos Eclipse. Para configurar el ADT Plugin, vamos a Window -> Preferences para abrir el Panel de preferencias. Se selecciona Android, y en la SDK Location se busca el directorio donde se ha guardado el SDK. Se da a Aplicar y se pulsa OK.

Una vez que tenemos instalado y configurado el ADT, se tiene que descargar los componentes. Para ello, vamos a Ventana -> Android SDK y ADV manager, vamos a la pestaña Available packages, seleccionamos los paquetes disponibles e instalamos.

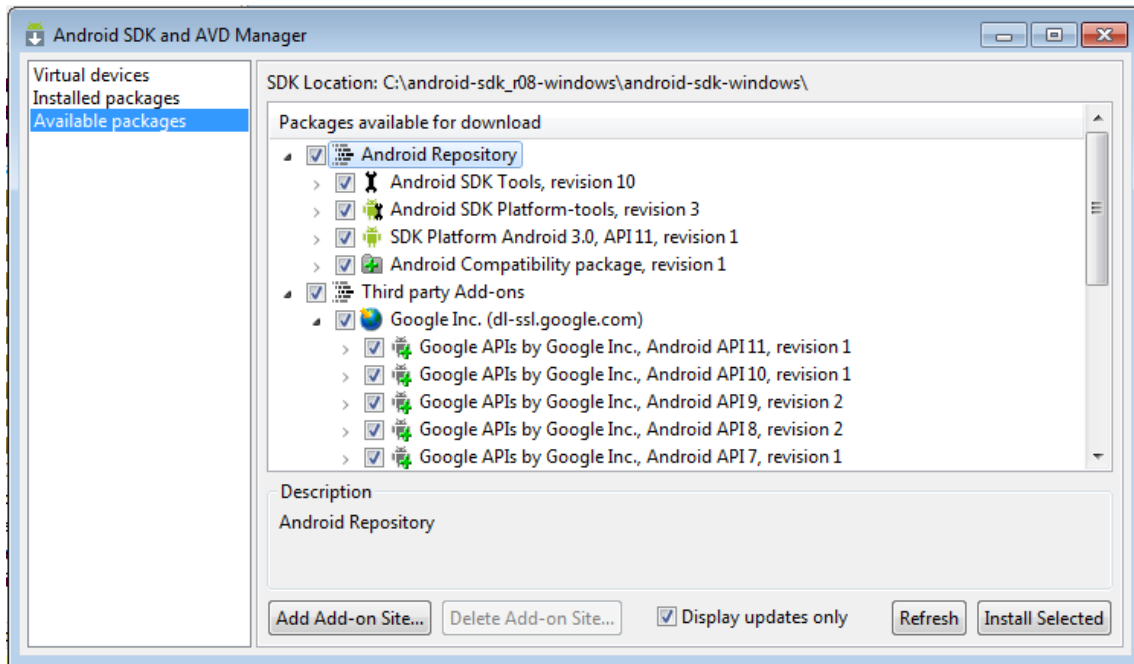


Fig. 2.4 Descarga de componentes

Una vez se llega a este punto ya se puede empezar a desarrollar en Android con Eclipse.

2.2. Estructura de un proyecto android

Para poder comprender cómo se construye una aplicación Android vamos a echar un vistazo a la estructura general de un proyecto.

Cuando creamos un nuevo proyecto Android en Eclipse se genera automáticamente la estructura de carpetas necesaria para poder generar posteriormente la aplicación, tal y como vemos en la figura 2.5 que se muestra a continuación. Esta estructura será común a cualquier aplicación, independientemente de su tamaño y complejidad.

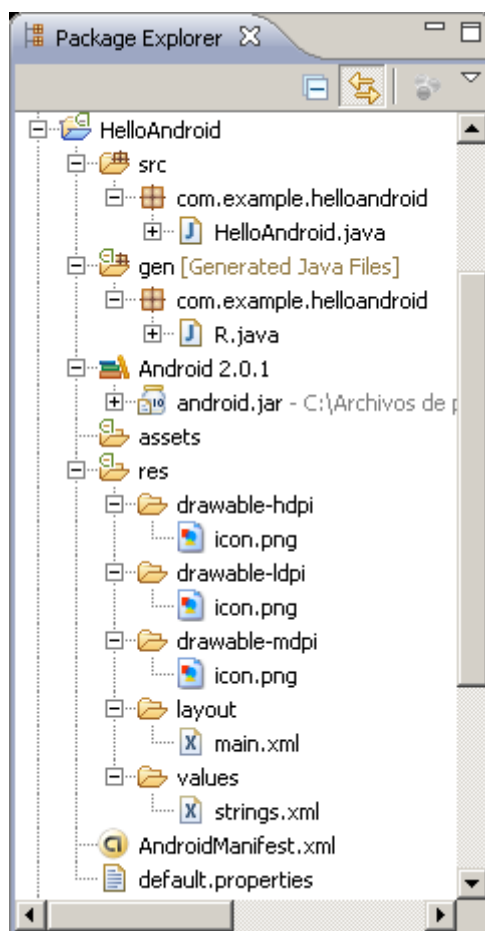


Fig. 2.5 Estructura de proyecto Android

2.2.1. Carpeta /src/

Contiene todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc. Inicialmente, Eclipse creará por nosotros el código básico

de la pantalla (Activity) principal de la aplicación, siempre bajo la estructura del paquete java definido.

2.2.2. Carpeta /res/

Contiene los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc. Los diferentes tipos de recursos se distribuyen en las siguientes carpetas:

- /res/drawable/. Contiene las imágenes de la aplicación. Se puede dividir en /drawable-ldpi, /drawable-mdpi y /drawable-hdpi, para utilizar diferentes recursos dependiendo de la resolución del dispositivo.
- /res/layout/. Contiene los ficheros de definición de las diferentes pantallas de la interfaz gráfica. Se puede dividir en /layout y /layout-land, para definir distintos layouts dependiendo de la orientación del dispositivo.
- /res/anim/. Contiene la definición de las animaciones utilizadas por la aplicación.
- /res/menú/. Contiene la definición de los menús de la aplicación.
- /res/values/. Contiene otros recursos de la aplicación como por ejemplo cadenas de texto, estilos, colores, etc.
- /res/xml/. Contiene los ficheros XML utilizados por la aplicación.
- /res/raw/. Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.

2.2.3. Carpeta /gen/

Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que generamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente en java dirigidos al control de los recursos de la aplicación. El más importante es el fichero R.java, y la clase R.

Esta clase R contendrá en todo momento una serie de constantes con los ID de todos los recursos de la aplicación incluidos en la carpeta /res/, de forma que podamos acceder fácilmente a estos recursos desde nuestro código a través de este dato. Así, por ejemplo, la constante R.drawable.icon contendrá el ID de la imagen "icon.png" contenida en la carpeta /res/drawable.

2.2.4. Carpeta /assets/

Contiene todos los demás ficheros auxiliares necesarios para la aplicación, como ficheros de configuración, de datos, etc.

