Universidad Carlos III de Madrid
Escuela Politécnica Superior

Ingeniería Técnica en Informática de Gestión

Proyecto Fin de Carrera

Buddy UC3M – Sistema de Ubicación Universitario

AUTOR: Pablo Torrejón Sánchez

TUTOR: Prof. Jesús García Herrero

Colmenarejo, septiembre 2015

Agradecimientos:

Después de este largo camino, marcado por las circunstancias de la vida, puedo decir que el ser ingeniero es más que estudiar, comprender o dominar conceptos complejos. Para mí ser ingeniero es una actitud, un comportamiento aprendido que nos habilita y predispone para enfrentarnos a los problemas que nos van surgiendo en nuestro día a día.

Durante estos años se nos ha enseñado a dar lo mejor de nosotros mismos siendo precisos con nuestra capacidad resolutiva empleando un arsenal de conocimientos técnicos.

Mi andadura en la universidad comenzó como la de cualquier otro chaval que le gustan los ordenadores. Indeciso de si dedicarme a la automoción, la cual es otra de mis grandes pasiones, o a la informática, que era mi dedicación y experimentación en aquellos tiempos. Finalmente opté por la última.

¡Porqué negarlo! Ha sido un camino duro con ciertos desengaños. Muchas horas sin dormir, estrés, nervios, euforia, felicidad, satisfacción, fracasos... etc. Pero creo que si hago un cómputo global de todo puedo decir con total seguridad que ha merecido la pena.

He conocido a **grandes profesores, personas** y **grandes compañeros** en esta andadura. Lástima que por circunstancias de la vida hayamos perdido el contacto o éste apenas exista.

Por esos momentos de prácticas **Germán** Ramírez Rodríguez, apreciado compañero.

La mención más importante y especial es para mi familia. A mi mujer, **María José,** y a mi hijo, **Héctor,** por infundirme el valor y arrojo necesario para seguir adelante cuando no veía más que un muro y estar en el punto de querer abandonar. Sois lo mejor que tengo.

A mis **padres** y a mi **hermano** por el apoyo incondicional que me brindaron durante los primeros años de universidad. Por soportar mis neuras y a veces mi falta de tacto cuando se acercaban los periodos de exámenes, era mi "yo estresado" quién se manifestaba.

Finalmente, no quisiera olvidar mencionar a mi mejor amigo y experto informático **Martín** Nardella Malvicini, el que SIEMPRE ha estado incondicionalmente a mí y me ha ayudado en los momentos donde no había claridad.

Por todos y cada uno de ellos, con todo mi afecto,

¡Muchas gracias!

Pablo Torrejón Sánchez.

Resumen:

Para el Proyecto de Fin de Carrera he decidido por desarrollar una aplicación de geoposicionamiento orientada al Campus de Colmenarejo. La aplicación, de ahora en adelante app, está implementaba sobre el sistema operativo móvil Android.

La app se podría definir como un sistema de ubicación universitario. La he llamado Buddy UC3M, ya que buddy significa en inglés: amigo, hermano, compañero, voluntario, etc. Y al fin y al cabo es la utilidad que he tratado de implementar al desarrollar este proyecto.

La finalidad de esta app es mostrar al usuario la ruta embebida en la propia app desde su posición actual hasta unos puntos clave predefinidos dentro del Campus de Colmenarejo. Estos son la Biblioteca Menéndez Pidal, las aulas del Edificio Miguel de Unamuno, la residencia de estudiantes Antonio Machado y el acceso a la secretaría del centro.

Una vez que el usuario selecciona a dónde quiere ir, la app muestra en un mapa la ruta hasta el punto destino y le permite navegar mediante indicaciones multimedia que son proporcionadas por la navegación de Google Maps.

Asimismo la app incluye características adicionales como el plano de la universidad, en este caso del Campus de Colmenarejo, que permite su interacción con el mismo pudiendo hacer zoom, inclinarlo e incluso ver el mapa interno de los edificios más representativos del Campus por plantas. También el usuario puede obtener su ubicación actual y hacer uso de Google Street View, a lo que yo he denominado modo inmersivo.

Para la implementación de esta aplicación se ha utilizado el entorno de desarrollo Android Studio y se ha trabajado con la información proporcionada por el hardware del móvil así como con diferentes APIs de Google que están a disposición de cualquier desarrollador.

Palabras claves: APP, aplicación, dispositivo, Android, Smartphone, GPS, WiFi, sensores, interfaz, Java, XML, API, Google Maps, Google API, Google Directions, Google Street View, Android Studio, layout, manifest, ruta, navegación, posición, longitud, latitud, pantalla, panorama, mapa.

Abstract:

For Final Career Project I decided to develop a geopositioning application oriented to Colmenarejo Campus. The application, henceforth app is was implementing on Android mobile operating system.

The app can be defined as a system of university location. I called Buddy UC3M as buddy means in English: friend, brother, partner, volunteer, etc. And after all is the utility that I've tried to implement developing this project.

The app purpose is to show the user the path, embedded in the app, from its current position to a predefined key points within the Colmenarejo Campus. These are the Library Menéndez Pidal, building classrooms Miguel de Unamuno, the student residence Antonio Machado and access to the Campus secretariat.

Once the user selects where to go, the app shows on a map the route to the destination point and allows you to navigate through multimedia instructions that are provided by Google Maps navigation.

The app also includes additional features such as the map of the university, in this case Colmenarejo Campus, which allows interaction with it being able to zoom, tilt and even see the internal map for plants of the most representative Campus buildings. The user can also get your current location and make use of Google Street View, what I call immersive mode.

To implement this application has used the Android Studio development environment and has worked with the information provided by the mobile hardware as well as various Google APIs that are available to any developer.

Keywords: APP, application, device, Android, Smartphone, GPS, WiFi, sensors, interface, Java, XML, API, Google Maps, Google API, Google Directions, Google Street View, Android Studio, layout, manifest, route navigation position, longitude, latitude, screen, panorama, map.

Índice general:

1) Introducción

La evolución de lo que es a día de hoy la informática ha sido sorprendente. De ser algo exclusivamente minoritario de las grandes empresas y corporaciones a integrarse en nuestras vidas siendo presente y necesaria en cada faceta de nuestro día a día.

Con la red de redes, Internet, ocurrió algo parecido. En sus inicios algo que fue utilizado por las universidades y las instituciones militares para el acceso inmediato a la información y la compartición de datos, con el tiempo, se abrió paso hacia los ciudadanos de a pie.

Recuerdo los tiempos en los que conectarte a la red tenía su encanto, el sonido de ese módem V90 marcando y estableciendo la conexión, la lenta velocidad de carga que hacía que navegar por web gráficas fuera un infierno y donde el conseguir descargarse algo era todo un triunfo.

En los últimos años estos paradigmas han cambiado sustancialmente. El hardware ha sufrido una miniaturización y se ha vuelto portable. Quién se iba a imaginar hace menos de 10 años que los teléfonos móviles iban a ser más funcionales y usables, sino que podríamos prácticamente llevar una computadora con las funcionalidades de un teléfono en el bolsillo, además de estar dotada de su sistema operativo distribuido, sus aplicaciones, sus sensores, su almacenamiento, etc.

De esta evolución aparece en nuestras vidas el término de Smartphone o teléfono inteligente, que podría definirse como un teléfono celular con pantalla táctil y la capacidad de conectarse a internet además de poder instalar aplicaciones a modo de un pequeño computador.

Respecto al acceso a Internet podemos decir que básicamente se ha pasado de una conectividad fundamentalmente a través de un medio guiado a que se vaya tendiendo al uso de datos móviles que permiten al usuario en cuestión el acceso a Internet en cualquier ubicación donde tenga cobertura.

Nos enfrentamos a un futuro dominado por la movilidad y el Cloud Computing donde el acceso a la información será continuado y ubicua.

1.1. Motivaciones

Siempre ha llamado mi atención el funcionamiento del sistema GPS así como el cálculo de rutas, teoría de grafos, etc. Durante mis años académicos siempre eché en falta saber dónde estaba un aula en particular cuando tenía que hacer un examen o asistir a una nueva clase.

Asimismo el tener un plano del Campus, saber tu posición dentro del mismo, etc. Lo considero algo de gran valor añadido para cualquier nuevo estudiante.

Debido al marco tecnológico actual dónde la gran mayoría dispone de un teléfono inteligente con acceso a internet, pensé que desarrollar una aplicación para Android que fuera como un asistente de posicionamiento dentro del Campus sería una buena idea.

Si sumamos a todo lo anterior mi interés en aprender a desarrollar aplicaciones móviles y que Google pone a disposición de cualquier persona las herramientas necesarias para llevarlo a cabo, hace un gran cóctel bastante difícil de rechazar.

1.2. Objetivos

El objetivo principal de este PFC es la creación de una aplicación móvil que permita el cálculo de rutas desde la ubicación actual del usuario hacia unos puntos predefinidos dentro del Campus de Colmenarejo.

Como uno de los objetivos siempre ha sido obtener el mayor alcance entre los usuarios de telefonía móvil, especialmente gente joven, se ha optado por realizar el desarrollo de la app en el sistema operativo Android ya que es en el que en mayor número de terminales se encuentra instalado.

En cuanto a la elección de cuáles serían los puntos de interés con los que el usuario podría interactuar se ha decidido por experiencia que los POI's a los que un alumno o nuevo visitante del Campus recurre son: las aulas para encontrarlas en sus primeros días o exámenes, el colegio mayor Antonio Machado para posibles estancias, la Biblioteca Menéndez Pidal ya sea para préstamo de libros, actos o estudio y por último la secretaria del centro donde se realizan la mayoría de las gestiones administrativas importantes.

Durante el desarrollo de la aplicación se estableció como condición sine qua non que todas las funcionalidades implementadas se encontraran integradas dentro de ella. Para ello

se ha realizado el cálculo de las rutas sobre mapas Google embebidos en la app, de este modo el usuario obtiene una buena experiencia de uso y simplifica enormemente el manejo de la misma.

Asimismo se ha considerado que sería interesante la integración de nuevas funcionalidades en la app que generan valor añadido. Es por este motivo por lo que entre otras se ha implementado la consulta e interacción sobre el mapa del Campus de Colmenarejo. Este mapa se presenta al usuario con una orientación lógica, es decir, no se orienta directamente al norte sino que se ha girado para simular una visión aérea desde la puerta principal de entrada al Campus de Colmenarejo. Para mejorar la experiencia en la interacción también se ha habilitado en la interfaz del mapa los edificios más representativos y se pueden visualizar desglosados por plantas.

Además se ha incorporado la posibilidad de visualizar la ubicación del usuario sobre un mapa de satélite e interaccionar del mismo modo como se puede interactuar con la opción de mostrar el mapa del Campus.

Otro de los objetivos a lograr ha sido el conseguir una sensación de realismo de cara al usuario y respecto a su ubicación actual. Para ello se ha implementado que tanto en el cálculo de rutas como en la visualización de la ubicación actual del usuario en vista de satélite, se tenga la opción de mostrar imágenes reales de los alrededores en 360 grados. Para esta función se ha integrado en la app Google Street View.

Por último, otros de los objetivos ha sido el conseguir la navegación guiada por voz e indicaciones desde nuestra ubicación actual hasta unos de los cuatro puntos predefinidos. En este caso no ha sido posible integrarlo dentro de la app ya que Google no lo permite. Por tanto como solución se ha optado por externalizar está funcionalidad a Google Maps, que tendrá que estar instalada en nuestro terminal para poder explotar esta característica.

Finalmente podría decirse que conceptualmente el objetivo de Buddy UC3M es un conglomerado de funcionalidades de ayuda o asistencia a los nuevos estudiantes o a las personas que nunca hayan visitado el Campus de Colmenarejo.

Cualquiera que quiera llegar desde su ubicación a un determinado punto de interés de los predefinidos, poseer en cualquier lugar un plano del Campus en un modo interactivo y detallado (con los edificios más representativos con sus planos internos), conocer su ubicación actual en un plano satélite con posibilidad de imágenes reales en 360 grados o bien navegación guiada hacia cualquiera de los cuatro puntos de interés, tendrá en Buddy UC3M la solución.

1.3. Requisitos del dispositivo

Para poder instalar Buddy UC3M en el dispositivo y garantizar un correcto funcionamiento, este tiene que tener las siguientes características:

- Sistema Operativo Android, se puede decir que Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tablets; aunque también está siendo usado para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró. El principal motivo por el que me he decantado por el uso de este S.O es que se puede decir que los dispositivos con Android venden más que las ventas combinadas de Windows Phone e IOS, por lo que podrá ser usada por un mayor número de usuarios.

- Conectividad a Internet Móvil, mi APP trabaja con varios servicios Google que se encuentran alojados en sus servidores. Es por ello por lo que la app necesita conexión a internet.

- Sensor de geoposicionamiento, mi app utiliza el posicionamiento GPS para obtener la ubicación del usuario y así poder tener un punto de partida desde el que calcular la ruta hacia el destino. La mayoría de los Smartphones modernos ya traen GPS.

1.4. Fases de desarrollo

Durante el tiempo que transcurrió la realización de este PFC, podemos decir que éste se dividió en varias etapas o fases bien diferenciadas. A continuación se procede a explicar cada etapa ordenada temporalmente de comienzo a fin:

- ### Documentación previa:

  Durante esta etapa del PFC tuve que investigar las posibilidades existentes para el desarrollo de una APP Android, así como estudiar la estructura de una aplicación Android, refrescar mis conocimientos de Java, etc.

- ### Fase de Análisis:

Nombre de tarea	Duración	Comienzo	Fin
FASE DE ANÁLISIS	**11 días**	**mié 01/04/15**	**mié 15/04/15**
Análisis del entorno físico	6 días	mié 01/04/15	mié 08/04/15
Análisis de requerimientos software	5 días	jue 09/04/15	mié 15/04/15
FIN DE LA FASE DE ANÁLISIS	0 días	mié 15/04/15	mié 15/04/15

  Esta fase se analizó cuáles eran los objetivos de nuestra app y la manera correcta de abordar el problema. Se estudió el entorno geográfico del Campus de Colmenarejo, como conseguir el cálculo de rutas dentro de los caminos del Campus, se obtuvieron la longitud y latitud de los puntos de interés para los cuales se ha configurado la app, etc.
  Asimismo se hizo un análisis de los requerimientos software tanto de la app a desarrollar como del equipo informático donde se iba a ejecutar el entorno de desarrollo.

- ### Fase de Diseño:

Nombre de tarea	Duración	Comienzo	Fin
FASE DE DISEÑO	**18 días**	**mié 15/04/15**	**vie 08/05/15**
Configuración entorno de desarrollo Android Studio	3 días	jue 16/04/15	lun 20/04/15
Determinación tipo de APP Android	2 días	mar 21/04/15	mié 22/04/15
Diseño de navegación en la APP	2 días	jue 23/04/15	vie 24/04/15
Obtención Google API Key	1 día	lun 27/04/15	lun 27/04/15
Ámbito y limitaciones Google Maps API	5 días	mar 28/04/15	lun 04/05/15
Ámbito y limitaciones Google Directions API	3 días	mar 05/05/15	jue 07/05/15
FIN FASE DE DISEÑO	0 días	jue 07/05/15	jue 07/05/15

En esta fase del PFC se procedió a configurar correctamente el entorno de desarrollo Android Studio con todas sus dependencias y librerías, se determinó que tipo de app Android se iba a desarrollar, una vez decidido el diseño se trabajó con el tipo de interfaz (colores, botones, imágenes y diversos elementos), se esquematizó la navegación de la app así como el comportamiento ante diferentes eventos (giro de la pantalla, botón atrás, GPS deshabilitado, etc.), se obtuvo la clave Google API Key que posibilita el desarrollo de app Android con acceso a servicios Google y finalmente se estudió en detalle las limitaciones existentes en las diversas APIs empleadas como la de Google Maps y Google Directions.

- ## Fase de Implementación:

Nombre de tarea	Duración	Comienzo	Fin
FASE DE IMPLEMENTACIÓN	**42 días**	**vie 08/05/15**	**lun 06/07/15**
Implementación archivo Manifest	3 días	vie 08/05/15	mar 12/05/15
Implementación Clases JAVA	30 días	mié 13/05/15	mar 23/06/15
Implementación Layouts	30 días	mié 13/05/15	mar 23/06/15
Creación elementos gráficos interfaz	7 días	lun 15/06/15	mar 23/06/15
Depuración código	7 días	mié 24/06/15	jue 02/07/15
Generación APK	2 días	vie 03/07/15	lun 06/07/15

Durante esta fase o etapa se procedió a la codificación de cada una de las partes que contiene una app Android. Se implementó el archivo manifest, cada una de las clases Java así como los XML que sirven para la creación de cada layout y por tanto la definición de los elementos dentro de cada vista.
Asimismo se crearon los elementos de la interfaz gráfica tales, como botones, logos, splash screen, etc.
Por último de depuró el código y se generó el correspondiente fichero empaquetado .APK para instalar la app en cualquier dispositivo Android.

- ## Fase de Documentación:

Nombre de tarea	Duración	Comienzo	Fin
FASE DE DOCUMENTACIÓN		**mié 01/04/15**	
Documentación en Microsoft Project	61 días	**jue 16/04/15**	jue 09/07/15
Elaboración Memoria PFC	61 días	jue 16/04/15	jue 09/07/15
Revisión Documentación	7 días	vie 14/08/15	lun 24/08/15

La fase de documentación es una etapa que comenzó desde el mismo momento en que se empezó con la fase de diseño. Durante este PFC las tareas fueron documentadas en MS Project y se fue documentando a medida que avanzaba el desarrollo paso a paso.

Finalmente y una vez que se acabó de confeccionar la documentación de la app, se procedió a su revisión y mejora si procedía.

- ## Fase de Pruebas:

Nombre de tarea	Duración	Comienzo	Fin
FASE DE PRUEBAS	**59 días**	**mié 15/04/15**	lun 06/07/15
Pruebas sensores geoposicionamiento	33 días	**mié 15/04/15**	vie 29/05/15
Pruebas comportamiento interfaz APP	12 días	lun 01/06/15	mar 16/06/15
Pruebas alcance requisitos funcionales	14 días	mié 17/06/15	lun 06/07/15

Esta fase ha sido recurrente a medida que se ha ido desarrollando la app. Se ha probado el funcionamiento de los sensores en aspectos tales como su disponibilidad, refresco y precisión.
Como es lógico se ha probado la navegación entre pantallas, comportamiento de cada uno de los elementos de la app y como no podía ser de otra manera si el PFC planteado alcanzaba los requisitos funcionales propuestos.

- ## Fase de Entrega:

Nombre de tarea	Duración	Comienzo	Fin
FASE DE ENTREGA	**1 día**	**mar 01/09/15**	**mar 01/09/15**
Defensa aproximada PFC (Primeros de Septiembre)	1 día	mar 01/09/15	mar 01/09/15

Esta es la última fase del PFC, una vez terminado se procederá a defender en una fecha marcada por la Universidad la se estima que será a primeros de Septiembre. Durante la defensa del PFC ante el tribunal se explicará de la manera más breve, concisa y precisa la app Buddy UC3M.

1.5. Estructura de la memoria

El documento está estructurado de la siguiente manera:

1. **Introducción:** En esta sección se hace un repaso por la evolución y coyuntura actual de las tecnologías que nos aplican, así como una descripción de las motivaciones, objetivos, requisitos y fases de desarrollo para este PFC.

2. **Estado del Arte:** En este capítulo se procederá a detallar el entorno y tecnologías empleadas para este PFC, así como la base en la que se sustenta el valor añadido que genera la app Buddy UC3M.

3. **Análisis:** A lo largo de esta sección se detalla el trabajo realizado en el análisis de la app para la formulación de los requisitos.

4. **Diseño:** En este capítulo se detallará cada uno de los pasos que hubo que realizar para parametrizar este tipo de app dentro del entorno de desarrollo Android Studio. También se hace mención de las particularidades y restricciones de las API's de Google utilizadas.

5. **Implementación:** Aquí se detalla la codificación de cada uno de los componentes implementados para la app. También se analizan los principales algoritmos y soluciones implementadas para alcanzar las funcionalidades buscadas.

6. **Pruebas:** Esta fase ha sido recurrente a medida que se ha ido desarrollando la app. Se ha probado el funcionamiento de los sensores en aspectos tales como su disponibilidad, refresco y precisión.
   Como es lógico se ha probado la navegación entre pantallas, comportamiento de cada uno de los elementos de la app y como no podía ser de otra manera si el PFC planteado alcanzaba los requisitos funcionales propuestos.

7. **Presupuesto:** Se detallan los costes para la implementación y uso de Buddy UC3M.

8. **Documentación:** En este capítulo de habla sobre las herramientas utilizadas en el proceso de documentación.

9. **Bibliografía:** Información empleada en la realización de este PFC.

10. **Conclusión y futuribles:** Conclusión de los conocimientos y experiencia aprendida durante este PFC, así como líneas futuras de trabajo o mejoras en la app.

2) Estado del Arte

Después del capítulo de introducción se continúa con el titulado Estado del Arte. Este capítulo se caracteriza por la explicación de las tecnologías empleadas durante este PFC. Asimismo se hará hincapié en la parte más compleja de la app que es el dibujo y cálculo de rutas sobre un mapa.

2.1. Android

Para la realización de este Proyecto de Fin de Carrera se ha implementado una aplicación para el Sistema Operativo Android.

Android es un Sistema Operativo basado en el núcleo Linux. Inicialmente fue desarrollado por Android Inc., empresa que tras la financiación de Google esta compró en 2005.

Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tablets. La plataforma de hardware principal de Android es la arquitectura ARM y existe soporte para x86.

Android, al contrario que otros sistemas operativos para dispositivos móviles como iOS o Windows Phone, se desarrolla de forma abierta y se puede acceder sin problemas al código fuente. Las aplicaciones se desarrollan habitualmente en el lenguaje Java con Android Studio (Android SDK), pero están disponibles otras herramientas de desarrollo como un entorno visual para programadores poco experimentados.

Se puede afirmar que el desarrollo de aplicaciones para Android no requiere aprender lenguajes complejos de programación. Con tener unos buenos conocimientos de Java y estar en posesión del entorno de desarrollo provisto por Google, el cual se puede descargar gratuitamente, sería suficiente.

Todas las aplicaciones están comprimidas en formato APK y se pueden instalar sin problemas en cualquier dispositivo Android, que cumpla los requisitos mínimos de la misma, siempre y cuando esta no esté firmada digitalmente o sea posea la firma correspondiente.

Por estos motivos, Android tiene una gran comunidad de desarrolladores creando aplicaciones para extender la funcionalidad de los dispositivos. A fecha de hoy, se ha llegado ya al 1.000.000 de aplicaciones disponibles para la tienda de aplicaciones oficial de Android: Google Play, sin tener en cuenta aplicaciones de otras tiendas no oficiales para Android.

Es importante destacar que una de las motivaciones por las que se eligió trabajar con este S.O. es que los dispositivos de Android venden más que las ventas combinadas de Windows Phone e IOS. En la actualidad existen aproximadamente 1.000.000 de aplicaciones para Android y se estima que 1.500.000 teléfonos móviles se activan diariamente. Asimismo también es importante hacer hincapié que el sistema operativo Android es accesible ya que se usa tanto en teléfonos inteligentes, ordenadores portátiles, tabletas, etc. Algunos con precios menores de 100 €.

2.1.1. Elementos básicos de una app Android

Una aplicación Android se compone de los siguientes elementos básicos que se pueden apreciar cuando nos encontramos dentro del SDK. Bibliografía

- **Fichero Manifest:**

  El fichero Manifest es un fichero XML donde se definen una serie de características relativas a la configuración y comportamiento de la aplicación.

  En el fichero se pueden definir los siguientes aspectos: versión de nuestra app, la versión mínima del S.O que podrá utilizar nuestra app así como la para la que ha sido desarrollada, permisos a los que el usuario tendrá que autorizarnos para que la app funcione debido a ser requisitos funcionales de la misma, las API KEY's necesarias para su correcto funcionamiento, en nuestro caso la de Google Map; nombre de la app, icono de la app o por ejemplo las actividades que forman nuestra app y el comportamiento de cada una de ellas.

  Asimismo el fichero Manifest siempre debe de llamarse con la nomenclatura AndroidManifest.xml

- **Actividades:**

  Una aplicación Android se compone de una o varias actividades que se ubican dentro del directorio app/java/nombre_paquete. En nuestro caso app/java/es.uc3m.buddyuc3m

  Podría decirse que cada actividad es una clase Java donde se encuentra el código de una determinada tarea. Dentro de cada actividad puede hacerse referencia a otras clases así como a la interfaz de la aplicación.

  En nuestro caso, se ha tratado de reutilizar código empleando una misma actividad para diferentes casuísticas dentro de la app y también se ha dividido en clases determinadas acciones dentro de Buddy UC3M.

- **Recursos:**

  Dentro de los recursos que puede tener una aplicación Android se van a destacar los siguientes:

  - Drawable: En esta parte de la estructura de la app es donde se guardan los elementos gráficos a los que la aplicación puede recurrir. Se pueden considerar como tal logos, botones, fondos, pantallas de bienvenida, etc.

  - Layout: Es la parte más importante de la app en cuanto a la representación al usuario de la información. Esta sección contiene ficheros XML asociados a una actividad en particular, donde se precisa el comportamiento, apariencia y estilo de la interfaz gráfica de la app para esa actividad.

  - Values: Aquí se almacenan en ficheros XML los diferentes valores de variables y constantes para que si se quieren puedan ser referenciadas sin problemas a lo largo de todo el código fuente de la app.

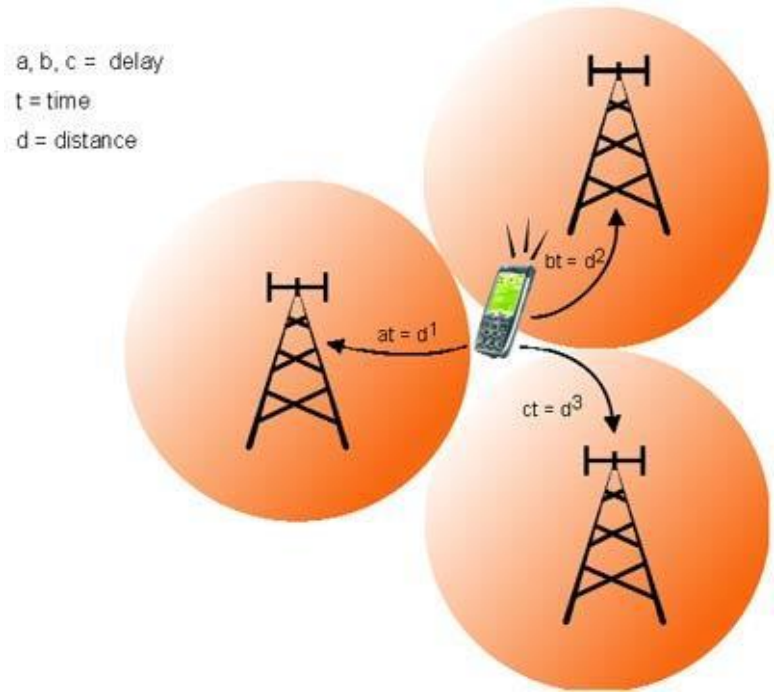
2.2. Localización GSM

Hablaremos de los fundamentos de la localización GSM, ya que en nuestra app hasta que se inicializa el GPS y recibimos unas coordenadas válidas, se utiliza esta tecnología para obtener una ubicación aproximada.

La localización GSM es un servicio ofrecido por las empresas operadoras de telefonía móvil que permite determinar, con una cierta precisión, donde se encuentra físicamente un terminal móvil determinado.

En principio la tecnología más empleada para la localización GSM es la basada por la información aportada por las antenas de la red de telefonía. Nuestro terminal cuando se encuentra encendido se encuentra conectado a todas las antenas del proveedor que se encuentren a su alcance y por tanto esta tecnología requiere que se conozca cada una de las antenas de la red.

Aproximadamente dos antenas con una buena cobertura se encuentran separadas unos 5 kilómetros sino menos, se podría decir que en un radio de 10 kilómetros podríamos tener más de 10 antenas que nos proveen de cobertura. La clave de este concepto es el nivel de señal que posee el teléfono con cada una de las antenas a las que está conectado. Esto permite determinar aproximadamente a que distancia se encuentra de cada una de ellas, haciendo posible la triangulación de la ubicación aproximada del terminal. Bibliografía



Existen más datos como TA (Time Advance) para ubicar al terminal, pero estos datos no son muy exactos.

Finalmente podemos decir que aunque hay una variación de unos metros con respecto a la ubicación real, la gran ventaja es que no se requiere GPS por lo que ha sido útil para obtener la posición inicial de nuestro terminal hasta que los servicios GPS estaban accesibles.

2.3. Localización GPS

Para el funcionamiento de Buddy UC3M es imprescindible el uso del sistema de posicionamiento global o GPS. Es un sistema que permite determinar en todo el mundo la posición de un objeto (una persona, un vehículo) con una precisión de hasta centímetros, aunque lo habitual son unos pocos metros de precisión.

El sistema fue desarrollado, instalado y empleado por el Departamento de Defensa de los Estados Unidos. Para poder obtener la posición de un objeto en el globo terráqueo, el sistema GPS está constituido por 24 satélites y utiliza la trilateración.

El funcionamiento de cualquier receptor GPS puede llevarse a cabo debido a la existencia de una red de 24 satélites que orbitan sobre el planeta tierra, a 20 200 km de altura, con trayectorias sincronizadas para cubrir toda la superficie de la Tierra.

Cuando se desea determinar la posición del receptor se utiliza la trilateración que básicamente consta de las siguientes fases:

- El receptor localiza automáticamente como mínimo cuatro satélites de la red. Cada satélite indica que el receptor se encuentra en un punto en la superficie de la esfera, con centro en el propio satélite y de radio la distancia total hasta el receptor.

- Obteniendo información de dos satélites queda determinada una circunferencia que resulta cuando se intersecan las dos esferas en algún punto de la cual se encuentra el receptor.

- Teniendo información de un tercer satélite, se elimina el inconveniente de la falta de sincronización entre los relojes de los receptores GPS y los relojes de los satélites. Asimismo se calcula el tiempo que tardan en llegar las señales al equipo, y de tal modo mide la distancia al satélite mediante el método de trilateración inversa.

- Con base en estas señales, el aparato sincroniza el reloj del GPS, la cual se basa en determinar la distancia de cada satélite respecto al punto de medición. Por último conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los satélites y conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtienen las posiciones absolutas o coordenadas reales del punto de medición. Es en este momento cuando el receptor GPS puede determinar una posición 3D exacta (latitud, longitud y altitud).

Actualmente Europa también posee un sistema propio de posicionamiento, éste se encuentra en vías de desarrollo. Se denomina Galileo y según los expertos es más rápido y preciso que GPS. Bibliografía

2.4. Cálculo de rutas

Una de las funcionalidades que más valor añadido ha generado a Buddy UC3M es el cálculo de rutas desde nuestra ubicación actual hasta cualquiera de los puntos predefinidos dentro del Campus de Colmenarejo.

Para esta app se ha empleado la API de Google Map para la visualización de mapas y su gestión. La decisión de utilizar la capa de Google es porqué la lógica del mapa ya se encuentra definida, a la hora de la construcción de la ruta sobre el objeto mapa nos encontramos con menos dificultades y con más funcionalidades, y por último Google posee un mapeo de prácticamente todo el mundo que no nos limita a la hora de obtener un mapa de cualquier ubicación.

En caso de que no hubiéramos optado por el uso de la API de Google Maps nos hubiéramos obligado a desarrollar algoritmos que fueran capaces de reconocer los diferentes elementos de un mapa, calcular distancias, calcular cual sería de todos los caminos disponibles el más cercano a un punto ubicado fuera de camino, etc.

Asimismo el principal requisito software que se siempre se trató de mantener en el desarrollo de la app era que todas sus funciones quedaran embebidas en ella. Se ha conseguido implementar de esta manera salvo para la función de navegación que se ha tenido que recurrir a la ejecución externa de la app Google Maps preinstalada en el dispositivo.

Durante el desarrollo de la app se pensaron diferentes formas de implementar el cálculo de las rutas y su posterior construcción sobre un objeto de Google Map.

Al intentar utilizar los servicios proporcionados por la API de Google Maps se ha detectado que a nivel general el servicio no permite entre otras cosas su uso para la navegación en tiempo real, el cálculo de rutas, guiado autónomo de vehículos, etc.

Lo anteriormente mencionado queda reflejado en los términos de servicio de las API's de los productos Google Map y Google Earth. Bibliografía

"Google Maps/Google Earth APIs Terms of Service

Last updated: May 27, 2009

...

10. License Restrictions. Except as expressly permitted under the Terms, or unless you have received prior written authorization from Google (or, as applicable, from the provider of particular Content), Google's licenses above are subject to your adherence to all of the restrictions below. Except as explicitly permitted in Section 7 or the

Maps APIs Documentation, you must not (nor may you permit anyone else to):
...
10.9 use the Service or Content with any products, systems, or applications for or in connection with:

(a) **real time navigation or route guidance, including but not limited to turn-by-turn route guidance that is synchronized to the position of a user's sensor-enabled device;"**

Es por ello por lo que se llega a la conclusión de que Google y su API no permite el cálculo de rutas dentro de un mapa Google.

Para solucionar esta restricción legal y poder seguir cumpliendo nuestro requisito de "todo embebido" de la app se ha desarrollado un método para conseguir la funcionalidad deseada.

Este consiste en dibujar la ruta obtenida sobre un objeto mapa que se encuentra centrado en unas coordenadas origen y destino.

En este procedimiento la acción que se realiza básicamente es el uso de la API del servicio Google Directions.

La explicación completa de la funcionalidad se realizará en el capítulo de implementación, no obstante se procede a resumir a grandes rasgos los pasos que se llevan a cabo para el cálculo de la ruta:

- Para poder llevar a cabo este proceso, una vez que tenemos recogida nuestra latitud y longitud actual se procede a calcular la ruta hasta uno de los 4 puntos predefinidos en la app pertenecientes al Campus de Colmenarejo.

- Para ello obtenemos una lista de puntos de direcciones (latitud y longitud), esta se obtiene filtrando la información de la ruta completa desde nuestro origen al destino que proporcionan los servicios web de Google Directions.

- Por último, el siguiente paso se trata en dibujar la ruta en el mapa que mostraremos en la app. Para ello se construye una línea poligonal formada por el conjunto de puntos obtenidos. Una vez construida se dibuja centrada sobre el mapa.

Las clases y métodos Java implementados para esta funcionalidad de la aplicación se reutilizan para los diferentes puntos de interés ya que el procedimiento para su cálculo es el mismo y sólo varía la posición destino.

3) Análisis

A lo largo de este capítulo se va a detallar aspectos importantes acerca del trabajo realizado en el análisis de Buddy UC3M. Para ello se realizaron las siguientes actividades durante esta fase:

3.1. Análisis del entorno físico

Antes de comenzar con el planteamiento de la aplicación se estuvo trabajando con las características cartográficas que el Campus de Colmenarejo nos ofrece.

Para ello se analizaron los diferentes caminos que hay construidos por todo el Campus y las posibilidades de rutas que ofrecían. Incluso se llegó a estudiar el tipo de camino que era, si peatonal o para tráfico rodado y si tenía alguna dirección en su circulación.

La idea en un principio era trabajar con teoría de grafos para el cálculo de los caminos mínimos, pero establecer una lógica cuando la persona se encuentra fuera un camino predefinido resultaba muy complejo y costoso. Había que desarrollar un mapa, definir todos sus vértices, aristas, pesos como distancias y lo más fundamental, un algoritmo que fuera capaz hacia que arista tendría que dirigirse el usuario cuando había varias a su alcance y éste se encontraba en medio.

Debido a estos condicionantes se estuvieron estudiando diferentes aplicaciones disponibles. Se estuvo trabajando con los mapas personales que Google te permitía crear, pero la cantidad de opciones, vértices y aristas a definir era muy numeroso y la aplicación daba problemas a la hora de manejarlos.

Finalmente y tras muchos buscar se encontró que Google Maps tenía la lógica de los caminos del Campus de Colmenarejo implementada y permitía la construcción de rutas desde cualquier punto ubicado dentro del Campus sin problema alguno.

Una vez que ya se supo sobre la plataforma sobre la que se iba a trabajar, se procedió a la recogida de los datos necesarios para la implementación de la app. Bibliografía

Se necesitó recoger las coordenadas geográficas (latitud, longitud) de los 4 puntos de interés hacia donde la app construiría las rutas desde la posición actual del usuario. Estos puntos fueron los siguientes:

- **Colegio Mayor Antonio Machado:** (40.545578, -4.012233)
- **Acceso principal Aulas edificio Miguel de Unamuno:** (40.543515, -4.013181)
- **Biblioteca Menéndez Pidal:** (40.542441, -4.012414)
- **Secretaria del Campus:** (40.542952, -4.012303)

3.2. Análisis de requisitos software

El requerimiento principal de esta aplicación es que ha sido desarrollada para ser ejecutada sobre el sistema operativo Android. Las app Android están basadas en el uso del lenguaje orientado a objetos Java combinado con el uso de XML que es empleado para el desarrollo de algunos de los componentes del software.

En cuanto al aspecto de los requisitos del dispositivo móvil respecto al rendimiento de la máquina estos no son muy exigentes. En principio cualquier teléfono móvil inteligente o Smartphone que sea capaz de ejecutar una versión del S.O. Android mayor o igual a Ice Cream Sándwich (v.4.0.x) podrá mover sin problemas Buddy UC3M.

Se puede decir que como requisitos software del dispositivo móvil indispensables para poder explotar todas las funcionalidades de la app serían:

- Acceso a Internet
- Servicios Google Play instalados
- App Google Maps instalada

Al querer desarrollar la aplicación para dispositivos móviles Android, había que buscar una tecnología que me pudiera brindar un desarrollo fiable y con cierto soporte que asegurara la disponibilidad de la app en todo momento, por ello se optó finalmente por utilizar la tecnología Google para implementar este PFC.

También cabe reseñar la disponibilidad de diferentes API`s online que ha facilitado el desarrollo de este proyecto.

En cuanto a los requisitos para el desarrollo de la aplicación se utilizó el entorno de desarrollo denominado Android Studio. Este entorno está accesible a través de Internet mediante descarga en el equipo sobre el que se trabajará.

Asimismo, al desarrollarse los componentes principales de la app en lenguaje Java se necesario la instalación de la máquina virtual de Java (JRE) para la compilación de las clases que consta el programa.

A continuación se procede a detallar formalmente los requisitos que ha de cumplir el software a desarrollar o requisitos funcionales así como los no funcionales anteriormente mencionados.

3.2.1. Requisitos funcionales

Cada requisito funcional de la app se especificará con el siguiente formato.

- **ID**: Clave unívoca que idéntica el requisito.

- **Descripción**: Explicación del requisito.

- **Importancia**: Grado de importancia del requisito.

- **Verificación**: Prueba/s que comprueban el requisito.

ID	RF-01
Descripción	La aplicación ha de calcular la ruta desde la ubicación actual del usuario hasta un punto prefijado.
Importancia	Alta
Verificación	P-01

Requisito funcional 1: Cálculo de ruta.

ID	RF-02
Descripción	La aplicación mostrará un mapa interactivo embebido del Campus de Colmenarejo.
Importancia	Alta
Verificación	P-02

Requisito funcional 2: Mapa del Campus.

ID	RF-03
Descripción	La aplicación mostrará la ubicación actual del usuario en un mapa vía satélite interactivo embebido.
Importancia	Alta
Verificación	P-03

Requisito funcional 3: Ubicación usuario mapa satélite.

ID	RF-04
Descripción	La aplicación permitirá navegar al usuario hacia el destino.
Importancia	Alta
Verificación	P-04

Requisito funcional 4: Navegación hasta un punto predefinido.

ID	RF-05
Descripción	La aplicación permitirá visualizar el entorno actual del usuario con imágenes en 360 grados.
Importancia	Media
Verificación	P-05, P-05B

Requisito funcional 5: Mostrar entorno con Google Street View.

ID	RF-06
Descripción	La aplicación mostrará los créditos del autor y tutor.
Importancia	Baja
Verificación	P-06

Requisito funcional 6: Créditos de aplicación.

ID	RF-07
Descripción	El terminal tiene que tener los servicios GPS habilitados
Importancia	Alta
Verificación	P-07

Requisito funcional 7: GPS Habilitado.

3.2.2. Requisitos no funcionales

A continuación se trataran los requisitos no funcionales de la aplicación, que son aquellos que salen fuera del ámbito de los funcionales como podría ser la disponibilidad, usabilidad, rendimiento, etc.

Cada requisito no funcional de la app se especificará con el siguiente formato.

- **ID**: Clave unívoca que idéntica el requisito.

- **Descripción**: Explicación del requisito.

- **Importancia**: Grado de importancia del requisito.

- **Verificación**: Prueba/s que comprueban el requisito.

ID	RNF-01
Descripción	La aplicación tiene que ser instalada en un terminal con sistema operativo Android.
Importancia	Alta
Verificación	P-08

Requisito no funcional 1: Sistema Operativo.

ID	RNF-02
Descripción	El terminal tiene que disponer de acceso a Internet.
Importancia	Alta
Verificación	P-09

Requisito no funcional 2: Acceso a Internet.