La diferencia entre los recursos incluidos en la carpeta /res/raw/ y los incluidos en la carpeta /assets/ es que para los primeros se generará un ID en la clase R y se deberá acceder a ellos con los diferentes métodos de acceso a recursos. Para los segundos sin embargo no se generarán ID y se pondrá acceder a ellos por su ruta como a cualquier otro fichero del sistema. Se usará uno u otro según las necesidades de la aplicación.

2.2.5. Fichero AndroidManifest.xml

Contiene la definición en XML de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, versión, icono), sus componentes (pantallas, mensajes, etc.), o los permisos necesarios para su ejecución. Algunas de las opciones que se pueden definir en el manifest son:

- Package: Situación de los ficheros que se ejecutan.
- Uses-permission: Permisos que se le otorgan a la aplicación que por defecto no tiene.
- Uses-library: Librerías de Google.
- Activity: Permite que se inicie una actividad. Todas las activities deben estar especificadas en el AndroidManifest.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="osmalib.test"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name" android:name="MyTest" android:debuggable="true">
        <activity android:name="Buscar2"/>
        <activity android:name="Buscar"/>
        <activity android:name="ListaDirecciones"/>
        <activity android:name=".Intro" android:label="LazAndroid">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".OSMapLibraryActivity"/>
        <activity android:name="DemoRouting"/>
        <activity android:name="coordenada"/>
        <activity android:name="DibujarRuta"/>
    </application>

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

Fig. 2.6 Ejemplo de fichero AndroidManifest.xml

2.3. Utilidades usadas

2.3.1. Text To Speech

Text To Speech (TTS) es una librería que permite a los desarrolladores añadir voz a sus aplicaciones. Esta operación es mucho más sencilla de lo que parece y viene incluida por defecto en el SDK de android desde la versión 1.6.

2.3.2. Speech To Text

Se trata del caso contrario que el Text To Speech. En esta ocasión podremos convertir una entrada de voz en texto.

Tanto el Text To Speech como el Speech To Text se utilizarán para poder facilitar el manejo de la aplicación sin necesidad de escribir o leer.

CAPÍTULO 3. ESTUDIO DE API DE MAPAS PARA ANDROID

3.1. Introducción

Una API (Application Programming Interface) consiste en un conjunto de librerías de clases que permiten ser usadas de forma fácil y rápida. De esta manera se pueden utilizar y manejar muchos elementos en las aplicaciones que ya han sido programados sin tener que empezar desde cero.

A continuación se explicará por separado la API de GoogleMaps y de OSM, que son los dos tipos de mapas que se han utilizado en este proyecto.

3.2. Google Maps

El complemento para las API de Google es una ampliación del entorno de desarrollo del SDK de Android que proporciona a las aplicaciones de Android un acceso sencillo a los servicios a los datos de Google. La función principal del complemento es la biblioteca externa de Google Maps, que permite añadir funciones de asignación potentes a la aplicación de Android.

Para utilizar el complemento, se debe instalar en el SDK de Android. Desde ahí, se puede acceder a las clases de la biblioteca de Google Maps y compilar la aplicación con respecto a ellas.

Para poder desarrollar con la API de Google Maps es necesario una API key. Esa clave se deberá insertar en el código de la aplicación para que cada vez que se utilicen los mapas el servidor de Google pueda ver que está registrado con el servicio. En el caso de que no se disponga de una clave o no sea válida, la aplicación sigue funcionando, pero no se muestran los mapas.

3.2.1. Pasos para conseguir la API Key de Google

Para conseguir la clave de Google debemos crear una huella digital de certificado llamada MD5. Para conseguir la MD5 se debe ejecutar la consola de comandos de Windows. Se coloca en la carpeta bin de Java, donde se encuentra el keytools, y se introduce el comando `"keytool -list -keystore dir"`, siendo `"dir"` la ubicación de la carpeta `.android` en donde se encuentra el archivo `debug.keystore`.

```

C:\Archivos de programa\java\jdk1.6.0_23\bin>keytool -list -keystore c:\Users\In
grid\.android\debug.keystore
Escriba la contraseña del almacén de claves:

***** ADVERTENCIA ADVERTENCIA ADVERTENCIA *****
* La integridad de la información almacenada en su almacén de claves *
* NO se ha comprobado. Para comprobar dicha integridad, *
* deberá proporcionar su contraseña de almacén de claves. *
***** ADVERTENCIA ADVERTENCIA ADVERTENCIA *****

Tipo de almacén de claves: JKS
Proveedor de almacén de claves: SUN

Su almacén de claves contiene entrada 1

androiddebugkey, 24-ene-2011, PrivateKeyEntry,
Huella digital de certificado (MD5): F6:80:67:FA:8A:3F:9F:58:76:94:91:E7:A0:D2:6
C:D9

```

Fig. 3.1 Pantalla para la obtención de la Huella digital.

Una vez se tiene la Huella digital, hay que ir a la página <http://code.google.com/intl/es-ES/android/add-ons/google-apis/maps-api-signup.html>, introducirla donde pone "My certificate's MD5 fingerprint:" y darle a Generate API Key.

☒ I have read and agree with the terms and conditions ([printable version](#))

My certificate's MD5 fingerprint:

Fig. 3.2 Generar API Key.

API de Google Maps

[Página principal de Google Code](#) > [API de Google Maps](#) > Suscripción al API de Google Maps

Gracias por suscribirte a la clave del API de Android Maps.

Tu clave es:

0Z8SxWvYzc3yvm1hLoy-y1vtpaD2ijKPmg60D-g

Esta clave es válida para todas las aplicaciones firmadas con el certificado cuya huella dactilar sea:

F6:80:67:FA:8A:3F:9F:58:76:94:91:E7:A0:D2:6C:D9

Incluimos un diseño xml de ejemplo para que puedas iniciarte por los senderos de la creación de mapas:

```

<com.google.android.maps.MapView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:apiKey="0Z8SxWvYzc3yvm1hLoy-y1vtpaD2ijKPmg60D-g"
/>

```

Consulta la [documentación del API](#) para obtener más información.

Fig. 3.3 API Key de Google Maps.

3.2.2. Objetos de mapas

A continuación se explicará algunos de los objetos y clases que manejan los mapas.

3.2.2.1. *GeoPoint*

La clase *GeoPoint* viene predefinida con el paquete de mapas. Esta clase almacena una posición en un mapa. Esta posición se representa mediante dos números de tipo *double*: latitud y longitud. Para pasar de estos dos números a *GeoPoint* solamente hace falta convertir los números a micro grados elevando cada número a la sexta potencia.

3.2.2.2. *MapActivity* y *MapController*

MapActivity es la clase que maneja las funciones básicas de una *Activity* que muestra un mapa, mientras que *MapController* permite gestionar la panorámica y el zoom del mapa.

3.2.2.3. *MapView*, *MyLocationOverlay* y *Overlay*

MapView es la vista que muestra el mapa, *MyLocationOverlay* permite dibujar la posición actual del usuario y *Overlay* permite insertar objetos para que aparezcan en el mapa.