ID	RNF-03
Descripción	El terminal tiene que tener instalados los servicios de Google Play.
Importancia	Alta
Verificación	P-10

Requisito no funcional 3: Servicios Google Play.

ID	RNF-04
Descripción	El terminal tiene que tener instalada la aplicación Google Maps.
Importancia	Alta
Verificación	P-11

Requisito no funcional 2: App Google Maps

3.3. Análisis de requisitos hardware

Comenzaremos haciendo mención a los requisitos o requerimientos hardware que se han considerado para el SmartPhone o teléfono inteligente que ejecutará Buddy UC3M.

Como ya se comentó en capitulo anterior, los requisito hardware del dispositivo móvil no son muy elevados. Basta con un teléfono que tenga instalado un sistema Android con no más de 5 años de antigüedad, que posea 1 gigabyte de memoria RAM, 10 megabytes disponibles de almacenamiento (Buddy UC3M no llegaría a 4 MB) y como condición sine qua non que equipe receptor GPS.

Respecto al tipo de receptor GPS a utilizar no hay unas restricciones muy férreas. En principio no se recomienda el uso con receptores GPS monocanal.

Un receptor GPS monocanal es lento para alcanzar un posicionamiento, porque hasta que no terminan de recabar la información completa de las efemérides (los datos) de un satélite, no pasarán al siguiente satélite. Si se necesita al menos 4 satélites para obtener una posición tridimensional, lo mínimo que necesitas es un par de minutos para conseguirla (ya que en el mejor de los casos las efemérides (los datos) necesitan 30 segundos cada una para obtenerse completas).

Si estas en un sitio con alta cobertura de árboles, puede ocurrir que uno de ellos te haga sombra de la señal enviada por los satélites en un momento dado, y entonces en GPS pierde su posición, y con ella, el track que estás siguiendo. En esos momentos se pone a buscar un satélite sustituto, pero en ello, pierde cierto tiempo. Es por lo que por razones operativas, un GPS monocanal solo tiene en consideración (procesa) la información de 4 satélites como máximo. No se pueden obtener sobredeterminaciones que es cuando se usa la información que envían más de 4 satélites al mismo tiempo.

Actualmente lo más común es que los Smartphones cuenten con receptores GPS de 12 ó 20 canales.

Un receptor de 12 canales paralelos es como tener 12 receptores monocanales al mismo tiempo en una única unidad. Cuando enciendes el aparato, el GPS recepciona al mismo tiempo las señales de todos los satélites (hasta 12) que están en el hemisferio celeste en ese momento. En la mayor parte de las veces, le bastará 30 segundos (o menos) para alcanzar el posicionamiento. Además, no perderá su posición en sitios con alta cobertura de árboles, porque aun cuando pierda la señal de uno o más satélites, siempre tendrá disponibles otros para mantener su posicionamiento.

La última ventaja es que un 12 canales es capaz de procesar la información de todos los satélites al mismo tiempo, con lo que se obtienen posiciones sobredeterminadas que suelen ser más precisas.

Los receptores de 20 canales se diferencian de los de 12, en que los de 20 canales muestran un comportamiento más preciso en el cálculo de las coordenadas y un tiempo de respuesta menor en obtenerlas.

Como suele pasar con la mayoría de los dispositivos, cuanto más moderno y con unas especificaciones técnicas más modernas sea el equipo, más preferible será para el uso de Buddy UC3M.

Por último, hacer hincapié sobre los requisitos a tener en cuenta en el equipo informático para desarrollar el proyecto.

Cualquier labor de desarrollo de software requiere máquinas con una buena cantidad de memoria principal y un procesador rápido que permitan manejar la cantidad de operaciones a realizar con cierta solvencia.

Asimismo considero imprescindible la conexión a Internet del equipo, ya que con ello garantizas tener el software más actualizado y el acceso a diversas fuentes de documentación que podrán ser útiles.

Para el desarrollo de este PFC se ha utilizado el siguiente sistema informático con estas características:

- Microprocesador Intel Core i5-3570K - Procesador de 3ª generación (3,4 GHz, L3-Cache, zócalo LGA 1155, 77W TDP)
- Memoria RAM G.Skill Ripjaws-X - DIMM DDR3 (8 GB, 2133 MHz, 240 pines, 2 x 4 GB, CL9)
- Almacenamiento:
    - Disco duro interno de 256 GB, SATA III, 2.5" Crucial
    - Disco duro interno de 2 TB SATA III, 3.5" Seagate Barracuda
- Placa base Asrock Z77 Extreme4
- Tarjeta gráfica Gigabyte GeForce GTX 660 (2 GB, 1033 MHz, GDDR5-SDRAM, 192 bit)
- Monitor LED 24" LG Electronics E2442V (16:9, 1920×1080, 5.000.000:1)
- Conexión a internet de fibra óptica con 50Mbps de velocidad.

4) Diseño

4.1 Configuración entorno de desarrollo Android Studio

Una vez instalado ambos entornos es necesario proceder a su configuración. Esencialmente sólo debe hacerse en Android Studio, para ello hay que definir la ubicación donde se encuentra los .bin de Java, solucionar conflictos de dependencias con componentes de Android, definir el tipo de codificación (UTF-8) e instalar y configurar el entorno ADB para poder utilizar directamente un teléfono físico como emulador para poder realizar las pruebas que se consideren oportunas.

A la hora de la resolución de dependencias entre diferentes librerías, lo más probable es que el desarrollador se tenga que descargar nuevos paquetes según sus necesidades que se integrarán en el entorno de desarrollo.

En nuestro caso se añadieron la librería de Google Play y las relativas a versiones más modernas del sistema operativo Android.

4.2 Determinación tipo de APP Android

Una vez que ya se tiene configurado el entorno Android hay que decidir el tipo de app que se va a realizar.

Existen diferentes tipos de aplicaciones Android en cuanto al comportamiento y aspecto de su interfaz. Para ello a la hora de crear una nueva actividad en nuestro proyecto, nos encontramos con una serie de plantillas para facilitar nuestra labor de creación.

Android Studio nos mostrará una lista de plantillas, tales como Blank Activity, Fullscreen Activity, y Tabbed Activity, para crear una clase Activity en blanco, a pantalla completa o con pestañas respectivamente.

Buddy UC3M es una app que utiliza principalmente Fragments (marcos embebidos en la app que permiten individualizar las acciones en su interior) por lo que el uso de actividades Fullscreen o a pantalla completa producía ciertos problemas de estabilidad y rendimiento en la app.

Finalmente se optó por el uso de una actividad estándar sin comportamiento automático a pantalla completa. De este modo se podía definir de un modo preciso el comportamiento de la interfaz, ubicación de los elementos gráficos, etc.

4.3 Diseño interfaz de la APP

Una de las tareas más importantes en el desarrollo de cualquier aplicación es el diseño de su interfaz gráfica. En Android las aplicaciones van navegando entre pantallas.

Podría decirse que por regla general cada pantalla de la aplicación va asociada a una o varias actividades (o clases Java) y a un fichero XML que modeliza los elementos a mostrar en cada pantalla.

Para lograr una buena experiencia al usuario se ha optado por desarrollar los siguientes elementos en la app:

- **Pantalla de introducción ó Splash screen:**
  Es la primera pantalla resultante al ejecutar Buddy UC3M, se muestra el nombre de la app junto con una descripción y el logo de la Universidad Carlos III de Madrid.


17:08
BUDDY UC3M
SISTEMA DE UBICACIÓN UNIVERSITARIO
Universidad
Carlos III de Madrid
www.uc3m.es

- **Pantalla de bienvenida:**

  Como su nombre indica da la bienvenida al usuario, muestra mediante una notificación toast la longitud y latitud actual del usuario, le permite comenzar a usar la app y también le da la opción en la esquina superior derecha de navegar a los créditos de la app, donde se muestra la información del autor de la misma y el tutor del PFC.


17:08
Buddy UC3M
Bienvenid@ a Buddy UC3M
Mis coordenadas son: Latitud =
40.4046876 Longitud = -3.6620651
COMENZAR

- **Pantalla Créditos:**

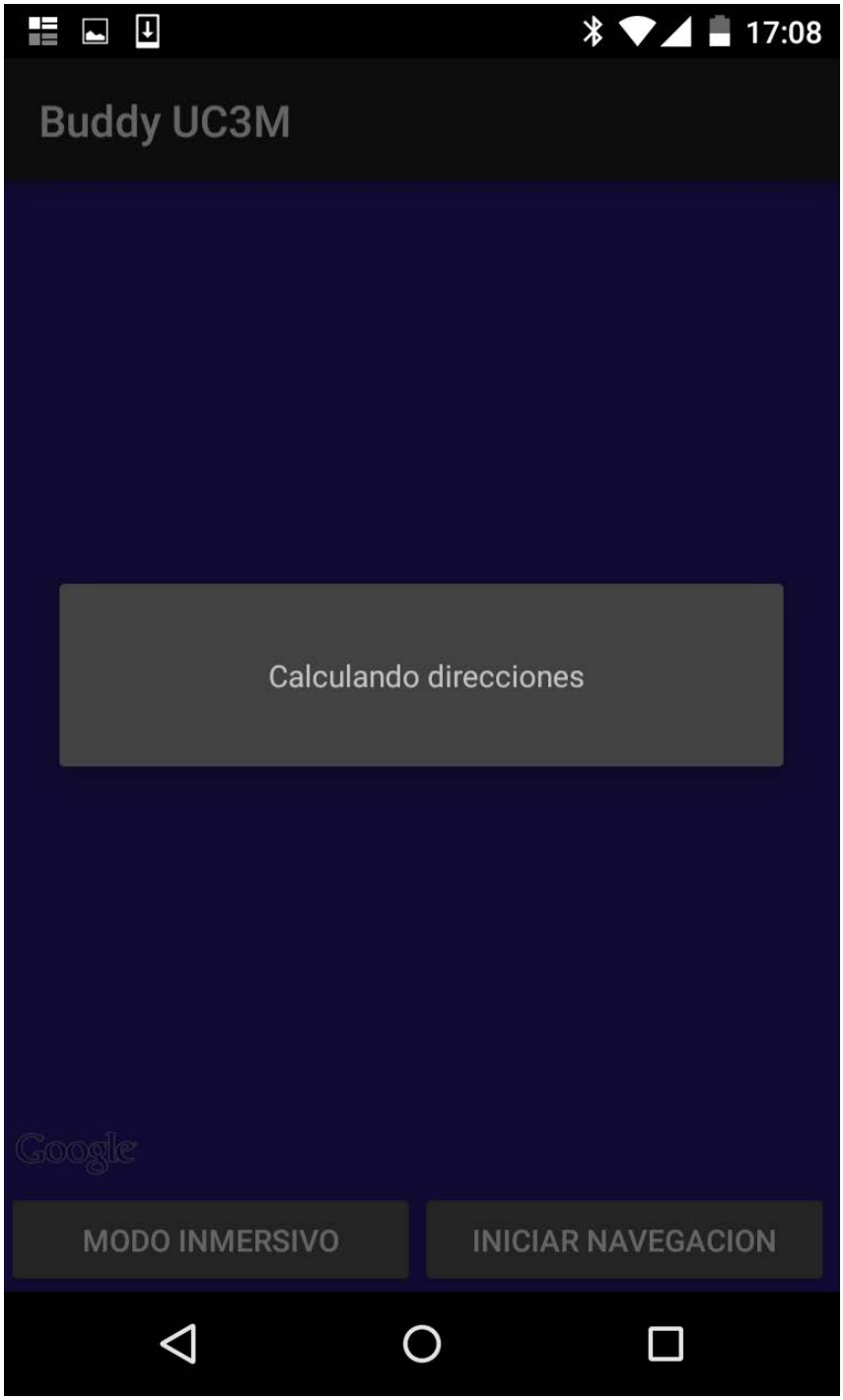
Como su nombre indica se muestran los créditos de la aplicación: autor, tutor.

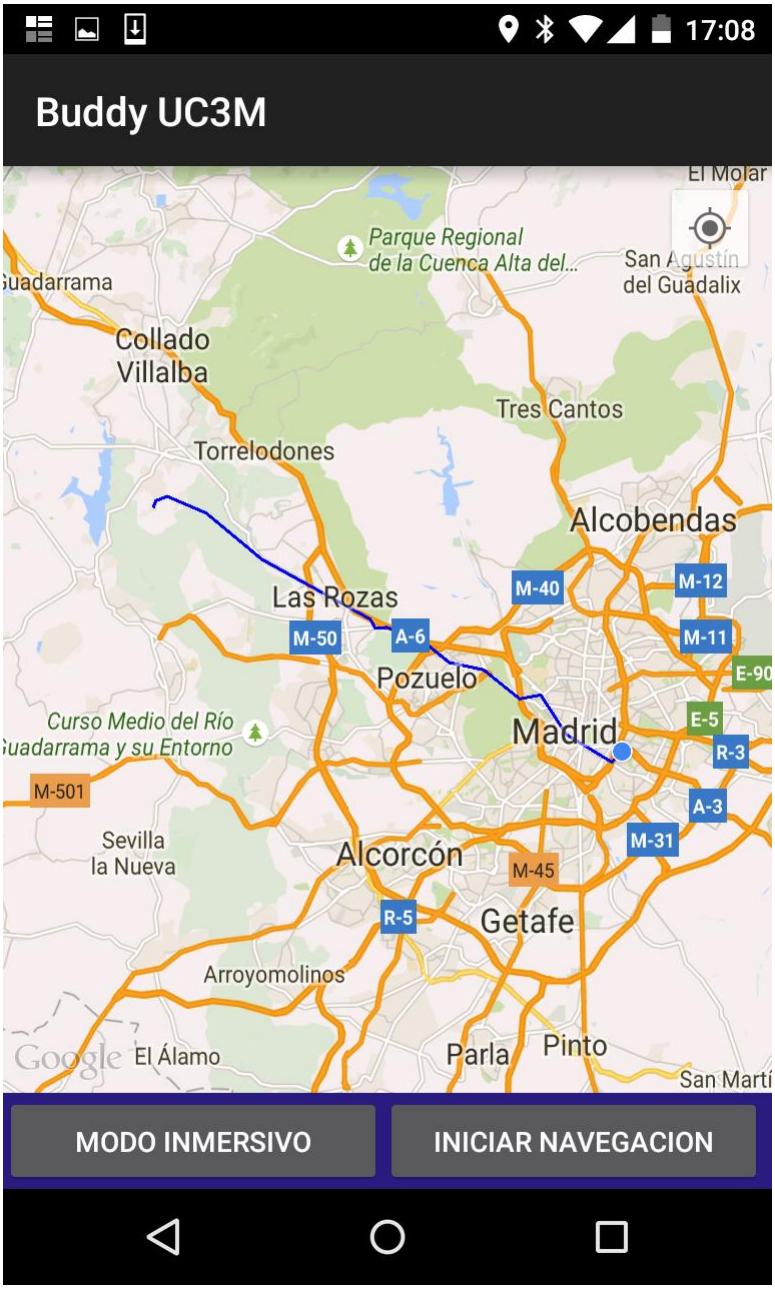
- **Pantalla menú de opciones:**

  En esta pantalla es donde se muestra al usuario todas las opciones posibles que podrá realizar con Buddy UC3M.

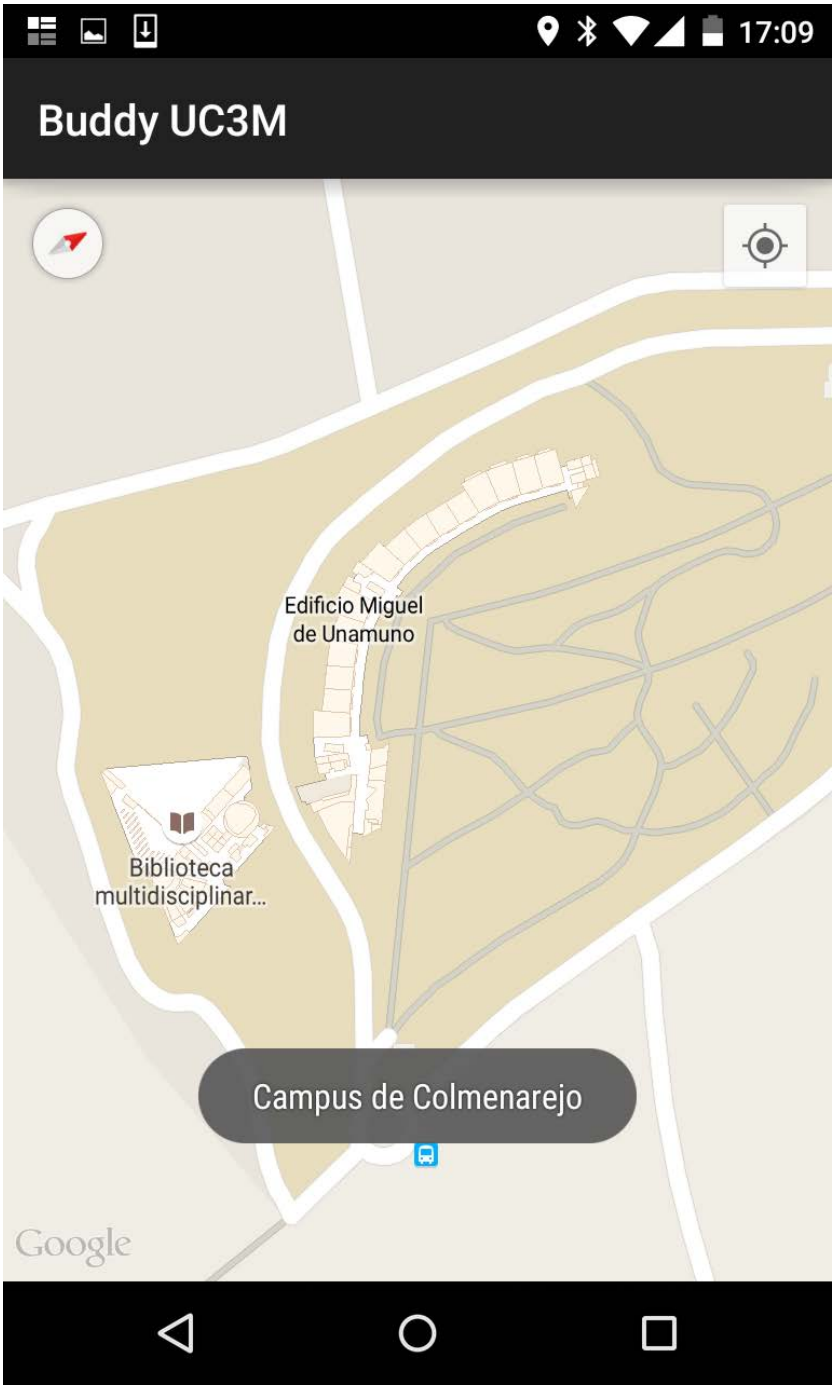
  En la pantalla se pueden observar seis iconos, baldosas o "tiles" que pulsándolos nos permiten hacer una de las 3 funciones que tiene Buddy UC3M.

Cuatro de ellas tienen la función del cálculo de la ruta dese la posición que nos encontramos hacia el punto de interés que se indica en el icono pulsado.


17:08
Buddy UC3M
Calculando direcciones
Google
MODO INMERSIVO
INICIAR NAVEGACION

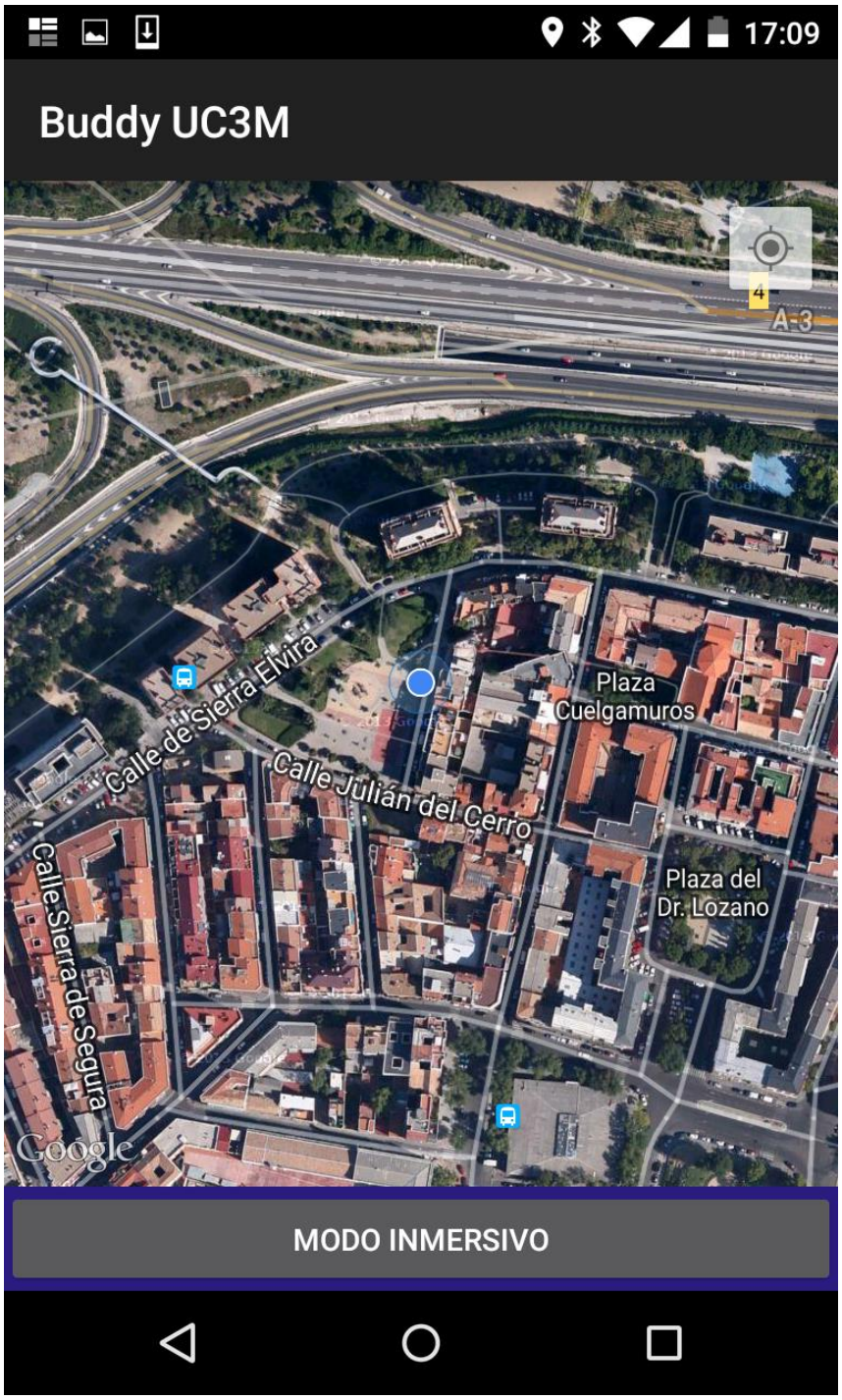


Otra opción es la de mostrar un mapa interactivo del Campus de Colmenarejo en el cual se muestran los edificios más representativos del Campus.



En este mapa se puede hacer zoom, inclinarlo, rotarlo o incluso seleccionar las diferentes plantas de los edificios mostrados. Si damos al botón disponible para mostrar nuestra ubicación actual y donde nos encontramos existen edificios altos, estos se mostraran en 3D en el mapa.

La última opción disponible es la de mostrar en una vista de satélite de la ubicación en la que se encuentra el usuario. Al igual que ocurre con la vista del mapa de la universidad el mapa que se muestra es completamente interactivo.

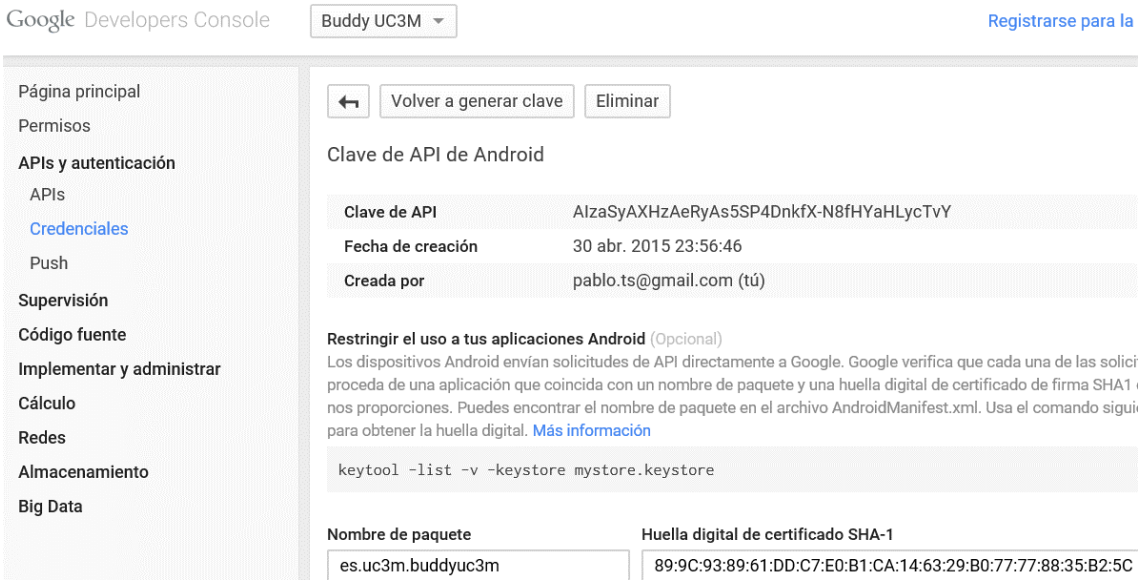


También se puede mostrar imágenes reales en 360º del lugar donde nos encontramos.

4.4 Obtención Google API Key

Cualquier aplicación que pretenda usar los servicios que proporciona Google necesitará de un clave API de desarrollador o Google API Key.

Esta clave hay que solicitarla a Google y entonces se nos proporciona un token asociado a nuestra cuenta de usuario y que tiene que ser introducido en el archivo Manifest de nuestra app.



Cuando la app necesita acceder a los servidores Google para utilizar sus servicios como puede ser obtener un mapa, cálculo de indicaciones, etc. Lo primero que hace es autenticar que tenemos acceso a esos servicios y no hemos superado la cuota mensual asignada.

Una Google API Key incorrecta o el haber superado la cuota asignada, implicaría que la app en cuestión dejara de funcionar en cuanto a las funciones que necesiten servicios Google.

La navegación anteriormente expuesta es aplicable cuando realizamos una navegación "hacia adelante" en la app. Para el retroceso en las pantallas finalmente se optó por la gestión automatiza que posee Android de la pila de actividades. Sólo en un caso muy particular como cuando no tenemos el GPS activado se destruye la actividad correspondiente a ajustes del sistema una vez que se activa el GPS. Luego se devuelve el control a la aplicación.

4.5 Ámbito y limitaciones de Google Maps API

Las Google APIs son un conjunto de APIs desarrolladas por Google que permiten la comunicación con los servicios de Google y la integración de estas en otros servicios. Tal y como se ha explicado en el apartado anterior el uso de Google API requiere la autenticación y autorización del usuario a través del protocolo OAuth 2.0. Bibliografía

En cuanto a Google Maps API, podría decirse que ofrece las siguientes características:

- Renderización 3D de mapas.
- Carga vectorial de mapas que acelera su carga y minimiza el ancho de banda a emplear.
- Interior de los edificios detallados por planta.
- Dibujar sobre el mapa polilíneas, polígonos y marcadores personalizados.
- Control de gestos, permitiendo hacer zoom, rotar e inclinar los mapas mediante gestos predeterminados.
- Servicios de localización, que permiten determinar la ubicación para su uso en las aplicaciones.

4.5.1. Particularidades del servicio de direcciones y sus limitaciones

Una de las funcionalidades de la API de Google es el servicio Google Directions. Este servicio permite el realizar solicitudes vía web para el cálculo de direcciones desde un punto origen a un punto destino proporcionado. El proceso se realiza mediante el uso de Javascript.

Como ya se comentó en el capítulo anterior, actualmente Google y su API no permite el cálculo de rutas dentro de un mapa Google. Esto queda plasmado en los términos de servicio de las API's de los productos Google Map y Google Earth. Bibliografía

"Google Maps/Google Earth APIs Terms of Service

Last updated: May 27, 2009

...

10. License Restrictions. Except as expressly permitted under the Terms, or unless you have received prior written authorization from Google (or, as applicable, from the provider of particular Content), Google's licenses above are subject to your adherence to all of the restrictions below. Except as explicitly permitted in Section 7 or the Maps APIs Documentation, you must not (nor may you permit anyone else to):

...

10.9 use the Service or Content with any products, systems, or applications for or in connection with:

(a) **real time navigation or route guidance, including but not limited to turn-by-turn route guidance that is synchronized to the position of a user's sensor-enabled device;"**

Por lo que hubo que utilizar otro método para el cálculo de rutas embebidas en la app. Es por ello que en el capítulo siguiente, el de implementación, se procederá a detallar los algoritmos desarrollados para su cálculo.

5) Implementación

La finalidad en este capítulo es la explicar la estructura y la codificación de cada uno de los componentes implementados para la app. Se va a detallar aspectos relativos al fichero Manifest, las actividades, ficheros XML e interfaz gráfica.

Asimismo también se procederá a analizar los principales algoritmos y soluciones implementadas para alcanzar las funcionalidades buscadas.

Para el desarrollo de la aplicación se ha tratado de utilizar una metodología de desarrollo lineal o modelo en cascada, aunque veces este proceso se ha visto mezclado con la utilización de un framework iterativo o desarrollo prototipado ya que a medida que se iba confeccionando el software se realizaban pruebas para que la app se fuera ajustando a los requerimientos acordados.

Una aplicación Android se estructura mediante actividades que son clases Java que se asocian a un fichero XML donde se indica el comportamiento y presentación de la interfaz. Es por tanto que la gran mayoría de las clases se han implementado utilizando la herencia de la clase Activity.

5.1. Fichero Manifest

A continuación y tras la explicación en el apartado 2.1.1 de las características de un fichero Manifest dentro de la estructura de una app Android, se procede a detallar el implementado para Buddy UC3M.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="es.uc3m.buddyuc3m"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="22" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.LOCATION_HARDWARE" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
    <meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyAXHzAeRyAs5SP4DnkfX-N8fHYaHLycTvY" />

    <android:uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <android:uses-permission
        android:name="android.permission.READ_EXTERNAL_STORAGE"
        android:maxSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name" >
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyAXHzAeRyAs5SP4DnkfX-N8fHYaHLycTvY" />

        <activity
            android:name=".SplashScreenActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".WelcomeActivity"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".OptionActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait" >
        </activity>
```

```
        <activity
            android:name=".MapsActivity"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".MapsActivityP"
            android:configChanges="screenSize|orientation"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".MapsActivityGPS"
            android:configChanges="screenSize|orientation"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".MapaUniversidad"
            android:configChanges="screenSize|orientation"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name=".MapsRouteB"
            android:configChanges="screenSize|orientation"
            android:label="@string/title_activity_maps_route_b" >
        </activity>
        <activity
            android:name=".Creditos"
            android:label="@string/title_activity_creditos"
            android:screenOrientation="portrait" >
        </activity>
    </application>
</manifest>
```

Tal y como se puede apreciar es un fichero XML donde se van indicando ciertos atributos característicos de la app. A continuación explicaremos las partes más relevantes del mismo.

Lo primero que se puede observar es que indicamos el nombre del paquete que hemos implementado, es.uc3m.buddyuc3m, también se puede observar el número de versión de la app.

A continuación se indican los permisos de sistema que va a necesitar nuestra app para su correcto funcionamiento. Estos son los permisos que cuando el usuario procede a instalar la app tendrá que autorizar para que se lleve de un modo correcto la misma.

```
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.LOCATION_HARDWARE" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
```

Nuestra app tal y como se aprecia en la figura superior necesita acceso a internet para establecer la conexión a los servicios de Google para poder usarlos.

También necesita acceder al estado de la red, a la obtención inicial de nuestra ubicación aproximada (coarse_location), a la ubicación más precisa, a la memoria del

dispositivo para su instalación y por último al hardware del receptor GPS instalado en el Smartphone.

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"                                                >
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version"                      />
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyAXHzAeRyAs5SP4DnkfX-N8fHYaHLycTvY"                    />

        <activity
            android:name=".SplashScreenActivity"
            android:label="@string/app_name"                                            >
            <intent-filter>
                <action                android:name="android.intent.action.MAIN"       />

                <category            android:name="android.intent.category.LAUNCHER"    />
            </intent-filter>
        </activity>
        <activity
            android:name=".WelcomeActivity"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:label="@string/app_name"                                            >
        </activity>
        <activity
            android:name=".OptionActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait"                                        >
        </activity>
        <activity
            android:name=".MapsActivity"
            android:label="@string/app_name"                                            >
        </activity>
        <activity
            android:name=".MapsActivityP"
            android:configChanges="screenSize|orientation"
            android:label="@string/app_name"                                            >
        </activity>
        <activity
            android:name=".MapsActivityGPS"
            android:configChanges="screenSize|orientation"
            android:label="@string/app_name"                                            >
        </activity>
        <activity
            android:name=".MapaUniversidad"
            android:configChanges="screenSize|orientation"
            android:label="@string/app_name"                                            >
        </activity>
        <activity
            android:name=".MapsRouteB"
            android:configChanges="screenSize|orientation"
            android:label="@string/title_activity_maps_route_b"                         >
        </activity>
        <activity
            android:name=".Creditos"
            android:label="@string/title_activity_creditos"
            android:screenOrientation="portrait"                                        >
        </activity>
    </application>
```

Dentro de la etiqueta application lo primero que nos encontramos es la definición del icono de la app y el nombre que tendrá.

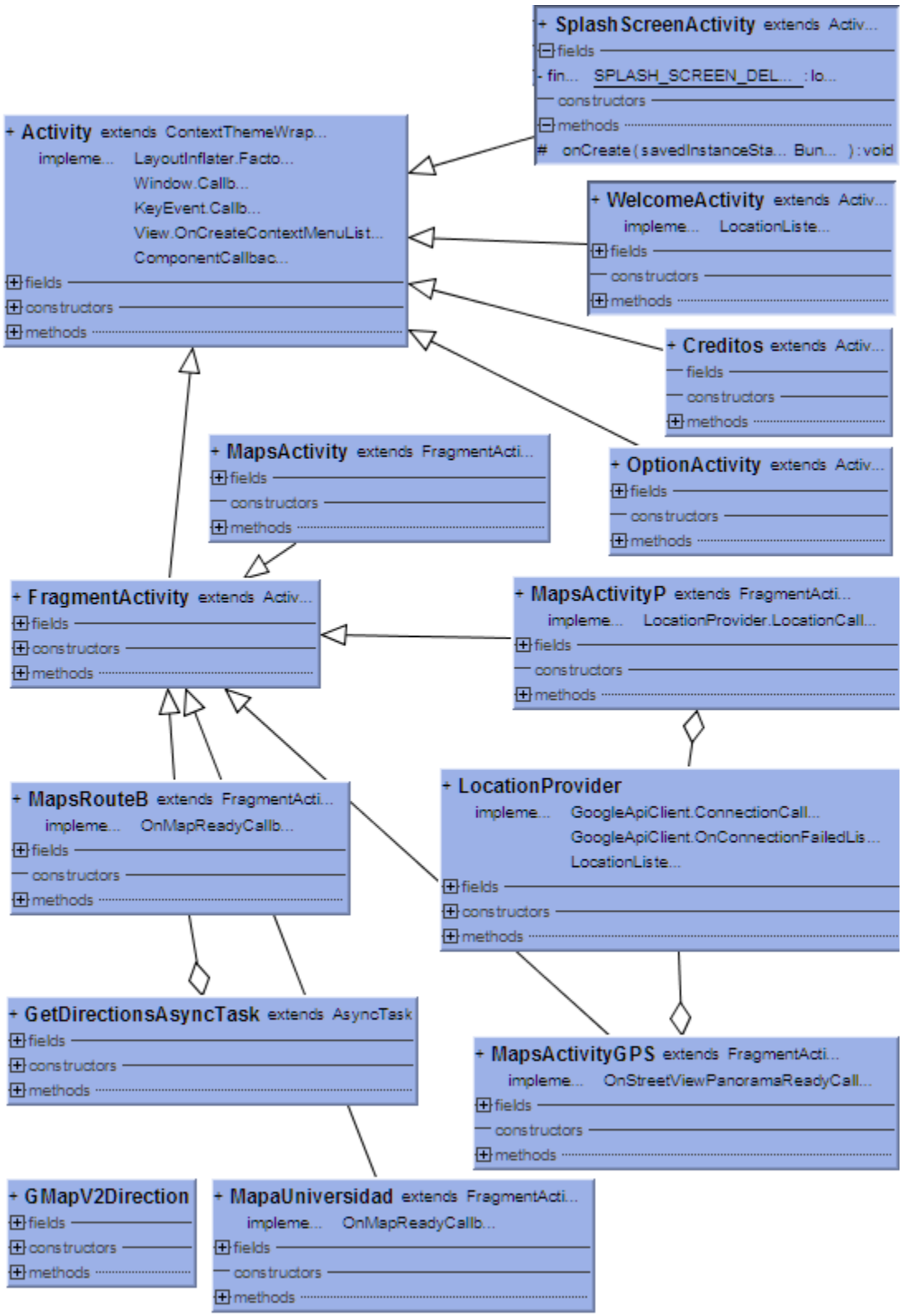
A continuación dos etiquetas meta-data que indican que por una parte que la app necesitará usar los servicios de Google Play y por último algo fundamental, el token de nuestra Google API Key que nos privilegia a usar los servicios Google.

Por último pueden apreciarse todas las actividades por las que está conformada Buddy UC3M. Hacer especial mención a la actividad SplashScreenActivity donde se puede apreciar que es la comienza la app y es usada como launcher.

5.2. Diagrama de clases

En el siguiente diagrama de clases se pueden observar las relaciones que tienen las diferentes clases entre ellas.

Tal y como se puede observar sólo se hereda de la clases Activity.java y FragmentActivity.java. La diferencia entre realizar la herencia de una u otra es básicamente si se van a manejar Fragments o no, que lógicamente coincide cuando se lleva a cabo el uso de mapas o panoramas embebidos en la aplicación.

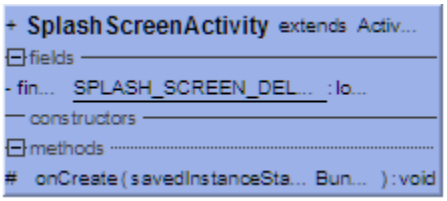

+ SplashScreenActivity extends Activ...
fields
- fin... SPLASH_SCREEN_DEL... : lo...
constructors
methods
onCreate(savedInstanceSta... Bun...):void
+ Activity extends ContextThemeWrap...
impleme... LayoutInflater.Facto...
Window.Callb...
KeyEvent.Callb...
View.OnCreateContextMenuList...
ComponentCallbac...
fields
constructors
methods
+ WelcomeActivity extends Activ...
impleme... LocationListe...
fields
constructors
methods
+ Creditos extends Activ...
fields
constructors
methods
+ MapsActivity extends FragmentActi...
fields
constructors
methods
+ OptionActivity extends Activ...
fields
constructors
methods
+ FragmentActivity extends Activ...
fields
constructors
methods
+ MapsActivityP extends FragmentActi...
impleme... LocationProvider.LocationCall...
fields
constructors
methods
+ MapsRouteB extends FragmentActi...
impleme... OnMapReadyCallb...
fields
constructors
methods
+ LocationProvider
impleme... GoogleApiClient.ConnectionCall...
GoogleApiClient.OnConnectionFailedLis...
LocationListe...
fields
constructors
methods
+ GetDirectionsAsyncTask extends AsyncTask
fields
constructors
methods
+ MapsActivityGPS extends FragmentActi...
impleme... OnStreetViewPanoramaReadyCall...
fields
constructors
methods
+ GMapV2Direction
fields
constructors
methods
+ MapaUniversidad extends FragmentActi...
impleme... OnMapReadyCallb...
fields
constructors
methods

5.3. Actividades

Las actividades son clases Java donde se codifica la parte lógica de la app. A continuación se procede a detallar la mayoría de las clases que contienen el código relevante para el funcionamiento de Buddy UC3M. Existen otras clases pero que no añaden funcionalidad a la aplicación sino que son más bien para sustentar la interfaz de usuario.