3.2.3. Servicios ofrecidos

3.2.3.1. *Codificación geográfica*

La codificación geográfica es el proceso de transformar direcciones (como “Calle Concejal Santiago Falcón Pérez”) en coordenadas geográficas (latitud y longitud), que se pueden utilizar para colocar marcadores o situar el mapa. El API de Google Maps incluye un servicio de codificación geográfica al que se puede acceder directamente mediante solicitudes HTTP o un objeto *GClientGeocoder*.

Se puede modificar el geocoder del API de Google Maps para dar preferencia a los resultados situados dentro de una determinada ventana gráfica mediante el método *GClientGeocoder.setViewport()*. La ventana gráfica se expresa como un cuadro delimitador del tipo *GLatLngBounds*.

Otro dato interesante es la posibilidad de hacer que los resultados se devuelvan ajustados a medida de un dominio (país) concreto mediante el método `GClientGeocoder.setBaseCountryCode()`. Se puede enviar solicitudes de codificación geográfica para todos los dominios en los que la aplicación principal de Google Maps ofrezca la función de codificación geográfica. Como ejemplo, si se busca “Toledo” se devolverá distintos resultados en el dominio de España (<http://maps.google.es>), especificado con el código de país “es”, de los que se devuelve con el dominio predeterminado de Estados Unidos (<http://maps.google.com>).

3.2.3.2. Extracción de direcciones estructuradas

Si se desea acceder a información estructurada sobre una dirección, `GClientGeocoder` también proporciona un método `getLocations()` que devuelve un objeto JSON que consta de la siguiente información:

- Status
 - o request: El tipo solicitado (geocode).
 - o code: Código de respuesta que indica si la solicitud de codificación geográfica se ha realizado correctamente o no.
- Placemark: Si el geocoder encuentra varias coincidencias, es posible que se devuelvan varios marcadores de posición.
 - o Address: Una versión de la dirección con formato elegante y uso correcto de mayúsculas y minúsculas.
 - o AddressDetails: Dirección en formato xAL (Lenguaje de direcciones extensible), estándar internacional para el formato de direcciones.
 - o Point: Punto en el espacio 3D.
 - Coordinates: longitud, latitud y altitud de la dirección.

3.2.3.3. Codificación geográfica inversa

Consiste en traducir una dirección interpretable por humanos en un punto en el mapa.

El método `GClientGeocoder.getLocations()` es compatible tanto con la codificación geográfica estándar como con la codificación inversa. Si se transmite este método a un objeto `GLatLng` en lugar de una dirección String, el geocoder realizará una búsqueda inversa y devolverá un objeto JSON estructurado de la ubicación localizable mediante dirección más cercana.

3.2.3.4. Codificación geográfica mediante solicitudes HTTP

Otra manera de acceder al geocoder del API de Google Maps es directamente mediante secuencias de comandos de servidor. No se recomienda el uso de este método frente al uso del geocoder de cliente, pero resulta útil con fines de depuración, en aquellos casos en los que no ha objeto GClientGeocoder de JavaScript disponible.

Para acceder al geocoder del API de Google Maps, se envía una solicitud a <http://maps.google.com/maps/geo?> Con los siguientes parámetros en el URI:

- q(obligatorio): La dirección a la que deseas asignar un identificador geográfico.
- key(obligatorio): La clave del API.
- sensor(obligatorio): Indica si la solicitud de codificación geográfica proviene o no de un dispositivo con sensor de ubicación. Este valor debe ser true o false.
- output(obligatorio): Formato en el que se debe generar el elemento de salida. Las opciones son xml, kml, csv o json.
- ll(opcional): el par (latitud, longitud) del centro de la ventana gráfica expresado como una cadena separada por comas (por ejemplo, "ll=40.479581, -117.773438).
- spn(opcional): Amplitud de la ventana gráfica expresada como una cadena separada por comas de un par (latitud, longitud).
- gl(opcional): Código del país, especificado como un valor de dos caracteres ccTLD ("dominio de nivel superior").

En el siguiente ejemplo, se solicitan las coordenadas geográficas de la Avenida Pineda de Castelldefels:

<http://maps.google.com/maps/geo?q=32+Avenida+de+la+Pineda+32,+Castelldefels,+Spain&output=xml&sensor=true&key=abcdefg>

Si se especifica json como formato de salida, la respuesta se proporciona en forma de objeto JSON. Si se especifica xml o kml, la respuesta se devuelve en código KML.

En la figura 3.4 se puede ver la respuesta a dicha llamada en formato kml, y en la figura 3.5 la respuesta para la misma llamada en formato json.

```

-<kml>
  -<Response>
    <name>Avenida de la Pineda 32, Castelldefels, Spain</name>
    <Status>
      <code>200</code>
      <request>geocode</request>
    </Status>
    -<Placemark id="p1">
      -<address>
        Avinguda de la Pineda, 32, 08860 Castelldefels, España
      </address>
      -<AddressDetails Accuracy="8">
        -<Country>
          <CountryNameCode>ES</CountryNameCode>
          <CountryName>Espanya</CountryName>
        -<AdministrativeArea>
          <AdministrativeAreaName>Catalunya</AdministrativeAreaName>
          -<SubAdministrativeArea>
            <SubAdministrativeAreaName>Barcelona</SubAdministrativeAreaName>
            -<Locality>
              <LocalityName>Castelldefels</LocalityName>
              -<Thoroughfare>
                <ThoroughfareName>Avinguda de la Pineda, 32</ThoroughfareName>
                </Thoroughfare>
              -<PostalCode>
                <PostalCodeNumber>08860</PostalCodeNumber>
                </PostalCode>
              </Locality>
            </SubAdministrativeArea>
          </AdministrativeArea>
        </Country>
      </AddressDetails>
      -<ExtendedData>
        <LatLonBox north="41.2758958" south="41.2696006" east="1.9836821"
          west="1.9773868"/>
      </ExtendedData>
      -<Point>
        <coordinates>1.9805433,41.2727512,0</coordinates>
      </Point>
    </Placemark>
  </Response>
</kml>

```

Fig. 3.4 Respuesta kml mediante solicitudes HTTP

```

{
  "name": "32 Avenida de la Pineda 32, Castelldefels, Spain",
  "Status": {
    "code": 200,
    "request": "geocode"
  },
  "Placemark": [ {
    "id": "p1",
    "address": "Avinguda de la Pineda, 32, 08860 Castelldefels, España",
    "AddressDetails": {
      "Accuracy" : 8,
      "Country" : {
        "AdministrativeArea" : {
          "AdministrativeAreaName" : "Catalunya",
          "SubAdministrativeArea" : {
            "Locality" : {
              "LocalityName" : "Castelldefels",
              "PostalCode" : {
                "PostalCodeNumber" : "08860"
              },
              "Thoroughfare" : {
                "ThoroughfareName" : "Avinguda de la Pineda, 32"
              }
            },
            "SubAdministrativeAreaName" : "Barcelona"
          }
        },
        "CountryName" : "Espanya",
        "CountryNameCode" : "ES"
      }
    },
    "ExtendedData": {
      "LatLonBox": {
        "north": 41.2758958,
        "south": 41.2696006,
        "east": 1.9836821,
        "west": 1.9773868
      }
    },
    "Point": {
      "coordinates": [ 1.9805433, 41.2727512, 0 ]
    }
  } ]
}

```

Fig. 3.5 Respuesta json mediante solicitudes HTTP

3.2.4. Términos de licencia

El uso de Google Geocoding API está sujeto a un límite de consultas de 2500 solicitudes de codificación geográfica al día. Este límite se aplica para evitar el abuso o el uso indebido de Geocoding API y puede cambiarse en el futuro sin previo aviso. Además, se aplica un límite de índice de solicitudes para evitar un uso inadecuado de este servicio. Si se supera el límite en un plazo de 24 horas o se abusa de cualquier otra forma del servicio, Geocoding API puede dejar de funcionar temporalmente.

Geocoding API solo se puede utilizar junto con los resultados de visualización de un mapa de Google. Sin embargo, con los resultados de codificación geográfica que no se visualizan en un mapa no está permitido.

3.3. OSM

Debido a las limitaciones de Google Maps en cuanto a las licencias, se ha decidido realizar el proyecto utilizando OSM.

OpenStreetMap (OSM) es un proyecto colaborativo para crear mapas libres y editables.

El levantamiento de información en campo es realizado por voluntarios. Aprovechando sus desplazamientos a pie, en bicicleta o coche y utilizando un dispositivo GPS, van capturando las trazas y waypoints. El proyecto se fundamenta principalmente en el gran número de pequeñas ediciones realizadas por la mayoría de los contribuyentes, que corrigen errores o añaden nuevos datos al mapa.

3.3.1. Formato de datos

OpenStreetMap utiliza una estructura de datos topológica. Los elementos básicos de la cartografía OSM son:

- Nodos: Puntos que recogen una posición geográfica dada.
- Vías (Ways): Lista ordenada de nodos que representa una poli línea o polígono (cuando una poli línea empieza y finaliza en el mismo punto).
- Relaciones (Relations): Grupos de nodos, caminos y otras relaciones a las que se pueden asignar determinadas propiedades.
- Etiquetas (Tags): Se pueden asignar a nodos, caminos o relaciones y constan de una clave (key) y de un valor (value). Por ejemplo: highway=trunk.

3.3.2. Cálculo de rutas y navegación

El cálculo de las rutas óptimas utilizando los datos de OpenStreetMap no está totalmente desarrollado, sin embargo el avance en este sentido en los últimos meses ha sido muy importante. En numerosas regiones los datos existentes hasta la fecha todavía no son suficientemente detallados para que lleguen a ser plenamente fiables, ya que a menudo se carece de información sobre nombre de calles, o la situación de los números, por ejemplo.

Actualmente existen tres sitios web principales que ofrecen servicios piloto de enrutamiento mediante conocidos algoritmos de búsqueda en grafos (Dijkstra). En la mayoría de los casos estas implementaciones no tratan necesariamente el camino más corto, sino el de menor impedancia en función de las etiquetas OSM tenidas en cuenta.

3.3.3. Herramientas de edición

Para editar los datos existen diferentes posibilidades, la gran mayoría muy sencillas de utilizar. El usuario debe registrarse gratuitamente mediante una dirección de correo verificada para evitar el spam y el vandalismo. Si no se va a realizar edición alguna y sólo se desea visualizar la cartografía no es necesario realizar el registro.

Los datos que los contribuyentes han capturado con sus dispositivos GPS sirven como guía para dibujar sobre ellas las nuevas vías. Estos datos en bruto suelen cargarse desde el equipo local del usuario o solicitando al servidor OSM que nos descargue aquellas trazas de la zona que vamos a editar y que otros usuarios han subido previamente a OpenStreetMap.

Una vez se tiene la información geográfica básica sobre la que dibujar es el momento de añadir los elementos del mapa que se quieren representar mediante nodos y vías. A estos puntos y líneas se les asigna uno o varios atributos que los caracterizan. Por ejemplo, a una línea se le indica la etiqueta y la clave `highway: motorway` para señalar que es una autopista, y `name: autovía del Cantábrico` para indicar su nombre.

Para editar los datos es necesario un editor, y con este fin el proyecto OpenStreetMap facilita diferentes opciones:

- Editor online Potlatch. Es el más sencillo de utilizar.
- Editor offline JOSM. Actualmente es el editor más avanzado.
- Editor offline MerKaartor.

3.3.4. Servicios utilizados

3.3.4.1. Nominatim

Nominatim es una herramienta para buscar los datos de OSM por su nombre y dirección para generar direcciones de síntesis de los puntos de OSM. Se puede encontrar en <http://nominatim.openstreetmap.org>. Nominatim también se utiliza como una de las fuentes para el cuadro de búsqueda en la página principal de OpenStreetMap.

Algunos de los parámetros de búsqueda son:

- format = [html | xml | json]: Formato de salida.
- accept-language=<browser language string>: Orden de preferencia de idioma para mostrar resultados de búsqueda.
- q=<query>: Cadena de consulta para buscar.
- Countrycodes=<countrycode>: Limitar los resultados de búsqueda a un determinado país (o una lista de países).
- Viewbox=<let>, <top>, <right>, <bottom>: La zona preferida para encontrar resultados de búsqueda.
- Limit=<integer>: Limitar el número de resultados devueltos.

Como ejemplo, se puede ver que para la siguiente entrada:

<http://nominatim.openstreetmap.org/search?q=135+pilkington+avenue,+birmingham&format=xml&polygon=1&addressdetails=1>

nominatim nos devuelve el resultado que se observa en la Fig. 3.6.

```
<searchresults timestamp="Sat, 16 Apr 11 14:19:31 -0400" attribution="Data
Copyright OpenStreetMap Contributors, Some Rights Reserved. CC-BY-SA
2.0." querystring="135 pilkington avenue, birmingham" polygon="true"
exclude_place_ids="2061235323"
more_url="http://open.mapquestapi.com/nominatim/v1/search?format=xml&excl
ude_place_ids=2061235323&accept-
language=&polygon=1&addressdetails=1&q=135+pilkington+avenue%2C+birmi
ngham">
- <place place_id="2061235323" osm_type="way" osm_id="90394480"
place_rank="30" boundingbox="52.5487442016602,52.5488510131836,-
1.81651306152344,-1.81634628772736" polygonpoints="[["-
1.816513","52.5487566"],["-1.8164913","52.548824"],["-
1.8164685","52.5488213"],["-1.8164599","52.5488481"],["-
1.8163464","52.5488346"],["-1.8163717","52.5487561"],["-
1.816429","52.5487629"],["-1.816434","52.5487473"],["-
1.816513","52.5487566"]]" lat="52.5487969264788" lon="-1.81642935385411"
display_name="135, Pilkington Avenue, Castle Vale, City of Birmingham, West
Midlands, England, B72 1LH, United Kingdom" class="place" type="house">
<house_number>135</house_number>
<road>Pilkington Avenue</road>
<suburb>Castle Vale</suburb>
<city>City of Birmingham</city>
<county>West Midlands</county>
<state_district>West Midlands</state_district>
<state>England</state>
<postcode>B72 1LH</postcode>
<country>United Kingdom</country>
<country_code>gb</country_code>
</place>
</searchresults>
```

Fig. 3.6 Respuesta Nominatim

3.3.4.2. Reverse Geocoding

Reverse geocoding genera una dirección de una latitud y longitud. En este caso se repiten los parámetros “format” y “accept-language”, a los que hay que añadir:

- Osm_id=<value>: Un nodo específico osm /camino/ relacionado con la dirección que debe devolver.
- Lat=<valor>, long=<valor>: La ubicación para generar la dirección.
- Zoom=[0-18]: Nivel de detalle requerido, en donde 0 es el país y 18 es la casa/edificio.