5.3.1. SplashScreenActivity.java

En esta clase la utilizamos para el inicio de la app. Básicamente lo que hace es mostrar la pantalla de bienvenida 4 segundos y automáticamente ejecuta la siguiente actividad.




```
package es.uc3m.buddyuc3m;

import java.util.Timer;
import java.util.TimerTask;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.Window;

public class SplashScreenActivity extends Activity {

    // Aqui fijamos la duración de la pantalla de presentacion cuando la app se inicia.
    private static final long SPLASH_SCREEN_DELAY = 4000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Se fija el tipo de orientacion
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        // Ocultamos la barra del titulo
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.splash_screen);
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                // Comienzo de la siguiente actividad
                Intent mainIntent = new Intent().setClass(
                        SplashScreenActivity.this, WelcomeActivity.class);
                startActivity(mainIntent);
                // Cerramos la actividad para que al dar al botón atrás no se
                // pueda volver a ella.
                finish();
            }
        };
        // Simulamos la carga larga de la app.
        Timer timer = new Timer();
        timer.schedule(task, SPLASH_SCREEN_DELAY);
```

```
    }
}
```

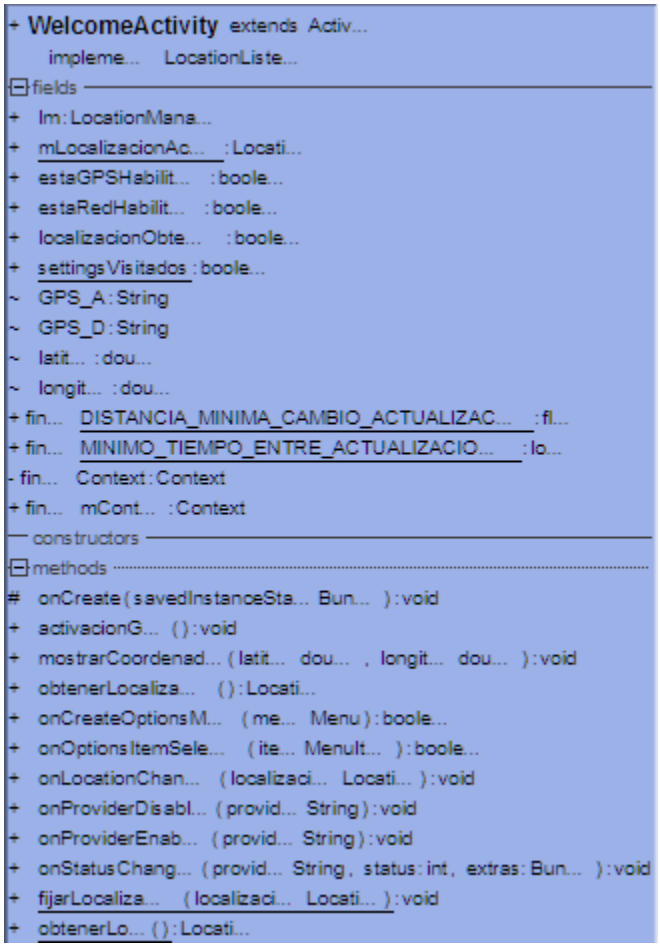
5.3.2. WelcomeActivity.java

En esta actividad a parte de dar la bienvenida al usuario también se obtiene la ubicación del usuario accediendo directamente a la red y el sensor GPS

En caso de que el GPS no se encuentre activo en el teléfono, la app lanza un cuadro de dialogo al usuario para que lo active y le envía a las propiedades del sistema para que así lo haga si lo desea.

Una vez que hemos obtenido la localización se procede a fijarla y es cuando la app ya estaría en disposición de navegar hasta la actividad OptionActivity si el usuario pulsa el botón COMENZAR.

Indicar que esta actividad hereda los métodos de la clase Activity ya predefinida.




```
package es.uc3m.buddyuc3m;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.provider.Settings;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesUtil;


public class WelcomeActivity extends Activity implements LocationListener{
    public LocationManager lm;
    public static Location mLocalizacionActual;
    public boolean estaGPSHabilitado ;
```

```
    public boolean estaRedHabilitada ;
    public boolean localizacionObtenida;
    public static boolean settingsVisitados = false;
    String GPS_A = "Gps Activado";
    String GPS_D = "Gps Desactivado";
    double latitud;
    double longitud;
    public static final float DISTANCIA_MINIMA_CAMBIO_ACTUALIZACION = 5; // 10 metros
    public static final long MINIMO_TIEMPO_ENTRE_ACTUALIZACIONES = 1000 * 10 * 1; // 10
segundos
    private  final Context Context = WelcomeActivity.this;
    public final android.content.Context mContext = WelcomeActivity.this;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        obtenerLocalizacion();
        fijarLocalizacion(mLocalizacionActual);
        if (estaGPSHabilitado){
            mostrarCoordenadas(latitud,longitud);
        }
        setContentView(R.layout.activity_welcome);
        findViewById(R.id.button_options).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (estaGPSHabilitado) {
                    fijarLocalizacion(mLocalizacionActual);
                    startActivity(new Intent(WelcomeActivity.this, OptionActivity.class));
                } else {
                    obtenerLocalizacion();
                    estaGPSHabilitado = lm.isProviderEnabled(LocationManager.GPS_PROVIDER);
                    if (estaGPSHabilitado){
                        fijarLocalizacion(mLocalizacionActual);
                        startActivity(new Intent(WelcomeActivity.this,
OptionActivity.class));
                    }
                }
            }
        });
    }
    public void activacionGPS(){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("¿Utilizar la ubicación?");
        builder.setMessage("Buddy UC3M necesita saber tu posición. Por favor, activa el
GPS");
        builder.setPositiveButton("ACTIVAR GPS", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialogInterface, int i) {
                //Para saber si ha visitado los settings y luego borrar de la pila la
actividad.
                settingsVisitados = true;
                // Muestra los settings de localizacion cuando el usuario le da a activar
el GPS.
                Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                startActivity(intent);
            }
        });
        builder.setNegativeButton("NO, FINALIZAR", new DialogInterface.OnClickListener() {
            public void onClick(final DialogInterface dialog, @SuppressWarnings("unused")
final int id) {
                dialog.cancel();
                finish();
            }
        });
        Dialog alertDialog = builder.create();
        alertDialog.setCanceledOnTouchOutside(false);
        alertDialog.show();
    }
    public void mostrarCoordenadas(double latitud, double longitud){
        String coordenadas = "Mis coordenadas son: " + "Latitud = " + latitud + " Longitud
= " + longitud;
        Toast.makeText( getApplicationContext(),coordenadas,Toast.LENGTH_LONG).show();
    }
    public Location obtenerLocalizacion() {
        try {
```

```
            int chkGooglePlayServices = GooglePlayServicesUtil
                    .isGooglePlayServicesAvailable(Context);

            if (chkGooglePlayServices != ConnectionResult.SUCCESS) {
                GooglePlayServicesUtil.getErrorDialog(chkGooglePlayServices,
                        (Activity) mContext, 1122).show();
            } else {

                lm = (LocationManager) getSystemService(mContext.LOCATION_SERVICE);
                Criteria criteria = new Criteria();
                criteria.setAccuracy(Criteria.ACCURACY_FINE);

                // esta el GPS habilitado?
                estaGPSHabilitado = lm.isProviderEnabled(LocationManager.GPS_PROVIDER);

                // esta la red habilitada?
                estaRedHabilitada = lm.isProviderEnabled(LocationManager.NETWORK_PROVIDER);

                if (!estaGPSHabilitado) {
                    activacionGPS();
                } else {
                    this.localizacionObtenida = true;
                    // Primero obtenemos la localizacion de un proveedor de red
                    if (estaRedHabilitada) {

lm.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,MINIMO_TIEMPO_ENTRE_ACTUALIZACIO
NES,DISTANCIA_MINIMA_CAMBIO_ACTUALIZACION, this);
                        Log.d("Red", "Red");
                        if (lm != null) {
                            mLocalizacionActual = lm

.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                            if (mLocalizacionActual != null) {
                                latitud = mLocalizacionActual.getLatitude();
                                longitud = mLocalizacionActual.getLongitude();
                            }
                        }
                    }
                    // si el GPS esta habilitado obtenemos latitud y longitud usando los
servicios del GPS
                    if (estaGPSHabilitado) {
                        if (mLocalizacionActual == null) {

lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,MINIMO_TIEMPO_ENTRE_ACTUALIZACIONES,
DISTANCIA_MINIMA_CAMBIO_ACTUALIZACION, this);
                            Log.d("GPS Habilitado", "GPS Habilitado");
                            if (lm != null) {
                                mLocalizacionActual = lm

.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                                if (mLocalizacionActual != null) {
                                    latitud = mLocalizacionActual.getLatitude();
                                    longitud = mLocalizacionActual.getLongitude();

                                }
                            }
                        }
                    }
                }

            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return mLocalizacionActual;
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_welcome, menu);
        return true;
    }
    @Override
```

```
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_acerca) {
            startActivity(new Intent(WelcomeActivity.this, Creditos.class));
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
    public void onLocationChanged(Location localizacion) {
        mLocalizacionActual = localizacion;
    }
    public void onProviderDisabled(String provider){
        Toast.makeText(getApplicationContext(), GPS_D, Toast.LENGTH_SHORT).show();
    }
    public void onProviderEnabled(String provider){
        Toast.makeText(getApplicationContext(), GPS_A, Toast.LENGTH_SHORT).show();
    }
    public void onStatusChanged(String provider, int status, Bundle extras){}
    public static void fijarLocalizacion(Location localizacion) {
        mLocalizacionActual = localizacion;
    }
    public static Location obtenerLoca() {
        return mLocalizacionActual;
    }
}
```

A continuación se procede a explicar la codificación de los métodos más relevantes dentro de la actividad.

- **Método obtenerLocalizacion:**

```
public Location obtenerLocalizacion() {
    try {
        int chkGooglePlayServices = GooglePlayServicesUtil
                .isGooglePlayServicesAvailable(Context);

        if (chkGooglePlayServices != ConnectionResult.SUCCESS) {
            GooglePlayServicesUtil.getErrorDialog(chkGooglePlayServices,
                    (Activity) mContext, 1122).show();
        } else {

            lm = (LocationManager) getSystemService(mContext.LOCATION_SERVICE);
            Criteria criteria = new Criteria();
            criteria.setAccuracy(Criteria.ACCURACY_FINE);

            // esta el GPS habilitado?
            estaGPSHabilitado = lm.isProviderEnabled(LocationManager.GPS_PROVIDER);

            // esta la red habilitada?
            estaRedHabilitada = lm.isProviderEnabled(LocationManager.NETWORK_PROVIDER);

            if (!estaGPSHabilitado) {
                activacionGPS();
            } else {
                this.localizacionObtenida = true;
                // Primero obtenemos la localización de un proveedor de red
                if (estaRedHabilitada) {

lm.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,MINIMO_TIEMPO_ENTRE_ACTUALIZACIO
NES,DISTANCIA_MINIMA_CAMBIO_ACTUALIZACION, this);
                    Log.d("Red", "Red");
                    if (lm != null) {
                        mLocalizacionActual = lm
```

```
                                .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                        if (mLocalizacionActual != null) {
                            latitud = mLocalizacionActual.getLatitude();
                            longitud = mLocalizacionActual.getLongitude();
                        }
                    }
                }
                // si el GPS está habilitado obtenemos latitud y longitud usando los
servicios del GPS
                if (estaGPSHabilitado) {
                    if (mLocalizacionActual == null) {

lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,MINIMO_TIEMPO_ENTRE_ACTUALIZACIONES,
DISTANCIA_MINIMA_CAMBIO_ACTUALIZACION, this);
                        Log.d("GPS Habilitado", "GPS Habilitado");
                        if (lm != null) {
                            mLocalizacionActual = lm
                                    .getLastKnownLocation(LocationManager.GPS_PROVIDER);
                            if (mLocalizacionActual != null) {
                                latitud = mLocalizacionActual.getLatitude();
                                longitud = mLocalizacionActual.getLongitude();

                            }
                        }
                    }
                }
            }

        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    return mLocalizacionActual;
}
```

Lo primero que se verifica es si el terminal donde está instalada la app consta de los servicios Google Play instalados. Si no es así mostrará un mensaje emergente invitando al usuario a su instalación.

A continuación se crea un objeto del tipo Criterio donde se guarda que la localización a utilizar será lo más precisa posible.

Posteriormente almacenamos en unas variables booleanas si tenemos acceso a la red y si está el GPS habilitado.

Si no se encontrara el GPS activado ejecutamos el método **activacionGPS**:

```
public void activacionGPS(){
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("¿Utilizar la ubicación?");
    builder.setMessage("Buddy UC3M necesita saber tu posición. Por favor, activa el GPS");
    builder.setPositiveButton("ACTIVAR GPS", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialogInterface, int i) {
            //Para saber si ha visitado los settings y luego borrar de la pila la
actividad.
            settingsVisitados = true;
            // Muestra los settings de localización cuando el usuario le da a activar el
GPS.
            Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
            startActivity(intent);
        }
    });
```

```
    builder.setNegativeButton("NO, FINALIZAR", new DialogInterface.OnClickListener() {
        public void onClick(final DialogInterface dialog, @SuppressWarnings("unused") final
int id) {
            dialog.cancel();
            finish();
        }
    });
    Dialog alertDialog = builder.create();
    alertDialog.setCanceledOnTouchOutside(false);
    alertDialog.show();
}
```

En este método mostramos un mensaje emergente al usuario solicitándole que active el GPS. Una vez activado devolvemos el control a nuestra app mediante finish().

Continuando con el método obtener localización, ya tendríamos el GPS activado. Como los servicios de ubicación vía GPS tardan un tiempo en llevar a cabo la trilateración, si la red se encuentra activa obtenemos la ubicación de ella y si lo está el GPS pues del mismo.

Para almacenar estos datos se utiliza un gestor de localización al que se le aplican como parámetros para que lleve a cabo la actualización de la posición si nos movemos más de 10 metros o bien si han pasado 10 segundos desde la última actualización. Esta forma de refrescar la posición consume más batería pero así nos garantizamos la mayor precisión posible en nuestra app.

Finalmente la ubicación actual se guarda en un objeto del tipo Location que es el que devuelve el método.

- **Método fijarLocalizacion:**

```
public static void fijarLocalizacion(Location localización) {
    mLocalizacionActual = localización;
}
```

Este método va asignando a la variable mLocalizacionActual la nueva posición cada vez que se produce una nueva variación.

- **Método mostrarCoordenadas:**

```
public void mostrarCoordenadas(double latitud, double longitud){
    String coordenadas = "Mis coordenadas son: " + "Latitud = " + latitud + " Longitud = "
+ longitud;
    Toast.makeText( getApplicationContext(),coordenadas,Toast.LENGTH_LONG).show();
}
```

En este método se muestra una notificación toast por pantalla con las coordenadas de la ubicación actual. Es decir, longitud y latitud. Para que el funcionamiento del

método sea el esperado este debería de ser invocado cuando tengamos una posición válida, sino se mostrará 0,0.

- **Método obtenerLoca:**

```
public static Location obtenerLoca() {
    return mLocalizacionActual;
}
```

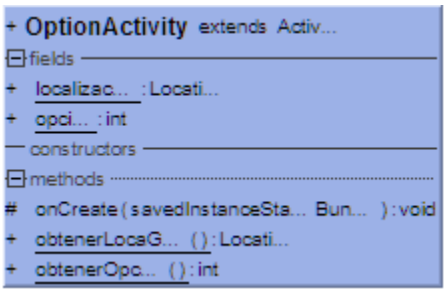
Este método se ha implementado para poder pasar la variable de la posición actual a las diferentes clases, ya que esta va siendo actualizada según nos vamos desplazando con nuestro Smartphone.

En la clase anteriormente explicada existen más métodos que provienen de la clase LocationListener. Algunos de ellos se han sobrescrito para que ante su invocación automática según vayan surgiendo determinados eventos tengan el comportamiento deseado y no el predeterminado. Por ejemplo, cuando estamos en la app y desactivamos/activamos el GPS se nos muestra una notificación toast al respecto para ser consciente de ello.

5.3.3. OptionActivity.java

En esta actividad es donde se da opción al usuario para elegir que desea realizar, como la clase anterior también hereda de la clase Activity.java.

Podrá o bien calcular la mejor ruta desde su ubicación actual a cualquiera de los 4 puntos predeterminados (Biblioteca Menéndez Pidal, las aulas del Edificio Miguel de Unamuno, la residencia de estudiantes Antonio Machado y el acceso a la secretaría del centro), mostrar el mapa interactivo del Campus de Colmenarejo o por último un mapa satélite de su ubicación actual incluso pudiendo observar imágenes reales de su alrededor.




```
package es.uc3m.buddyuc3m;

import android.app.Activity;
import android.content.Intent;
import android.location.Location;
import android.os.Bundle;
import android.view.View;

public class OptionActivity extends Activity {
    public static Location localizacion;
    public static int opcion;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        localizacion = WelcomeActivity.obtenerLoca();
        setContentView(R.layout.activity_options);
        //Para mostrar el mapa del campus:
        findViewById(R.id.imageCampus).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(OptionActivity.this, MapaUniversidad.class));
            }
        });
        //StreetView
        //Mostramos la ubicacion actual.
        findViewById(R.id.imageUbicacion).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(OptionActivity.this, MapsActivityP.class));
            }
        });
        findViewById(R.id.imageBiblioteca).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

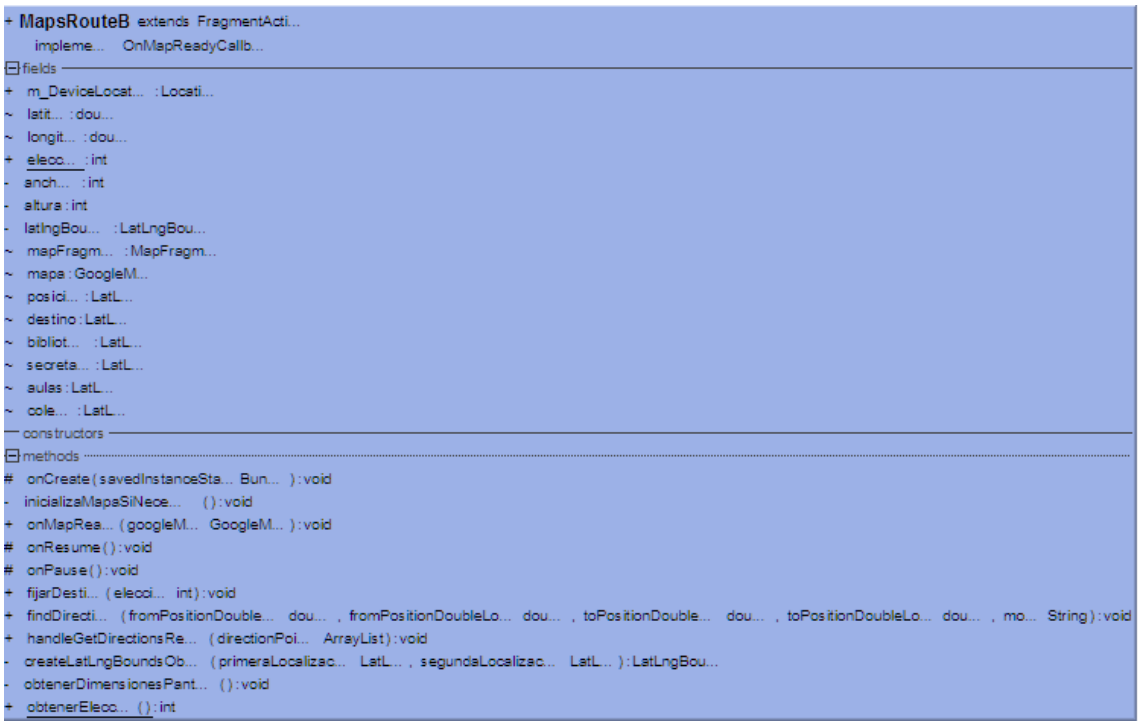
```
                //Para ir a la biblioteca
                opcion = 1;
                startActivity(new Intent(OptionActivity.this, MapsRouteB.class));
            }
        });
        findViewById(R.id.imageSecretaria).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Para ir a la secretaria
                opcion = 2;
                startActivity(new Intent(OptionActivity.this, MapsRouteB.class));
            }
        });
        findViewById(R.id.imageAulas).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Para ir a las aulas
                opcion = 3;
                startActivity(new Intent(OptionActivity.this, MapsRouteB.class));
            }
        });
        findViewById(R.id.imageResidencia).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Para ir al colegio mayor
                opcion = 4;
                startActivity(new Intent(OptionActivity.this, MapsRouteB.class));
            }
        });
    }
    public static Location obtenerLocaGPS() {
        return localizacion;
    }
    public static int obtenerOpcion() {
        return opcion;
    }
}
```

Como aspectos importantes a destacar de la definición de esta clase Java, es que cuando seleccionamos cualquiera de las 4 opciones de cálculo de ruta le hemos asignado un número que es almacenado en una variable.

Esto se ha hecho para poder facilitarnos la reutilización de las clases y métodos implicados en el cálculo de ruta. Desde la actividad MapsRouteB recuperamos el número almacenado en la variable y mediante una estructura de datos condicional se puede asignar según el valor de la variable opción y asignar un determinado valor a la posición destino.

5.3.4. MapsRouteB.java

En esta actividad es donde se realiza el cálculo de la ruta entre la ubicación actual del usuario y cualquiera de los 4 puntos predeterminados en la app. En esta clase también se hacen llamadas a la clase GetDirectionsAsyncTask.java que permite ir calculando la ruta en segundo plano. El funcionamiento preciso de la misma se detallará en su momento.




```
package es.uc3m.buddyuc3m;

import android.content.Intent;
import android.graphics.Color;
import android.graphics.Point;
import android.location.Location;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import android.view.Display;
import android.view.View;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.LatLngBounds;
import com.google.android.gms.maps.model.PolylineOptions;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
```

```
public class MapsRouteB extends FragmentActivity implements OnMapReadyCallback {
    public Location m_DeviceLocation = OptionActivity.obtenerLocaGPS();
    double latitud = m_DeviceLocation.getLatitude();
    double longitud = m_DeviceLocation.getLongitude();
    public static int eleccion;
    private int anchura, altura;
    private LatLngBounds latlngBounds;
    MapFragment mapFragment;
    GoogleMap mapa;
    LatLng posicion = new LatLng(latitud, longitud);
    //Destinos
    LatLng destino;
    //Opcion 1:
    LatLng biblioteca = new LatLng(40.542441, -4.012414);
    //Opcion 2:
    LatLng secretaria = new LatLng(40.542952, -4.012303);
    //Opcion 3:
    LatLng aulas = new LatLng(40.543515, -4.013181);
    //Opcion 4:
    LatLng colegio = new LatLng(40.545578, -4.012233);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps_route_b);
        obtenerDimensionesPantalla();
        eleccion = OptionActivity.obtenerOpcion();
        fijarDestino(eleccion);
        inicializaMapaSiNecesario();
        findViewById(R.id.button_inmersivo).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(MapsRouteB.this, MapsActivityGPS.class));
            }
        });
        findViewById(R.id.button_navegacion).setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                startActivity(new Intent(MapsRouteB.this, MapsActivity.class));
            }
        });
    }
    //Inicializa el mapa si se encuentra a null:
    private void inicializaMapaSiNecesario() {
        // Verifica que no se ha instanciado el mapa:
        mapFragment = (MapFragment) getFragmentManager().findFragmentById(R.id.map);
        if (mapFragment == null){
            return;
        }
        mapa = mapFragment.getMap();
        onMapReady(mapa);
    }
    @Override
    public void onMapReady(GoogleMap googleMap) {
        findDirections(latitud, longitud, destino.latitude, destino.longitude,
GMapV2Direction.MODE_WALKING);
    }
    @Override
    protected void onResume() {
        super.onResume();
        inicializaMapaSiNecesario();
        latlngBounds = createLatLngBoundsObject(posicion, destino);
        mapa.moveCamera(CameraUpdateFactory.newLatLngBounds(latlngBounds, anchura, altura,
150));
    }
    @Override
    protected void onPause() {
        super.onPause();
    }
    //Con este metodo se fija la posicion de destino a mostrar la ruta:
    public void fijarDestino(int eleccion){
        if(eleccion==1){
```

```
            destino = new LatLng(biblioteca.latitude,biblioteca.longitude);
        }else if (eleccion==2){
            destino = new LatLng(secretaria.latitude,secretaria.longitude);
        }else if (eleccion==3){
            destino = new LatLng(aulas.latitude,aulas.longitude);
        }else if (eleccion==4){
            destino = new LatLng(colegio.latitude,colegio.longitude);
        }else {

        }
    }
    //Este método se emplea para fijar la configuración de posición origen y destino y como
se va a llegar hasta alli (caminando, en coche, etc.)
    public void findDirections(double fromPositionDoubleLat, double fromPositionDoubleLong,
double toPositionDoubleLat, double toPositionDoubleLong, String mode)
    {
        Map<String, String> map = new HashMap<String, String>();
        map.put(GetDirectionsAsyncTask.USER_CURRENT_LAT,
String.valueOf(fromPositionDoubleLat));
        map.put(GetDirectionsAsyncTask.USER_CURRENT_LONG,
String.valueOf(fromPositionDoubleLong));
        map.put(GetDirectionsAsyncTask.DESTINATION_LAT,
String.valueOf(toPositionDoubleLat));
        map.put(GetDirectionsAsyncTask.DESTINATION_LONG,
String.valueOf(toPositionDoubleLong));
        map.put(GetDirectionsAsyncTask.DIRECTIONS_MODE, mode);
        //Se crea el objeto asincrono para poder realizar el calculo de direcciones en
segundo plano
        GetDirectionsAsyncTask asyncTask = new GetDirectionsAsyncTask(this);
        asyncTask.execute(map);
    }
    //Una vez obtenida la lista de puntos de la ruta a dibujar, este método la dibuja sobre
nuestro mapa y lo presenta centrado.
    Public void handleGetDirectionsResult(ArrayList directionPoints)
    {
        // Verifica que no se ha instanciado el mapa:
        mapFragment = (MapFragment) getFragmentManager().findFragmentById(R.id.map);
        //Para poder dibujar el fragment debe de existir.
        if (mapFragment == null){
            return;
        }
        //obtenemos el mapa sobre el cual dibujaremos
        mapa = mapFragment.getMap();
        PolylineOptions rectLine = new PolylineOptions().width(3).color(Color.BLUE);
        //Se construye iterativamente en azul la ruta hasta que hayamos leido todas las
posiciones de la lista
        for(int i = 0 ; i < directionPoints.size() ; i++)
        {
            rectLine.add((LatLng) directionPoints.get(i));
        }
        mapa.addPolyline(rectLine);
        //Creamos un objeto latlngBounds que nos permite crear un recuadro donde centrar el
mapa entre origen y destino
        latlngBounds = createLatLngBoundsObject(posicion, destino);
        //Centramos la camara
        mapa.moveCamera(CameraUpdateFactory.newLatLngBounds(latlngBounds, anchura, altura,
150));
        //Habilitamos ciertas opciones de la interfaz de Google Maps
        mapa.setMyLocationEnabled(true);
        mapa.setBuildingsEnabled(true);
        mapa.setIndoorEnabled(true);
    }
    //Creamos un rectangulo alrededor de la ruta
    private LatLngBounds createLatLngBoundsObject(LatLng primeraLocalizacion, LatLng
segundaLocalizacion)
    {
        if (primeraLocalizacion != null && segundaLocalizacion != null)
        {
            LatLngBounds.Builder builder = new LatLngBounds.Builder();
            builder.include(primeraLocalizacion).include(segundaLocalizacion);

            return builder.build();
        }
        return null;
```

```
    }
    //Para llevar a cabo una representacion precisa según el tamaño de la pantalla del
dispositivo obtenemos sus dimensiones.
    private void obtenerDimensionesPantalla()
    {
        Point punto = new Point(0,0);
        Display display = getWindowManager().getDefaultDisplay();
        display.getSize(punto);
        anchura = punto.x;
        altura = punto.y;
    }
    //Metodo para poder pasar la variable elección a través de las clases.
    public static int obtenerEleccion() {
        return eleccion;
    }
}
```

Como se habrá podido apreciar al comienzo de la clase se importan numerosas clases. Estos es debido a que se utilizan Mapas de Google (objeto GoogleMap), se dibuja sobre un mapa, se utilizan servicios de ubicación, arrays, etc.

Esta clase hereda de la clase FragmentActivity, esto es así porque para poder mostrar un mapa embebido dentro de la aplicación y poder crear una interfaz que no sólo el mapa en la pantalla, una de las partes de la layout tiene que ser un Fragment o fragmento.

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

Asimismo podría decirse que un Fragment no puede considerarse ni un control ni un contenedor, aunque se parecería más a lo segundo.

Nuestra clase en cuestión implementa métodos de la clase OnMapReadyCallback.java ya que nos permite tener un control más preciso del mapa y sobretodo tener un control automatizado del mismo ya implementado en Android.

Lo primero que se hace en la clase es obtener la ubicación actual del usuario e informar las variables correspondientes y dar el valor de las coordenadas geográficas a los 4 puntos predeterminados con los que se trabaja.

A continuación se procederá a explicar los métodos más representativos de esta clase:

- **Método obtenerDimensionesPantalla:**

```
//Para llevar a cabo una representación precisa según el tamaño de la pantalla del
dispositivo obtenemos sus dimensiones.
private void obtenerDimensionesPantalla()
{
    Point punto = new Point(0,0);
    Display display = getWindowManager().getDefaultDisplay();
    display.getSize(punto);
    anchura = punto.x;
    altura = punto.y;
}
```

El cálculo de ruta de Buddy UC3M está preparado para que este pueda ser llevado a cabo en cualquier tamaño de pantalla. Para ello se implementó este método que accediendo a las valores del sistema obtiene la anchura y altura actual de la pantalla del dispositivo.

- **Método fijarDestino:**

```
//Con este metodo se fija la posicion de destino a mostrar la ruta:
public void fijarDestino(int eleccion){
    if(eleccion==1){
        destino = new LatLng(biblioteca.latitude,biblioteca.longitude);
    }else if (eleccion==2){
        destino = new LatLng(secretaria.latitude,secretaria.longitude);
    }else if (eleccion==3){
        destino = new LatLng(aulas.latitude,aulas.longitude);
    }else if (eleccion==4){
        destino = new LatLng(colegio.latitude,colegio.longitude);
    }else {

    }
}
```

En este método tras obtener el valor de la opción que el usuario eligió en la pantalla anterior, se procede a asignar un valor a la variable destino que se utilizará para el cálculo de ruta.

- **Método inicializaMapaSiNecesario:**

```
//Inicializa el mapa si se encuentra a null:
private void inicializaMapaSiNecesario() {
    // Verifica que no se ha instanciado el mapa:
    mapFragment = (MapFragment)
getFragmentManager().findFragmentById(R.id.map);
    if (mapFragment == null){
        return;
    }
    mapa = mapFragment.getMap();
    onMapReady(mapa);
}
```

Esta implementación nos asegura que antes de realizar cualquier operación con el fragmento mapa, éste exista. En caso contrario se recupera y se indica que se encuentra listo para trabajar con él.

Este método invoca a la función onMapReady que lo que hace es a su vez invocar al método findDirections que es el que comienza con el cálculo de la ruta.

- **Método onMapReady:**

```
@Override
public void onMapReady(GoogleMap googleMap) {
    findDirections(latitud, longitud, destino.latitude, destino.longitude,
GMapV2Direction.MODE_WALKING);
}
```

Como antes se dijo hay ciertos métodos que se sobrescriben, este es uno de ellos. Cuando el mapa se encuentra listo se invoca al método que procede al cálculo de direcciones findDirections. Como se ha considerado que lo más realista es que el usuario se mueva a pie por el Campus, se ha indicado que el modo del cálculo de la ruta sea como si este fuera a recorrerla caminando. Para ello se hace una llamada a la clase GMapV2Direction a la constante MODE_WALKING.

- **Método findDirections:**

```
//Este método se emplea para fijar la configuración de posición origen y destino y como se
va a llegar hasta allí (caminando, en coche, etc.)
public void findDirections(double fromPositionDoubleLat, double fromPositionDoubleLong,
double toPositionDoubleLat, double toPositionDoubleLong, String mode)
{
    Map<String, String> map = new HashMap<String, String>();
    map.put(GetDirectionsAsyncTask.USER_CURRENT_LAT,
String.valueOf(fromPositionDoubleLat));
    map.put(GetDirectionsAsyncTask.USER_CURRENT_LONG,
String.valueOf(fromPositionDoubleLong));
    map.put(GetDirectionsAsyncTask.DESTINATION_LAT, String.valueOf(toPositionDoubleLat));
    map.put(GetDirectionsAsyncTask.DESTINATION_LONG, String.valueOf(toPositionDoubleLong));
    map.put(GetDirectionsAsyncTask.DIRECTIONS_MODE, mode);
    //Se crea el objeto asincrono para poder realizar el calculo de direcciones en segundo
plano
    GetDirectionsAsyncTask asyncTask = new GetDirectionsAsyncTask(this);
    asyncTask.execute(map);
}
```

Este método se encarga de informar las contantes de la clase GetDirectionsAsyncTask, la cual se explicará en su momento.

Es en este momento donde vamos a hacer uso de la multitarea real creando un objeto del tipo GetDirectionsAsyncTask y aplicando su método principal execute con los datos con los que se ha informado la variable map.

A continuación se procede a explicar un método que no se usa en esta clase pero se referencia en la clase GetDirectionsAsyncTask que es la encargada de crear la multitarea real en nuestra app:

- **Método handleGetDirectionsResult:**

```
//Una vez obtenida la lista de puntos de la ruta a dibujar, este método la dibuja sobre
nuestro mapa y lo presenta centrado.
public void handleGetDirectionsResult(ArrayList directionPoints)
{
    // Verifica que no se ha instanciado el mapa:
    mapFragment = (MapFragment) getFragmentManager().findFragmentById(R.id.map);
    //Para poder dibujar el fragment debe de existir.
    if (mapFragment == null){
        return;
    }
    //obtenemos el mapa sobre el cual dibujaremos
    mapa = mapFragment.getMap();
    PolylineOptions rectLine = new PolylineOptions().width(3).color(Color.BLUE);
    //Se construye iterativamente en azul la ruta hasta que hayamos leido todas las
posiciones de la lista
    for(int i = 0 ; i < directionPoints.size() ; i++)
    {
        rectLine.add((LatLng) directionPoints.get(i));
    }
    mapa.addPolyline(rectLine);
    //Creamos un objeto latlngBounds que nos permite crear un recuadro donde centrar el
mapa entre origen y destino
    latlngBounds = createLatLngBoundsObject(posicion, destino);
    //Centramos la camara
    mapa.moveCamera(CameraUpdateFactory.newLatLngBounds(latlngBounds, anchura, altura,
150));
    //Habilitamos ciertas opciones de la interfaz de Google Maps
    mapa.setMyLocationEnabled(true);
    mapa.setBuildingsEnabled(true);
    mapa.setIndoorEnabled(true);
}
```

Como se ha comentado, este método es invocado en la clase de la multitarea. Se ejecuta justo cuando el cálculo de ruta ha finalizado, procediendo a dibujar sobre el Mapa embebido en el Fragment la ruta obtenida. Su invocación se hace sobrescribiendo el método onPostExecute de la clase GetDirectionsAsyncTask.

El método handleGetDirectionsResult recibe como parámetros de entrada una lista de puntos de dirección. Cada punto está formado por una latitud y una longitud. Asimismo es importante reseñar que la unión de estos puntos uno tras otro, como si trazaramos una línea, sobre un mapa da lugar a la ruta buscada.

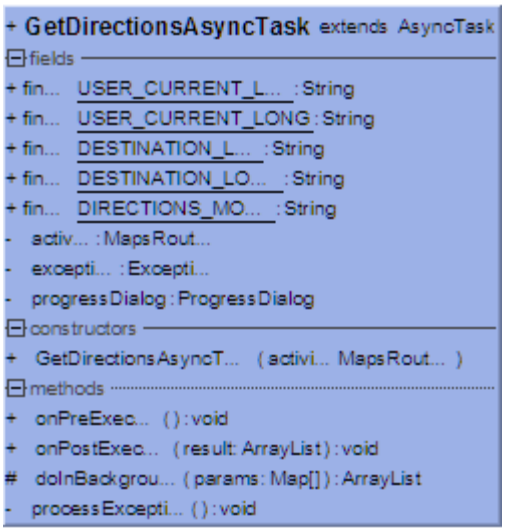
Una vez dibujada la ruta sobre el mapa, éste método centra el mapa con la ruta dibujada en la pantalla y se procede a habilitar ciertas opciones de la interfaz de Google Maps para que el usuario final pueda interaccionar con el mapa.

Y ya para terminar con la explicación de esta clase, se puede observar que se ha creado dos opciones mediante dos botones. Una de ellas permite al usuario ver su posición actual con imágenes reales en 360º y la otra hacer una llamada a la aplicación Google Maps procediendo a la navegación en coche desde la ubicación actual del usuario hasta el punto destino seleccionado.

5.3.5. GetDirectionsAsyncTask.java

Esta clase es la encargada de generar la multitarea real en la app. Para ello hereda de la clase AsyncTask.

El motivo de heredar de esta clase, es que AsyncTask ya que nos proporciona los mecanismos para un control adecuado de la multitarea. Se podría haber hecho con la clase Thread con los métodos synchronized y ser nosotros quienes controlemos todo. Pero gracias a la clase especial AsyncTask, nos facilitará mucho la vida y será mucho más óptimo.




```
package es.uc3m.buddyuc3m;

import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.widget.Toast;
import com.google.android.gms.maps.model.LatLng;
import org.w3c.dom.Document;
import java.util.ArrayList;
import java.util.Map;

public class GetDirectionsAsyncTask extends AsyncTask<Map<String, String>, Object,
ArrayList>
{
    //Definición de constantes de la clase.
    public static final String USER_CURRENT_LAT = "user_current_lat";
    public static final String USER_CURRENT_LONG = "user_current_long";
    public static final String DESTINATION_LAT = "destination_lat";
    public static final String DESTINATION_LONG = "destination_long";
    public static final String DIRECTIONS_MODE = "directions_mode";
    private MapsRouteB activity;
    private Exception exception;
    private ProgressDialog progressDialog;

    //Constructor de la clase a la cual le pasamos la actividad que le precede.
```

```
    public GetDirectionsAsyncTask(MapsRouteB activity)
    {
        super();
        this.activity = activity;
    }
    //Mientras se está calculando la ruta mostramos el siguiente mensaje de progreso.
    public void onPreExecute()
    {
        progressDialog = new ProgressDialog(activity);
        progressDialog.setMessage("Calculando direcciones");
        progressDialog.show();
    }
    //Una vez que hemos acabado el cálculo de la ruta, llamamos al método de la clase
MapsRouteB para dibujarlo en el mapa.
    @Override
    public void onPostExecute(ArrayList result)
    {
        progressDialog.dismiss();
        if (exception == null)
        {
            activity.handleGetDirectionsResult(result);
        }
        else
        {
            processException();
        }
    }

    @Override
    //En este método se procede a obtener la lista de puntos de direcciones gracias al
servicio de Google Directions.
    protected ArrayList doInBackground(Map<String, String>... params)
    {
        Map<String, String> paramMap = params[0];
        try
        {
            LatLng fromPosition = new LatLng(Double.valueOf(paramMap.get(USER_CURRENT_LAT))
, Double.valueOf(paramMap.get(USER_CURRENT_LONG)));
            LatLng toPosition = new LatLng(Double.valueOf(paramMap.get(DESTINATION_LAT)) ,
Double.valueOf(paramMap.get(DESTINATION_LONG)));
            GMapV2Direction md = new GMapV2Direction();
            Document doc = md.getDocument(fromPosition, toPosition,
paramMap.get(DIRECTIONS_MODE));
            ArrayList directionPoints = md.getDirection(doc);
            return directionPoints;
        }
        catch (Exception e)
        {
            exception = e;
            return null;
        }
    }
    //En caso de que haya problemas con el proceso.
    private void processException()
    {
        Toast.makeText(activity, activity.getString(R.string.error_when_retrieving_data),
Toast.LENGTH_LONG).show();
    }
}
```

Al comienzo de la clase se puede observar la definición de constantes. Estos son los métodos que se han implementado en la clase:

- **Método GetDirectionsAsyncTask:**

```
//Constructor de la clase a la cual le pasamos la actividad que le precede.
public GetDirectionsAsyncTask(MapsRouteB activity)
{
    super();
```

```
        this.activity = activity;
    }
```

Constructor de la clase a la cual le pasamos la actividad que le precede, MapsRouteB.

- **Método onPreExecute:**

```
    //Mientras se está calculando la ruta mostramos el siguiente mensaje de progreso.
    public void onPreExecute()
    {
        progressDialog = new ProgressDialog(activity);
        progressDialog.setMessage("Calculando direcciones");
        progressDialog.show();
    }
```

Este método se utiliza antes de llevar la ejecución principal de la actividad. En nuestro caso mostramos un mensaje de progreso mientras se procede al cálculo de la ruta.

- **Método onPostExecute:**

```
    //Una vez que hemos acabado el cálculo de la ruta, llamamos al método de la clase
    MapsRouteB para dibujarlo en el mapa.
    @Override
    public void onPostExecute(ArrayList result)
    {
        progressDialog.dismiss();
        if (exception == null)
        {
            activity.handleGetDirectionsResult(result);
        }
        else
        {
            processException();
        }
    }
```

Este método se emplea cuando ya se ha calculado la ruta y por tanto poseemos la lista de puntos formados cada uno con la tupla latitud, longitud. Se invoca al método handleGetDirectionsResult de la clase MapsRouteB que ya se explicó cuando se habló de esta clase anteriormente.

- **Método doInBackground:**