Para la siguiente llamada:

<http://nominatim.openstreetmap.org/reverse?format=xml&lat=52.5487429714954&lon=-1.81602098644987&zoom=18&addressdetails=1>

Se obtiene el siguiente resultado de la Fig.3.7:

```
<reversegeocode timestamp="Thu, 28 Apr 11 17:39:36 -0400" attribution="Data Copyright
OpenStreetMap Contributors, Some Rights Reserved. CC-BY-SA 2.0."
querystring="format=xml&lat=52.5487429714954&lon=-
1.81602098644987&zoom=18&addressdetails=1">
  <result place_id="2061235282" osm_type="way" osm_id="90394420"
    lat="52.5487800131654" lon="-1.81626922291265">
      137, Pilkington Avenue, Castle Vale, City of Birmingham, West Midlands, England, B72
      1LH, United Kingdom
    </result>
  <addressparts>
    <house_number>137</house_number>
    <road>Pilkington Avenue</road>
    <suburb>Castle Vale</suburb>
    <city>City of Birmingham</city>
    <county>West Midlands</county>
    <state_district>West Midlands</state_district>
    <state>England</state>
    <postcode>B72 1LH</postcode>
    <country>United Kingdom</country>
    <country_code>gb</country_code>
  </addressparts>
</reversegeocode>
```

Fig. 3.7 Respuesta Reverse Geocoding

3.3.4.3. XAPI

La xapi es un API de sólo lectura, basada en una versión modificada del estándar de API, que proporciona una búsqueda mejorada y capacidades de consulta.

Hay varias instancias del servicio xapi ejecutándose en diferentes servidores, aunque últimamente no están funcionando muy bien.

Se ha desarrollado recientemente una nueva versión Java de XAPI, que es la que se utilizará en este proyecto, cuya dirección URL es <http://jxapi.openstreetmap.org/xapi/api/0.6/>.

La consulta a los mapas es idéntica que la vista hasta ahora, y devuelve:

- Todos los “nodes” que se encuentran dentro de una “Bounding box” dada, y cualquier relación que los reference.
- Todos los “ways” que hagan referencia al menos a un nodo que esté dentro de la bounding box, cualquier relación que los reference, y cualquier nodo fuera del bounding box que pueda hacer referencia a el “way”.

Una posible llamada a la XAPI sería la siguiente:

http://jxapi.openstreetmap.org/xapi/api/0.6/*%5Bbbox=1.9803,41.2731,1.9918,41.2767%5D

El campo bbox se corresponde con el Bounding Box, [BBox = left, bottom, right, top], que define un cuadro delimitador que se utiliza para limitar la búsqueda del mapa que se quiere descargar.

El archivo que se devuelve es demasiado grande como para ponerlo aquí. En él se muestran todos los puntos de interés del mapa, su descripción, así como el usuario que los ha subido.

3.3.5. Pequeña contribución a OpenStreetMap

Con el fin de contribuir en el proyecto de OpenStreetMap hemos cartografiado el Parc Mediterrani de la Tecnologia. Para ello se ha utilizado dos GPS Garmin eTrex Vista HCx, como el que se muestra en la Fig. 3.8, y el programa de edición JOSM.



Fig. 3.8 GPS Garmin eTrex Vista HCx.

En la figura 3.9 se muestra una captura del programa JOSM enfocando a la zona del Parc Mediterrani de la Tecnologia.

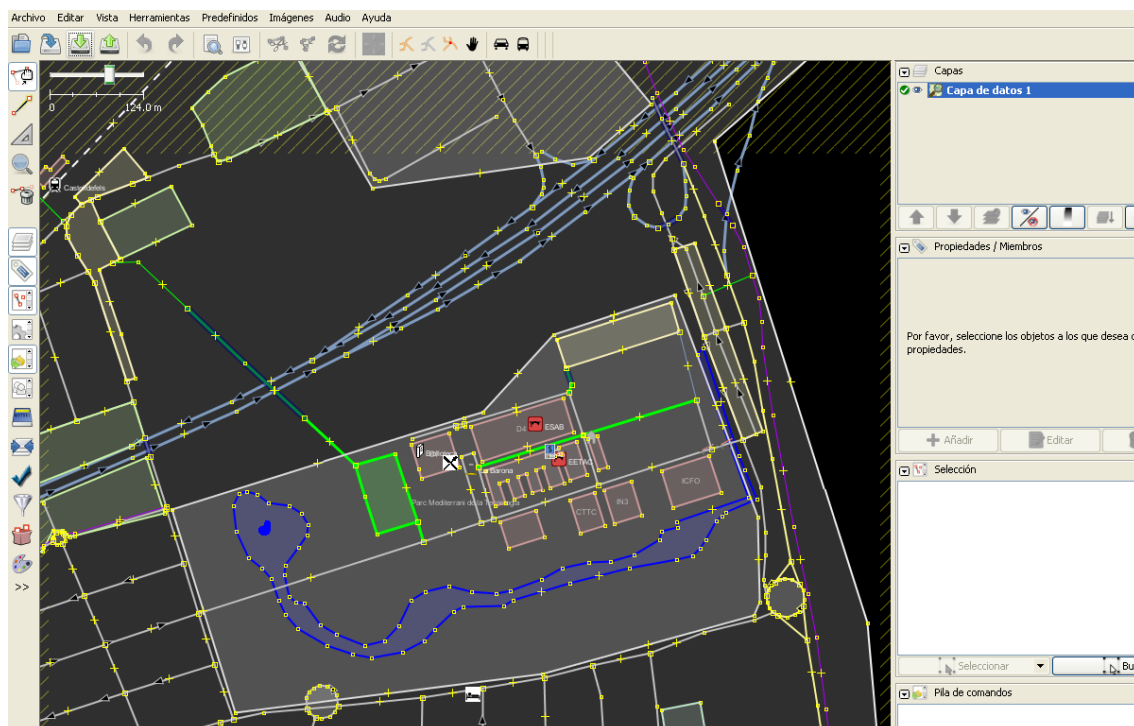


Fig. 3.9 Captura programa JOSM.

En las siguientes figuras se puede observar el cambio de antes del mapeo y después de subir los datos.