```
    @Override
    //En este método se procede a obtener la lista de puntos de direcciones gracias al servicio
    de Google Directions.
    protected ArrayList doInBackground(Map<String, String>... params)
    {
        Map<String, String> paramMap = params[0];
        try
        {
            LatLng fromPosition = new LatLng(Double.valueOf(paramMap.get(USER_CURRENT_LAT)) ,
    Double.valueOf(paramMap.get(USER_CURRENT_LONG)));
            LatLng toPosition = new LatLng(Double.valueOf(paramMap.get(DESTINATION_LAT)) ,
    Double.valueOf(paramMap.get(DESTINATION_LONG)));
            GMapV2Direction md = new GMapV2Direction();
            Document doc = md.getDocument(fromPosition, toPosition,
    paramMap.get(DIRECTIONS_MODE));
```

```
            ArrayList directionPoints = md.getDirection(doc);
            return directionPoints;
        }
        catch (Exception e)
        {
            exception = e;
            return null;
        }
    }
```

Se podría decir que este es el método más representativo de la clase. Es el método que realiza el cálculo de la lista de lista de puntos de nuestra ruta.

Como su nombre indica hace uso de la multitarea al ejecutarse en segundo plano. De los valores con los que se informaron las constantes de esta clase se definen dos objetos LatLng donde se almacena la posición origen y destino respectivamente, un objeto del tipo GMapV2Direction, un objeto de tipo Document donde se almacenará la ruta obtenida del objeto anteriormente definido del tipo GMapV2Direction y por último una lista donde se almacena la lista de puntos de dirección calculada mediante la invocación del método getDirection de la clase GMapV2Direction al que se le pasa como parámetro el objeto Documento anteriormente obtenido.

Si ocurriera una excepción se devuelve un objeto nulo.

- **Método processException:**

```
//En caso de que haya problemas con el proceso.
private void processException()
{
    Toast.makeText(activity, activity.getString(R.string.error_when_retrieving_data),
Toast.LENGTH_LONG).show();
}
```

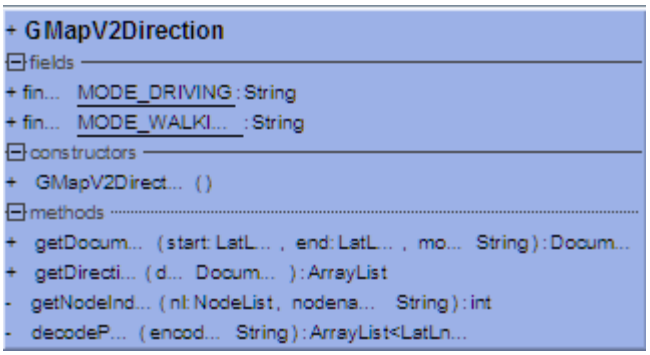
Debido a que como se ha podido apreciar antes en el método anterior se pueden producir excepción, se ha implementado este método que en caso de que ocurra algún error al recuperar los datos se muestra una notificación toast al usuario.

5.3.6. GMapV2Direction.java

Tal y como ya se ha hablado en capítulos anteriores, Google no permite la navegación directamente sobre Google Maps. Así que la opción fue encontrar la manera de generar una polilínea donde el comienzo de la misma fuera la posición origen y el final la posición destino del usuario.

Asimismo cada vértice de la polilínea sería una coordenada geográfica de la ruta a seguir, donde finalmente una vez calculada los valores de todos los puntos que conforman la polilínea se dibujaría sobre el mapa, obteniendo la ruta deseada.

Para ello se ha usado la clase GMapV2Direction, esta clase es la responsable de realizar una petición a la API de Google Directions y obtener las instrucciones de navegación entre dos puntos LatLng.




```
package es.uc3m.buddyuc3m;

import com.google.android.gms.maps.model.LatLng;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.protocol.BasicHttpContext;
import org.apache.http.protocol.HttpContext;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import java.io.InputStream;
import java.util.ArrayList;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

/**
 * Esta clase es la encargada de acceder al servicio de direcciones para poder calcular la
ruta en un mapa
 */
public class GMapV2Direction {
    public final static String MODE_DRIVING = "driving";
    //Finalmente se opta por el cálculo de la ruta caminando.
    public final static String MODE_WALKING = "walking";
```

```
    public GMapV2Direction() { }
    //Este método obtiene un documento donde se detalla toda la ruta desde nuestro origen
al destino.
    public Document getDocument(LatLng start, LatLng end, String mode) {
        String url = "http://maps.googleapis.com/maps/api/directions/xml?"
                + "origin=" + start.latitude + "," + start.longitude
                + "&destination=" + end.latitude + "," + end.longitude
                + "&sensor=false&units=metric&mode=" + mode;

        try {
            HttpClient httpClient = new DefaultHttpClient();
            HttpContext localContext = new BasicHttpContext();
            HttpPost httpPost = new HttpPost(url);
            HttpResponse response = httpClient.execute(httpPost, localContext);
            InputStream in = response.getEntity().getContent();
            DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
            Document doc = builder.parse(in);
            return doc;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    //Saca la lista de geopuntos recuperándolos del documento obtenido con la ruta a través
de los servicios web.
    public ArrayList getDirection (Document doc) {
        NodeList nl1, nl2, nl3;
        ArrayList<LatLng> listGeopoints = new ArrayList();
        nl1 = doc.getElementsByTagName("step");
        if (nl1.getLength() > 0) {
            for (int i = 0; i < nl1.getLength(); i++) {
                Node node1 = nl1.item(i);
                nl2 = node1.getChildNodes();

                Node locationNode = nl2.item(getNodeIndex(nl2, "start_location"));
                nl3 = locationNode.getChildNodes();
                Node latNode = nl3.item(getNodeIndex(nl3, "lat"));
                double lat = Double.parseDouble(latNode.getTextContent());
                Node lngNode = nl3.item(getNodeIndex(nl3, "lng"));
                double lng = Double.parseDouble(lngNode.getTextContent());
                listGeopoints.add(new LatLng(lat, lng));

                locationNode = nl2.item(getNodeIndex(nl2, "polyline"));
                nl3 = locationNode.getChildNodes();
                latNode = nl3.item(getNodeIndex(nl3, "points"));
                ArrayList<LatLng> arr = decodePoly(latNode.getTextContent());
                for(int j = 0 ; j < arr.size() ; j++) {
                    listGeopoints.add(new LatLng(arr.get(j).latitude,
arr.get(j).longitude));
                }

                locationNode = nl2.item(getNodeIndex(nl2, "end_location"));
                nl3 = locationNode.getChildNodes();
                latNode = nl3.item(getNodeIndex(nl3, "lat"));
                lat = Double.parseDouble(latNode.getTextContent());
                lngNode = nl3.item(getNodeIndex(nl3, "lng"));
                lng = Double.parseDouble(lngNode.getTextContent());
                listGeopoints.add(new LatLng(lat, lng));
            }
        }

        return listGeopoints;
    }

    private int getNodeIndex(NodeList nl, String nodename) {
        for(int i = 0 ; i < nl.getLength() ; i++) {
            if(nl.item(i).getNodeName().equals(nodename))
                return i;
        }
        return -1;
    }
```

```
    private ArrayList<LatLng> decodePoly(String encoded) {
        ArrayList<LatLng> poly = new ArrayList();
        int index = 0, len = encoded.length();
        int lat = 0, lng = 0;
        while (index < len) {
            int b, shift = 0, result = 0;
            do {
                b = encoded.charAt(index++) - 63;
                result |= (b & 0x1f) << shift;                  shift += 5;             }
while (b >= 0x20);
            int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
            lat += dlat;
            shift = 0;
            result = 0;
            do {
                b = encoded.charAt(index++) - 63;
                result |= (b & 0x1f) << shift;                  shift += 5;             }
while (b >= 0x20);
            int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
            lng += dlng;

            LatLng position = new LatLng((double) lat / 1E5, (double) lng / 1E5);
            poly.add(position);
        }
        return poly;
    }
}
```

A continuación se procede a explicar los métodos más relevantes de esta clase.

- **Método getDocument:**

```
//Este método obtiene un documento donde se detalla toda la ruta desde nuestro origen al
destino.
public Document getDocument(LatLng start, LatLng end, String mode) {
    String url = "http://maps.googleapis.com/maps/api/directions/xml?"
            + "origin=" + start.latitude + "," + start.longitude
            + "&destination=" + end.latitude + "," + end.longitude
            + "&sensor=false&units=metric&mode=" + mode;

    try {
        HttpClient httpClient = new DefaultHttpClient();
        HttpContext localContext = new BasicHttpContext();
        HttpPost httpPost = new HttpPost(url);
        HttpResponse response = httpClient.execute(httpPost, localContext);
        InputStream in = response.getEntity().getContent();
        DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document doc = builder.parse(in);
        return doc;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

Este método hace uso de los servicios web de direcciones de Google, que es conocido como Google Directions.

Lo primero que hacemos es construir una dirección url en el formato adecuado para que el servicio lo reconozca. Para ello se añade de la manera adecuada latitud y longitud origen, latitud y longitud destino, y por último como se va a recorrer la ruta.

En este caso se ha decidido que sea caminando. Ya que se entiende que al ser los puntos destino lugares peatonales del Campus de Colmenarejo no podrá acceder a ellos por otro método de transporte que no sea caminando.

Una vez construida la url, se emplean los servicios web para ejecutar la URL y la salida que devuelve Google Directions almacenarla en un objeto del tipo Documento.

En este objeto Documento se encontrarían por tanto todas las instrucciones de navegación detalladas entre nuestros puntos origen y destino

- **Método getDirection:**

```
//Saca la lista de geopuntos recuperándolos del documento obtenido con la ruta a través de
los servicios web.
public ArrayList getDirection (Document doc) {
    NodeList nl1, nl2, nl3;
    ArrayList<LatLng> listGeopoints = new ArrayList();
    nl1 = doc.getElementsByTagName("step");
    if (nl1.getLength() > 0) {
        for (int i = 0; i < nl1.getLength(); i++) {
            Node node1 = nl1.item(i);
            nl2 = node1.getChildNodes();

            Node locationNode = nl2.item(getNodeIndex(nl2, "start_location"));
            nl3 = locationNode.getChildNodes();
            Node latNode = nl3.item(getNodeIndex(nl3, "lat"));
            double lat = Double.parseDouble(latNode.getTextContent());
            Node lngNode = nl3.item(getNodeIndex(nl3, "lng"));
            double lng = Double.parseDouble(lngNode.getTextContent());
            listGeopoints.add(new LatLng(lat, lng));

            locationNode = nl2.item(getNodeIndex(nl2, "polyline"));
            nl3 = locationNode.getChildNodes();
            latNode = nl3.item(getNodeIndex(nl3, "points"));
            ArrayList<LatLng> arr = decodePoly(latNode.getTextContent());
            for(int j = 0 ; j < arr.size() ; j++) {
                listGeopoints.add(new LatLng(arr.get(j).latitude, arr.get(j).longitude));
            }

            locationNode = nl2.item(getNodeIndex(nl2, "end_location"));
            nl3 = locationNode.getChildNodes();
            latNode = nl3.item(getNodeIndex(nl3, "lat"));
            lat = Double.parseDouble(latNode.getTextContent());
            lngNode = nl3.item(getNodeIndex(nl3, "lng"));
            lng = Double.parseDouble(lngNode.getTextContent());
            listGeopoints.add(new LatLng(lat, lng));
        }
    }

    return listGeopoints;
}
```

Y finalmente este es el método que se encarga de parsear el objeto Documento obtenido con las instrucciones de navegación, obteniendo una lista de geopuntos del tipo LatLng.

El método básicamente va recorriendo el documento, para ello lo que obtiene es una lista de nodos de longitud igual al número de pasos o instrucciones de navegación que tiene el documento.

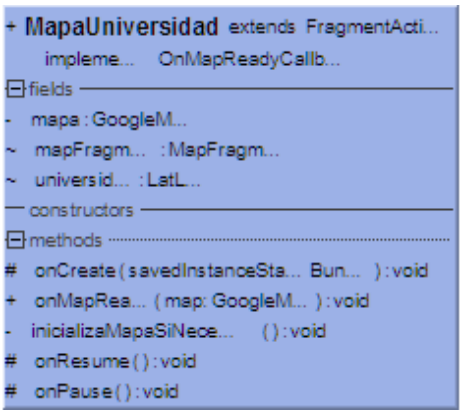
Posteriormente se va recorriendo los pasos y va parseando por cada nodo encontrado su latitud y longitud a una variable del tipo double.

A continuación por cada conjunto de variables double informadas, se conforma un objeto LatLng que añade a la lista de geopuntos.

Finalmente cuando se ha llegado al final del documento, se retorna la lista con los geopuntos que conforman la ruta.

5.3.7. MapaUniversidad.java

Tras explicar todas las actividades implicadas en la opción del cálculo de ruta, se procede a explicar la clase java responsable cuando el usuario selecciona la opción de mostrar el plano del Campus de Colmenarejo.




```
package es.uc3m.buddyuc3m;

import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.widget.Toast;
import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.LatLng;


public class MapaUniversidad extends FragmentActivity implements OnMapReadyCallback {
    private GoogleMap mapa;
    MapFragment mapFragment;
    LatLng universidad;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_universidad);
        Toast.makeText(getApplicationContext(), "Campus de Colmenarejo",
Toast.LENGTH_LONG).show();
        inicializaMapaSiNecesario();
    }

    @Override
    public void onMapReady(GoogleMap map) {
        universidad = new LatLng(40.543325, -4.012606);
        CameraPosition camPos = new CameraPosition.Builder()
                .target(universidad)
                .zoom(17)
                .bearing(300)
                .build();
        CameraUpdate camUpd = CameraUpdateFactory.newCameraPosition(camPos);
        mapa.moveCamera(camUpd);
        mapa.animateCamera(camUpd);
        mapa.getUiSettings().setIndoorLevelPickerEnabled(true);
        mapa.setMyLocationEnabled(true);
        mapa.setBuildingsEnabled(true);
```

```
        mapa.setIndoorEnabled(true);
    }
    //Inicializa el mapa si se encuentra a null:
    private void inicializaMapaSiNecesario() {
        // Verifica que no se ha instanciado el mapa:
        mapFragment = (MapFragment) getFragmentManager().findFragmentById(R.id.map);
        if (mapFragment == null){
            return;
        }
        mapa = mapFragment.getMap();
        onMapReady(mapa);
    }
    @Override
    protected void onResume() {
        super.onResume();
        inicializaMapaSiNecesario();
    }
    @Override
    protected void onPause() {
        super.onPause();
    }
}
```

Tras buscar cual era las coordenadas geográficas más adecuadas para mostrar un mapa centrado del Campus de Colmenarejo, finalmente se optó por las siguientes:

- **Latitud:** 40.543325
- **Longitud:** -4.012606

Esta clase hereda de la clase FragmentActivity e implementa la clase OnMapReadyCallback. Gracias a la utilización de la clase OnMapReadyCallback nos podemos despreocupar de ciertos aspectos del mapa y sobrescribir los métodos que se consideren oportunos para la funcionalidad buscada.

Para mostrar cualquier mapa Google en la app, es necesario la definición de un objeto MapFragment que contendrá el mapa a mostrar.

Lo primero que tenemos que hacer es inicializar dicho objeto, sino su valor será nulo y obtendremos una excepción al ejecutar esta clase.

- **Método onMapReady:**

```
@Override
public void onMapReady(GoogleMap map) {
    universidad = new LatLng(40.543325, -4.012606);
    CameraPosition camPos = new CameraPosition.Builder()
            .target(universidad)
            .zoom(17)
            .bearing(300)
            .build();
    CameraUpdate camUpd = CameraUpdateFactory.newCameraPosition(camPos);
    mapa.moveCamera(camUpd);
    mapa.animateCamera(camUpd);
    mapa.getUiSettings().setIndoorLevelPickerEnabled(true);
    mapa.setMyLocationEnabled(true);
    mapa.setBuildingsEnabled(true);
    mapa.setIndoorEnabled(true);
}
```

En el método onMapReady es donde se realizan las definiciones del mapa. Lo primero que hacemos es crear el objeto universidad del tipo LatLng que contendrá los datos de latitud y longitud anteriormente citados.

A continuación se definen las propiedades de la cámara que será la que hará foco sobre el mapa. Se indica para ello el target o punto donde se centrará el mapa, el nivel de zoom y la orientación. Con estos atributos se crea la cámara.

Posteriormente se crea un objeto de tipo CameraUpdate al que se le asigna la posición anteriormente definida y se indica que el mapa aplique esa cámara.

Para dotar de más funcionalidad el mapa se han habilitado las siguientes propiedades o atributos de un mapa Google.

- Se habilita la vista indoor de los edificios. De modo que el usuario puede observar por planta los edificios que tengan mapas internos definidos. En nuestro caso tanto la biblioteca Menéndez Pidal y el Edificio Miguel de Unamuno disponen de esta característica.

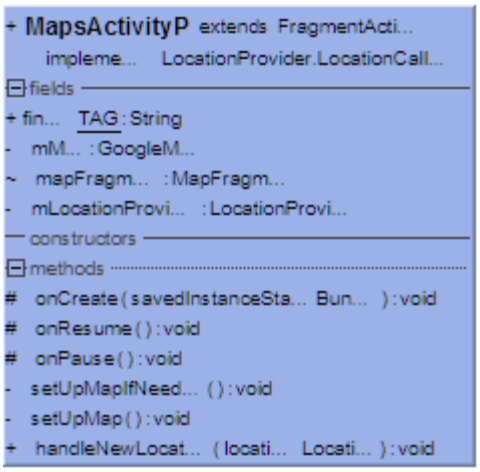
- Asimismo para que la funcionalidad anterior pueda ser usada se ha habilitado el selector de niveles o plantas aplicable sobre un edificio que cumpla con las características anteriormente citadas.

- Se habilita que el usuario pueda pulsar sobre el icono ubicación que se encuentra situado en la esquina superior derecha. Al pulsar sobre el mismo, el mapa se centra en la ubicación del usuario. Se considera que puede ser útil cuando el usuario quiera ver de una forma rápida en que parte del Campus de Colmenarejo (o cualquier otro lugar) se encuentra.

- Y por último, se ha habilitado que se muestre los edificios en 3D sobre el mapa. No aporta gran valor a la funcionalidad de la app, pero si el usuario sitúa el mapa sobre una zona céntrica y lo inclina podrá ver en tres dimensiones los edificios de su alrededor lo que le otorga una mayor perspectiva.

5.3.8. MapsActivityP.java

Esta actividad se ejecuta cuando el usuario desea mostrar su ubicación en un mapa.




```
package es.uc3m.buddyuc3m;

import android.content.Intent;
import android.location.Location;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.view.View;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.model.LatLng;

public class MapsActivityP extends FragmentActivity implements
LocationProvider.LocationCallback {

    public static final String TAG = MapsActivity.class.getSimpleName();
    private GoogleMap mMap; // Podría tener valor nulo si los servicios APK Google Play no
están disponibles.
    MapFragment mapFragment;
    private LocationProvider mLocationProvider;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps_p);
        setUpMapIfNeeded();
        mLocationProvider = new LocationProvider(this, this);
        findViewById(R.id.button_inmersivo).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(MapsActivityP.this, MapsActivityGPS.class));
            }
        });
    }

    @Override
    protected void onResume() {
        super.onResume();
        setUpMapIfNeeded();
        mLocationProvider.connect();
    }
```

```
    @Override
    protected void onPause() {
        super.onPause();
        mLocationProvider.disconnect();
    }

    //Instancia el mapa si no se ha hecho.
    private void setUpMapIfNeeded() {

        if (mMap == null) {
            // Intenta obtener el mapa de SupportMapFragment.
            mapFragment = (MapFragment) getFragmentManager().findFragmentById(R.id.map);
            mMap = mapFragment.getMap();
            // Verifica si se obtuvo satisfactoriamente el mapa
            if (mMap != null) {
                setUpMap();
            }
        }
    }

    private void setUpMap() {
    }
    //Fija las propiedades del mapa a mostrar.
    public void handleNewLocation(Location location) {
        Log.d(TAG, location.toString());
        double currentLatitude = location.getLatitude();
        double currentLongitude = location.getLongitude();
        LatLng latLng = new LatLng(currentLatitude, currentLongitude);
        mMap.getUiSettings().setIndoorLevelPickerEnabled(true);
        mMap.setMyLocationEnabled(true);
        mMap.setBuildingsEnabled(true);
        mMap.setIndoorEnabled(true);
        mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mMap.animateCamera(CameraUpdateFactory.zoomTo(17.0f));
        mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
    }
}
```

La implementación es muy similar a la clase MapaUniversidad.java, se podría decir que las diferencias fundamentales entre ambas son las siguientes:

- La clase MapsActivityP.java también hereda de la clase FragmentActivity.java pero implementa los métodos de la clase LocationProvider.LocationCallback. LocationProvider es una clase estándar ya definida en Android y hubo que importarla. Gracias a esta definición se posibilita poder calcular la ubicación del usuario sin tener que hacer uso de la que nosotros calculamos al principio (se ha querido hacer de ambos modos a propósito) y lo más importante tener control sobre el comportamiento del objeto mapa con el que se interactúa.