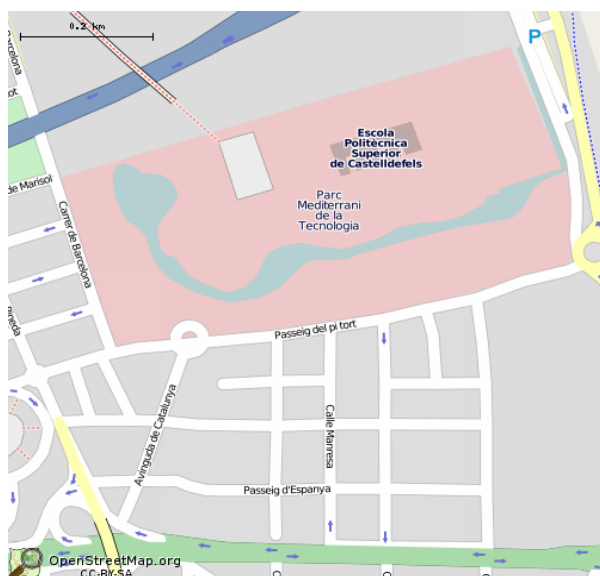


Fig. 3.10 Mapa antes del cartografiado

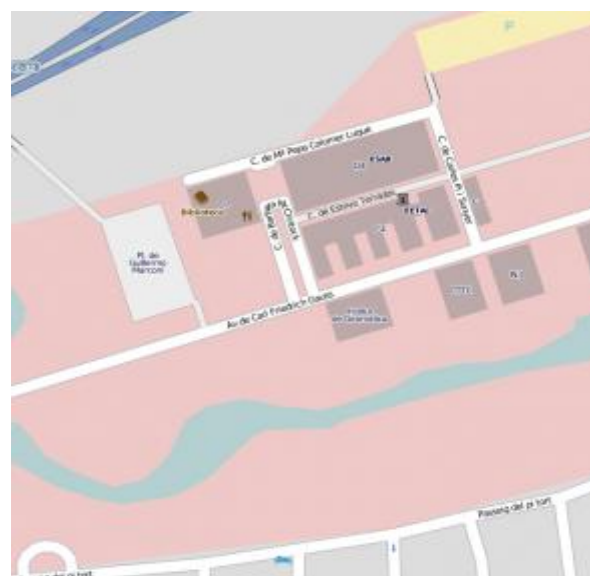


Fig. 3.11 Mapa después del cartografiado.

CAPITULO 4. LAZANDROID: LA APLICACIÓN

4.1. Google Maps

Se ha decidido hacer la aplicación sobre OSM, no obstante antes se desarrolla una pequeña aplicación con Google Maps, ya que es más sencillo de desarrollar y servirá como toma de contacto con Android y ver todas sus posibilidades.

La estructura del programa es la siguiente:

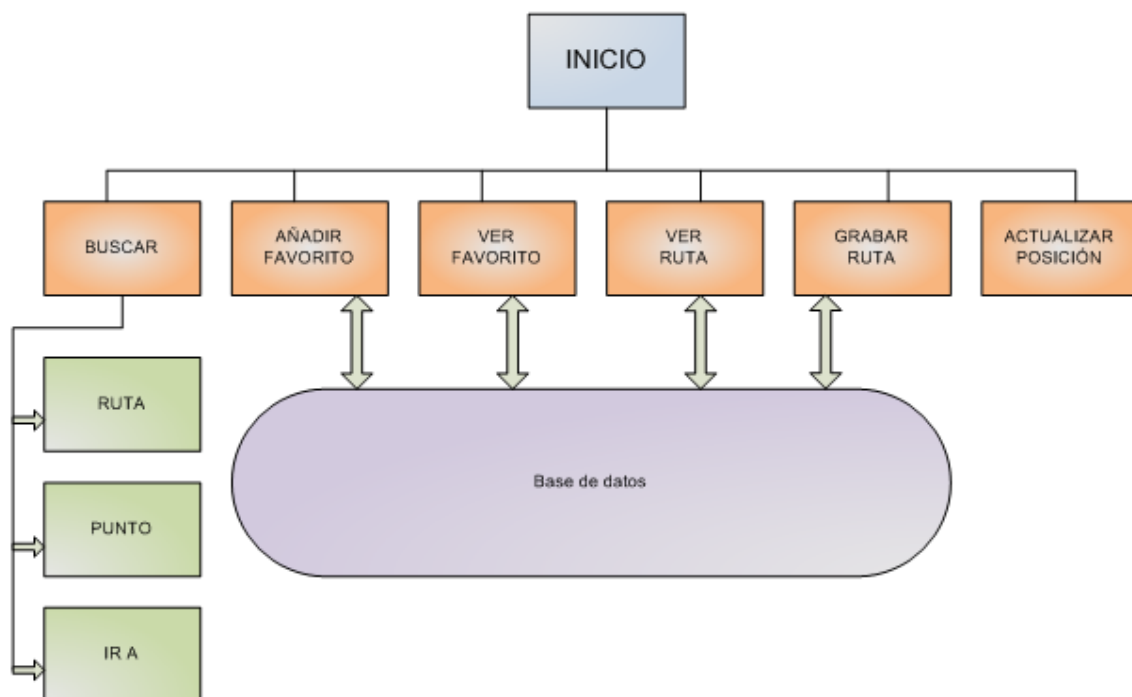


Fig. 4.1 Estructura programa GoogleMaps.

Para la base de datos se ha optado por Perst Lite, una implementación en código abierto para J2ME de una base de datos orientada a objetos.

A continuación se muestra algunas capturas de la aplicación con una breve explicación.



Fig. 4.2 Pantalla de inicio.

En la Fig. 4.2 se observa la pantalla inicial, en la cual se espera a recibir la señal GPS. Se tiene la opción de hacer una búsqueda, añadir la posición actual como favorita, ver rutas o puntos favoritos guardados, grabar una ruta nueva o forzar la actualización de la posición actual.

Se tiene la opción de buscar un destino desde la posición actual. Para ello se hace una llamada a la API de Google Earth. Esta llamada nos devuelve las indicaciones para poder llegar, que podrán ser utilizadas con el Text To Speech para que el usuario se pueda guiar. La pantalla de búsqueda se puede ver en la Fig. 4.3.



Fig. 4.3 Pantalla "Ir a"

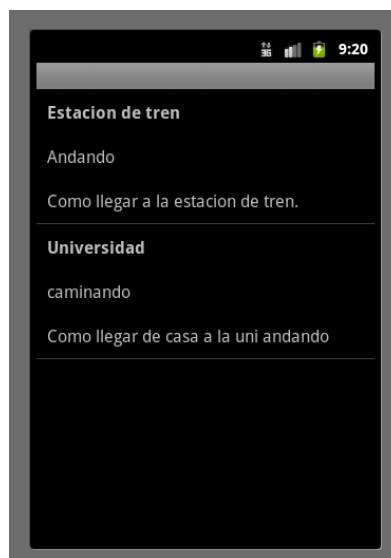
Hay posibilidad de grabar la ruta que vas a hacer y guardarla con una pequeña descripción. Así se tiene guardadas las rutas. Sólo se llegó a implementar guardar las posiciones, pero podrían guardarse también las indicaciones para poder escuchar cómo llegar a un sitio sin necesidad de recorrer la ruta en tiempo real. En la Fig. 4.4 se muestra cómo se guarda las rutas.



The screenshot shows a mobile application interface for saving a route. At the top, the status bar displays the time as 9:22. The form has three main sections: 'Nombre del recorrido' (Route Name) with a text input field containing 'Estacion de tren'; 'Tipo de actividad' (Activity Type) with a dropdown menu showing 'Andando'; and 'Descripción del recorrido' (Route Description) with a larger text input field containing 'Como llegar a la estacion de tren.'. At the bottom of the form are two buttons: 'Cancelar' (Cancel) and 'Guardar' (Save).

Fig. 4.4 Guardar Ruta.

En la Fig. 4.5 se puede ver una lista de las rutas que hay guardadas en la base de datos.



The screenshot displays a list of saved routes. The first route is titled 'Estacion de tren' (Train Station), with the activity type 'Andando' (Walking) and the description 'Como llegar a la estacion de tren.'. The second route is titled 'Universidad' (University), with the activity type 'caminando' (Walking) and the description 'Como llegar de casa a la uni andando'. The status bar at the top shows the time as 9:20.

Fig. 4.5 Lista de rutas guardadas.

Otra posibilidad es la de guardar un lugar concreto como favorito, tal y como se muestra en la Fig. 4.6 Esta opción es útil para facilitar una búsqueda que vaya a ser utilizada con frecuencia.

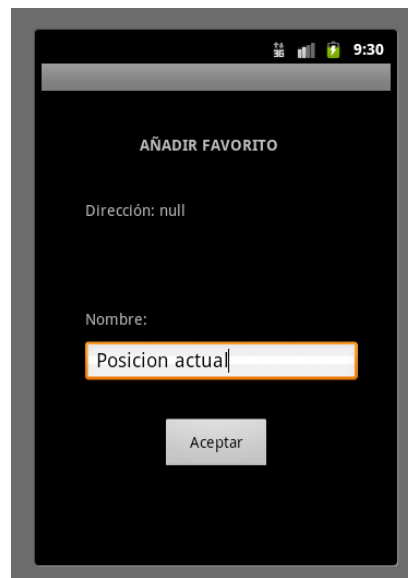


Fig. 4.6 Guardar Favorito

4.2. OSM

La aplicación sobre OSM tiene como estructura general la mostrada en la figura 4.7.



Fig. 4.7 Estructura programa LazAndroid con OSM.

Lo primero que se debe realizar es comprobar si el GPS está activado, con el menú de la Fig. 4.8 Si la respuesta es afirmativa directamente entrará en la vista del mapa centrándose en la posición actual. En el caso de que no se encuentre activa se mostrará un menú en el que se pregunta si se quiere activar o no.



Fig. 4.8 Pantalla de inicio.

En el caso de que se elija Cancelar se saldrá directamente de la aplicación.

Se estudió la manera de activar directamente el GPS sin necesidad de ir al menú del propio móvil, para facilitar esta tarea al usuario, pero esto sólo era posible en las primeras versiones de Android. Ahora, por motivos de seguridad, no hay manera de acceder a estas opciones directamente sin pasar por el menú que se ve en la Figura 4.9.



Fig. 4.9 Menú activación GPS

Si se vuelve a la pantalla anterior sin activar el GPS se volverá al menú de la imagen 4.8. Se volverá a preguntar si se desea activar, hasta que se active el GPS o se pulse cancelar en la ventana.



The screenshot shows an Android application interface with a dark background. At the top, there is a status bar with icons for signal, Wi-Fi, battery, and the time 13:18. Below the status bar, there is a form with several input fields and labels. The fields are arranged vertically, and the labels are to their right. The fields contain the following text: 'España', 'Barcelona', 'Barcelona', 'Avenida', 'Diagonal', '32', and an empty field for 'CP'. The '32' field is highlighted with a green border. At the bottom right of the form, there is a button labeled 'Buscar'.

País	Provincia	Localidad	Tipo de vía	Nombre	Número	CP
España	Barcelona	Barcelona	Avenida	Diagonal	32	

Fig. 4.11 Entrada datos de destino.

En la figura 4.11 se ve la actividad en la que se introducen los datos del destino. Una vez tenemos la dirección de destino hacemos una llamada a Nominatim para que nos devuelva todas las posibles opciones de la búsqueda.

En la Fig. 4.12 se muestra un menú con las posibles opciones de búsqueda que nos devuelve la consulta realizada.



Fig. 4.12 Lista de resultados de búsqueda

Llegados a este punto, según el esquema del programa que se vio al comienzo del apartado, ahora se debería invocar a un servlet dándole como parámetros de entrada una dirección origen (latitud y longitud de la posición actual) y una dirección del destino que se elija de las opciones que se nos muestran en la lista (Figura 4.12).

Una vez en el servlet se haría una petición a la XAPI indicándole el bbox que se quiere mostrar, y ésta debería devolver un mapa con sus puntos de interés, vías, coordenadas y demás.

Sin embargo, a lo largo de la realización de este proyecto, se han encontrado bastantes dificultades a la hora de conectar con el servidor de XAPI, ya que la gran mayoría están caídos o no funcionan correctamente.

Hace poco se ha incluido en la página de XAPI una nueva opción de búsqueda, de la que ya se ha hablado en capítulos anteriores: jxapi.

Esta versión de XAPI si funciona bien, pero también tiene un pequeño inconveniente: tarda demasiado en devolver una respuesta, haciendo difícil la búsqueda en tiempo real que se necesita para este proyecto.

Es por este motivo que se ha decidido dejar el proyecto en este punto, y hacer una pequeña demo de cómo funcionaría el encaminamiento, pero a partir de un fichero de XAPI descargado en local.

CAPITULO 5. DEMO DE ENCAMINAMIENTO

5.1. Encaminamiento

Como se ha comentado en el apartado anterior, para realizar esta demo se parte de un archivo de XAPI presente en disco. Dicho archivo se puede obtener haciendo la siguiente llamada sobre la XAPI:

http://xapi.openstreetmap.org/xapi/api/0.6/*%5Bbbox=1.9803,41.2731,1.9918,41.2767%5D

Lo primero que se tiene que hacer es instalar la librería jgraph, ya que se necesitará para realizar el encaminamiento.

Una vez instalada la librería y con el trozo de mapa, se elige un punto de origen y uno de destino, que tienen que estar incluidos dentro de este mismo mapa, ya que en principio en la llamada anterior hace referencia a un BoundingBox que contiene los puntos Origen y Destino de la ruta.

En primer lugar se recorre el mapa, extrayendo todas las vías de la zona. A continuación se construye el grafo, de dos en dos, calculando los pesos.

Una vez se tiene el grafo y los nodos origen y destino, se llama a la función DijkstraShortestPath, la cual nos devolverá el path con los puntos de la ruta más corta, los de menor peso.

En la figura 5.1 se muestra el resultado de la traza para dos puntos concretos del mapa:

```
30299095 (0.0, 0.0) to 60200333 (0.0, 0.0)
(30299095 (41.3619158, 2.1255063) : 30315289 (41.3620763, 2.1257815))
(30315289 (41.3620763, 2.1257815) : 30315304 (41.3622144, 2.1261077))
(30315304 (41.3622144, 2.1261077) : 30299049 (41.3623404, 2.1263517))
(30299049 (41.3623404, 2.1263517) : 30308140 (41.3624741, 2.1267319))
(30308140 (41.3624741, 2.1267319) : 30315193 (41.3628279, 2.1273266))
(30315193 (41.3628279, 2.1273266) : 30315196 (41.3632154, 2.127847))
(30315196 (41.3632154, 2.127847) : 30298985 (41.3633117, 2.1279831))
(30298985 (41.3633117, 2.1279831) : 30299022 (41.3634988, 2.1283033))
(30299022 (41.3634988, 2.1283033) : 257712797 (41.3636356, 2.1285293))
(257712797 (41.3636356, 2.1285293) : 60200014 (41.3646396, 2.1288371))
(60200014 (41.3646396, 2.1288371) : 60200015 (41.3656199, 2.128368))
(60200015 (41.3656199, 2.128368) : 30298966 (41.3662919, 2.1298485))
(30298966 (41.3662919, 2.1298485) : 60200333 (41.366529, 2.1303427))
Distance = 742.0341121961325 m
Time = 11.130511682941988 min
```

Fig. 5.1 Camino Dijkstra

5.2. Instalación Servidor OSM

Se ha estudiado una posible solución al problema de los servidores de XAPI, que consiste en instalar un Servidor OSM.

A continuación se indica los pasos a seguir para la instalación:

- Conseguir el último archivo “planet” – OpenStreetMap
- Instalar osm2pgsql
- Instalar PostGIS 8.4
- Instalar Mapnik
- Instalar Mapnik tolos
- Instalar Osmosis
- Importar Planet en PostGIS

Una vez instalado ya se puede utilizar el servidor OSM. En las referencias se puede encontrar los enlaces a los tutoriales donde se encuentra explicado más detalladamente la instalación.

Sería interesante estudiar esta posibilidad de mejora en algún proyecto futuro con el fin de poder terminar de implementar el encaminamiento.

CAPITULO 6. CONCLUSIONES

6.1. Resultados

Se ha desarrollado dos aplicaciones en Android enfocadas a la movilidad para invidentes.

Se ha podido realizar una aplicación que tiene cierta ayuda a la movilidad para invidentes, ya que el manejo de la aplicación está enfocado a ello.

Sin embargo, debido a que OSM no es tan potente como GoogleMaps se ha dificultado bastante el desarrollo de la aplicación.

6.2. Mejoras futuras

Debido a la implementación primero en GoogleMaps y después con OSM, con el esfuerzo extra que éste último ha supuesto, no ha sido posible realizar muchas de las implementaciones que se pensaron en un principio. Un proyecto como este presenta muchísimas posibilidades y mejoras, ya que lamentablemente no está muy desarrollado el mundo de las aplicaciones de ayuda a los discapacitados. Algunas de las posibles mejoras de este proyecto son:

6.2.1. Vibración

Sería interesante ampliar la aplicación añadiendo vibración al pasar por encima de las opciones, por ejemplo al entrar los datos del destino. Así sería mucho más fácil el manejo de la aplicación.

6.2.2. Abecedario entrada datos

Otra posible mejora sería la posibilidad de poder recorrer el abecedario, a la vez que éste se reproduce con el TextToSpeech, para introducir datos letra a letra en aquellos casos en los que debido al ruido o cualquier otro motivo no sea posible el reconocimiento de la voz o no devuelva el resultado deseado.

6.2.3. Sonidos

El uso de un simple sonido cuando se detecta una opción o cuando se ha introducido un dato ayudaría para saber si se están introduciendo correctamente los datos.

6.2.4. Opciones predeterminadas

Se podría implementar la posibilidad de guardar unas opciones predeterminadas de búsqueda. Por ejemplo, el País, Provincia o Localidad en la que se suele hacer la búsqueda, con la posibilidad de cambiarla si es necesario.

6.2.5. Favoritos

Se puede desarrollar, tal y como se hizo en el primer proyecto con Google Maps, la posibilidad de insertar Favoritos, con el fin de hacer más fácil y rápida la búsqueda de destinos frecuentes.

6.2.6. Brújula

La implementación de una brújula que indique hacia dónde debe orientarse el usuario

6.2.7. Puntos de interés

La detección de puntos de interés como paseos de peatones, semáforos adaptados, entradas a edificios, paradas de autobús / metro. Para ello hace falta un mapeo exhaustivo para OSM.

6.2.8. Ruta más corta contemplando servicio público

Indicar no sólo la ruta a pie, sino contemplar el uso de transporte público en función de los horarios de autobuses, las paradas de metro o autobús cercanas, tanto al punto de origen como al destino, y posibilidad de transbordo si es necesario.

6.2.9. Bases de datos

Implementar una base de datos que haga posible el punto anterior, en el que se guarden las distintas paradas, rutas, líneas de metro, etc.

6.2.10. Accesibilidad en edificios

Para hacer más fácil los transbordos necesarios sería útil poder guiar a los invidentes incluso dentro del metro.

6.2.11. Alerta de incidencias.

Debido al sencillo modo de acceder a los mapas de OSM, sería interesante poder incluir alertas de incidencias, por ejemplo aviso de obras o calles cortadas por las que no se podrá acceder por un periodo de tiempo.

6.2.12. Indicaciones de movilidad por voz

En el caso de GoogleMaps era bastante sencillo implementar esta opción, ya que la propia llamada a GoogleEarth o GoogleMaps devuelve las indicaciones de la ruta. Sin embargo, en OSM el encaminamiento se realiza en la propia máquina, haciendo uso de Dijkstra, por lo que habría que implementar las instrucciones de movilidad en la ruta para poder indicarlo por voz al usuario.

6.3. Impacto medioambiental

El proyecto se basa en una aplicación programada para el entorno Android, por lo que no ha tenido ningún impacto medioambiental.

CAPITULO 7. REFERENCIAS

- [1] Web oficial programadores Android: <http://developer.android.com>
- [2] Foro especializado en Android: <http://and.roid.es/foro>
- [3] Foro especializado en Android: <http://android-spa.com>
- [4] Manual para la instalación del SDK de Android:
<http://developer.android.com/sdk/installing.html>
- [5] Manual de introducción a la programación Android:
<http://www.cristalab.com/tutoriales/crear-un-hola-mundo-en-android-c76946/>
- [6] Usando GoogleMaps en Android:
<http://mobiforge.com/developing/story/using-google-maps-android>
- [7] Google Geocoding API:
<http://code.google.com/intl/es/apis/maps/documentation/geocoding/>
- [8] Manual cartografiado OSM:
http://wiki.openstreetmap.org/wiki/GvSIG_Valencia_mapping_party_Tutorial
- [9] Nominatim: <http://wiki.openstreetmap.org/wiki/Nominatim>
- [10] XAPI: <http://wiki.openstreetmap.org/wiki/Xapi>
- [11] Tutorial instalación "planet": <http://weait.com/content/build-your-own-openstreetmap-server>
- [12] Mirror y archivos de "planet": <http://planet.openstreetmap.org/>
- [13] Instalación Mapnik: <http://trac.mapnik.org/wiki/MapnikInstallation>
- [14] Mapnik: <http://wiki.openstreetmap.org/wiki/Mapnik>