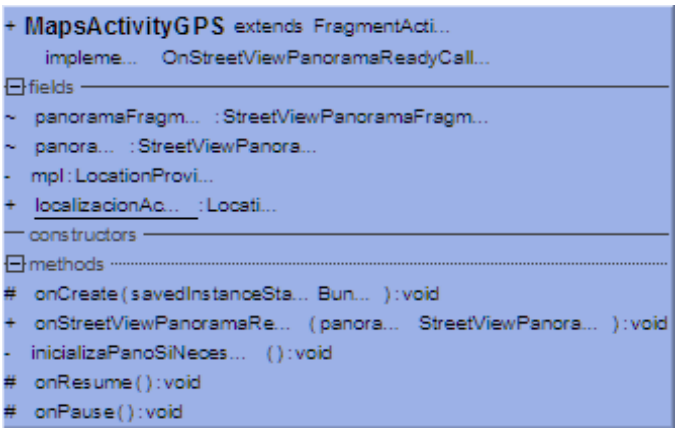
- Las propiedades al mapa que se mostrará son las mismas que para la vista del Campus, salvo que el tipo de mapa elegido será híbrido, es decir, que combina la vista de satélite con el mapa de cartografía estándar.

Tanto cuando se muestra el mapa del Campus como el mapa con nuestra posición actual, es importante recalcar que para los mapas puedan ser mostrados el usuario

tiene que tener los servicios Google Play instalados en su terminal, sino no funcionará y se ofrecerá al usuario su instalación (esto se verifica al comienzo de la app).

5.3.9. MapsActivityGPS.java

En alguna de las opciones de la app aparece un botón con la etiqueta **MODO INMERSIVO**. Las acciones que se realizan cuando el usuario presiona ese botón, es la de mostrar al usuario una vista panorámica de su ubicación con imágenes reales proporcionadas por el servicio Street View.



Se procede por tanto a explicar la implementación de la misma:

```
package es.uc3m.buddyuc3m;

import android.location.Location;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.widget.Toast;
import com.google.android.gms.maps.OnStreetViewPanoramaReadyCallback;
import com.google.android.gms.maps.StreetViewPanorama;
import com.google.android.gms.maps.StreetViewPanoramaFragment;
import com.google.android.gms.maps.model.LatLng;

public class MapsActivityGPS extends FragmentActivity implements
OnStreetViewPanoramaReadyCallback {
    StreetViewPanoramaFragment panoramaFragment;
    StreetViewPanorama panorama;
    //Mi proveedor de localización
    private LocationProvider mpl;
    //Se utilizará la localización heredada
    public static Location localizacionActual = OptionActivity.obtenerLocaGPS();;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps_gps);
        Toast.makeText(getApplicationContext(), "Mira a tu alrededor",
Toast.LENGTH_LONG).show();
        inicializaPanoSiNecesario();
    }

    @Override
```

```
    public void onStreetViewPanoramaReady(StreetViewPanorama panorama) {
        LatLng latLng = new LatLng(localizacionActual.getLatitude(),
localizacionActual.getLongitude());
        panorama.setPosition(latLng);

    }
    //Inicializa el panorama si se encuentra a null:
    private void inicializaPanoSiNecesario() {
        // Verifica que no se ha instanciado el panorama:
        panoramaFragment = (StreetViewPanoramaFragment)
getFragmentManager().findFragmentById(R.id.map);
        if (panoramaFragment == null){
            return;
        }
        panorama = panoramaFragment.getStreetViewPanorama();
        onStreetViewPanoramaReady(panorama);
    }

    @Override
    protected void onResume() {
        super.onResume();
        inicializaPanoSiNecesario();
        //mpl.connect();
    }
    @Override
    protected void onPause() {
        super.onPause();
        //mpl.disconnect();
    }
}
```

Como en todas nuestras actividades que tengan implicado las operaciones sobre un objeto mapa Google, la clase hereda la clase FragmentActivity. Tal y como ocurre con las otras clases donde se mostraba nuestra ubicación en el mapa o bien el mapa del Campus de Colmenarejo se ha utilizado la clase OnMapReadyCallback.

En este caso se trabaja con un nuevo tipo de objeto, el objeto StreetViewPanorama. Es por ello que por que para tener un mayor control sobre el panorama empleamos la clase OnStreetViewPanoramaReadyCallback.

Esta clase permite optimizar y simplificar el uso de panoramas, por lo que se han sobrescrito algunos métodos de la misma para obtener la funcionalidad deseada.

Esta vez sí hemos heredado la ubicación del usuario calculada al inicio de la app, ya que consideramos que no es necesario que esta se vaya refrescando a medida que nos movamos. Se supone que el modo inmersivo es algo puntual, además de que su uso implica un gran consumo de datos móviles al tener que descargarse las imágenes de nuestro entorno según nos movemos.

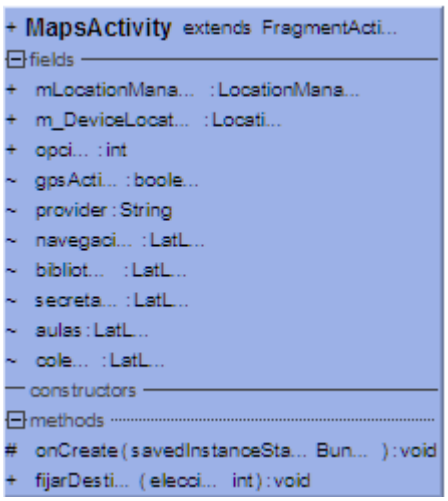
5.3.10. MapsActivity.java

Esta es la última clase que dota de una funcionalidad relevante a nuestra app. Esta clase es usada cuando el usuario presiona el botón **INICIAR NAVEGACION** que se encuentra situado en la pantalla del cálculo de ruta.

Se estuvo bastante tiempo investigando si se podía embeber la navegación dentro de la app, pero parece ser que no es posible según los términos del servicio de Google Maps.

Como no se quería prescindir de poder navegar a cualquiera de los 4 destinos posibles desde nuestra ubicación actual, se ha encontrado la manera de llamar por fuera a la aplicación Google Maps y que pase directamente a la navegación con los parámetros que le hayamos indicado.

En este proyecto se ha tratado de reutilizar código lo máximo posible, es por ello por lo que usando una manera muy similar a lo que se hizo de las opciones en el cálculo de ruta, con una sola actividad es posible navegar a cualquiera de los 4 destinos posibles.



A continuación se procede a explicar lo más relevante de la implementación de esta clase:

```
package es.uc3m.buddyuc3m;

import android.content.Intent;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
```

```
import android.net.Uri;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import android.widget.Toast;
import com.google.android.gms.maps.model.LatLng;


public class MapsActivity extends FragmentActivity {
    public LocationManager mLocationManager;
    public Location m_DeviceLocation = OptionActivity.obtenerLocaGPS();
    public int opcion;
    boolean gpsActivo = false;
    String provider;
    LatLng navegacion;
    //Opcion 1:
    LatLng biblioteca = new LatLng(40.542441, -4.012414);
    //Opcion 2:
    LatLng secretaria = new LatLng(40.542952, -4.012303);
    //Opcion 3:
    LatLng aulas = new LatLng(40.543515, -4.013181);
    //Opcion 4:
    LatLng colegio = new LatLng(40.545578, -4.012233);

    //Crea el mapa al inicio:
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        MiLocationListener gpsListener = new MiLocationListener();
        //Fijamos posicion destino segun la opcion escogida.
        opcion = MapsRouteB.obtenerEleccion();
        fijarDestino(opcion);
        gpsListener.dibujarRuta(navegacion);
        finish();
    }
    //Con este metodo se fija la posicion de destino a mostrar la ruta:
    public void fijarDestino(int eleccion){
        if(eleccion==1){
            navegacion = new LatLng(biblioteca.latitude,biblioteca.longitude);
        }else if (eleccion==2){
            navegacion = new LatLng(secretaria.latitude,secretaria.longitude);
        }else if (eleccion==3){
            navegacion = new LatLng(aulas.latitude,aulas.longitude);
        }else if (eleccion==4){
            navegacion = new LatLng(colegio.latitude,colegio.longitude);
        }else {

        }
    }
    //Implementación de Loacation Listener personalizado
    public class MiLocationListener implements LocationListener {
        String GPS_A = "Gps Activado";
        String GPS_D = "Gps Desactivado";

        //Inicializamos el GPS
        public void inicializarGPS(){
            mLocationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
            // Creamos un objeto criteria para recuperar el proveedor.
            Criteria criteria = new Criteria();
            // Obtenemos el nombre del mejor proveedor
            provider = mLocationManager.getBestProvider(criteria, true);
            // Getting Current Location
            m_DeviceLocation = mLocationManager.getLastKnownLocation(provider);
        }

        public void dibujarRuta(LatLng puntoFinal){
            String requestedMode = "car"; // or bike or car
            //Se opta por la navegación en coche.
            String mode = "";
            if(requestedMode.equals("walking")) {
                mode = "&mode=w";
            } else if(requestedMode.equals("bike")) {
                mode = "&mode=b";
            } else if(requestedMode.equals("car")) {
                mode = "&mode=c";
            }
```

```
            //Ejecuta el google maps
            //Va directamente a la navegacion
            final Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
                    Uri.parse(String.format("google.navigation:ll=%s,%s%s",
puntoFinal.latitude, puntoFinal.longitude, mode)));
            intent.setClassName( "com.google.android.apps.maps",
"com.google.android.maps.MapsActivity"); startActivity(intent);
        }
        public void onLocationChanged(Location localizacion) {
            m_DeviceLocation = localizacion;
        }

        public void onProviderDisabled(String provider) {
            Toast.makeText(getApplicationContext(), GPS_D, Toast.LENGTH_SHORT).show();
        }

        public void onProviderEnabled(String provider) {
            Toast.makeText( getApplicationContext(),GPS_A,Toast.LENGTH_SHORT ).show();
        }

        public void onStatusChanged(String provider, int status, Bundle extras) {
        }
    }
}
```

Lo primero que hacemos en esta clase esta clase es recuperar la opción que anteriormente ha seleccionado el usuario, es decir, hacia cuál de los 4 destinos posibles ha calculado la ruta.

Dentro de esta clase se ha tenido que crear una subclase que implementa la clase LocationListener, de este modo podemos crear la clase MiLocationListener que es al fin y al cabo un listener de localización personalizado.

- **Clase MiLocationListener.java**

En esta clase es donde se hace la inicialización del GPS y se llama a la app Google Maps para que calcule la ruta correspondiente.

A continuación se explicará en detalle el método que realiza esta función:

- **Método dibujarRuta:**

```
public void dibujarRuta(LatLng puntoFinal){
    String requestedMode = "car"; // or bike or car
    //Se opta por la navegación en coche.
    String mode = "";
    if(requestedMode.equals("walking")) {
        mode = "&mode=w";
    } else if(requestedMode.equals("bike")) {
        mode = "&mode=b";
    } else if(requestedMode.equals("car")) {
        mode = "&mode=c";
    }
    //Ejecuta el google maps
    //Va directamente a la navegacion
    final Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
            Uri.parse(String.format("google.navigation:ll=%s,%s%s", puntoFinal.latitude,
puntoFinal.longitude, mode)));
    intent.setClassName( "com.google.android.apps.maps",
```

```
"com.google.android.maps.MapsActivity"); startActivity(intent);
}
```

Lo primero que hace este método es definir de qué manera se va a realizar la navegación. Existen tres formas posibles de hacerlo: caminando, en coche o en bicicleta.

Para esta funcionalidad se ha optado por el coche, ya que es de prever que cuando el usuario inicie la navegación estará a una distancia lo suficientemente importante para tener que usar un vehículo motorizado en lugar de ir andando o bicicleta.

Finalmente se crea un contener con ambos datos de ubicación y modo de transporte, para posteriormente ejecutar por fuera de nuestra app la navegación de Google Maps.

En Buddy UC3M existen otras actividades que no se han descrito en este capítulo al no ser consideradas relevantes para la funcionalidad final de la app y por tanto para el lector.

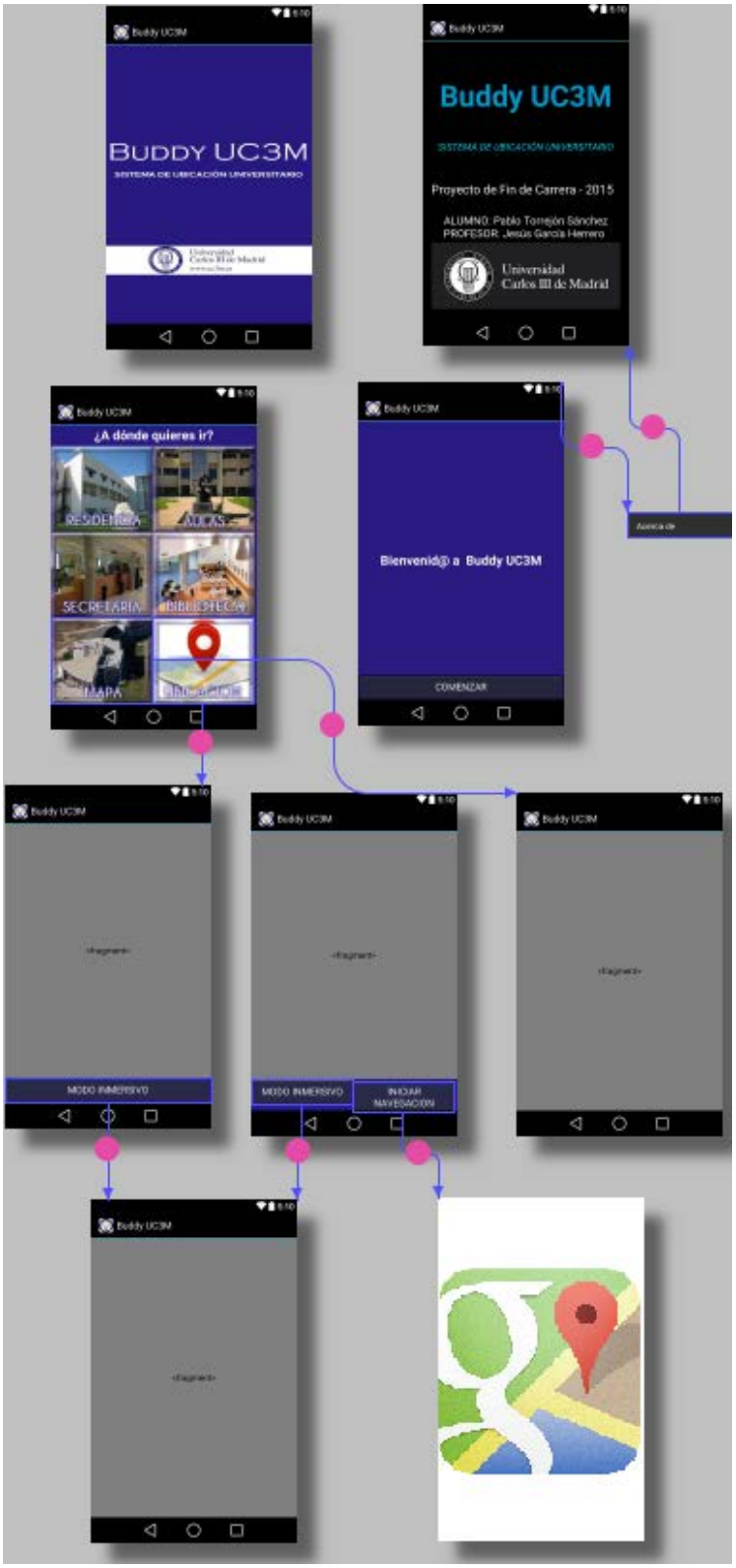
5.4. Mecánica de la navegación

Una vez que se ha podido observar el diagrama de clases y la implementación de cada una de ellas se procede a explicar como es la navegación o comportamiento de la aplicación entre cada una de las clases.

Una cosa son las relaciones a nivel de diseño que guardan las clases entre sí y otra muy diferente las relaciones en la conformación de la app existentes. El resultado de estas relaciones da lugar a un comportamiento o navegación de la aplicación.

La navegación es aplicable cuando realizamos una navegación “hacia adelante” en la app. Para el retroceso en las pantallas finalmente se optó por la gestión automatiza que posee Android de la pila de actividades. Sólo en un caso muy particular como cuando no tenemos el GPS activado se destruye la actividad correspondiente a ajustes del sistema una vez que se activa el GPS. Luego se devuelve el control a la aplicación.

Aquí se puede observar un diagrama de la navegación implementada en la APP:


BUDDY UC3M
SISTEMA DE UBICACIÓN UNIVERSITARIO
Buddy UC3M
Proyecto de Fin de Carrera - 2015
ALUMNO: Pablo Torrejón Sánchez
PROFESOR: Jesús García Herrero
Universidad Carlos III de Madrid
¿A dónde quieres ir?
RESIDENCIA
AULAS
SECRETARIA
BIBLIOTECA
MAPA
Bienvenid@ a Buddy UC3M
COMENZAR
MODO INMERSIVO
MODO INMERSIVO
INICIAR NAVEGACIÓN

Y a continuación la codificación implementada:

```
<?xml version="1.0" encoding="UTF-8"?>
<navigationModel
  ns =
"http://schemas.android.com?import=com.android.tools.idea.editors.navigation.model.*">
  <locations
    class = "java.util.ArrayList">
    <statePointEntry>
      <state
        class = "com.android.tools.idea.editors.navigation.model.ActivityState"
        className = "es.uc3m.buddyuc3m.MapsActivity"/>
      <point
        x = "1806"
        y = "4464"/>
    </statePointEntry>
    <statePointEntry>
      <state
        class = "com.android.tools.idea.editors.navigation.model.ActivityState"
        className = "es.uc3m.buddyuc3m.MapsActivityGPS"/>
      <point
        x = "517"
        y = "4471"/>
    </statePointEntry>
    <statePointEntry>
      <state
        class = "com.android.tools.idea.editors.navigation.model.ActivityState"
        className = "es.uc3m.buddyuc3m.MapsRouteB"/>
      <point
        x = "1106"
        y = "2958"/>
    </statePointEntry>
    <statePointEntry>
      <state
        class = "com.android.tools.idea.editors.navigation.model.ActivityState"
        className = "es.uc3m.buddyuc3m.SplashScreenActivity"/>
      <point
        x = "573"
        y = "42"/>
    </statePointEntry>
    <statePointEntry>
      <state
        class = "com.android.tools.idea.editors.navigation.model.ActivityState"
        className = "es.uc3m.buddyuc3m.MapsActivityP"/>
      <point
        x = "197"
        y = "2932"/>
    </statePointEntry>
    <statePointEntry>
      <state
        class = "com.android.tools.idea.editors.navigation.model.ActivityState"
        className = "es.uc3m.buddyuc3m.OptionActivity"/>
      <point
        x = "368"
        y = "1449"/>
    </statePointEntry>
    <statePointEntry>
      <state
        class = "com.android.tools.idea.editors.navigation.model.MenuState"
        className = "es.uc3m.buddyuc3m.OptionActivity"/>
      <point
        x = "2344"
        y = "36"/>
    </statePointEntry>
    <statePointEntry>
      <state
        class = "com.android.tools.idea.editors.navigation.model.ActivityState"
        className = "es.uc3m.buddyuc3m.Creditos"/>
      <point
        x = "1750"
        y = "28"/>
    </statePointEntry>
    <statePointEntry>
```

```
      <state
        class = "com.android.tools.idea.editors.navigation.model.ActivityState"
        className = "es.uc3m.buddyuc3m.WelcomeActivity"/>
      <point
        x = "1509"
        y = "1438"/>
    </statePointEntry>
    <statePointEntry>
      <state
        class = "com.android.tools.idea.editors.navigation.model.ActivityState"
        className = "es.uc3m.buddyuc3m.MapaUniversidad"/>
      <point
        x = "2092"
        y = "2958"/>
    </statePointEntry>
    <statePointEntry>
      <state
        class = "com.android.tools.idea.editors.navigation.model.MenuState"
        className = "es.uc3m.buddyuc3m.WelcomeActivity"/>
      <point
        x = "2508"
        y = "1920"/>
    </statePointEntry>
  </locations>
</navigationModel>
```

5.5. Layouts e Interfaz Gráfica.

Una layout podría definirse como un archivo XML donde se indican los objetos y las propiedades de una interfaz gráfica en concreto. Por regla general en Android una layout está asociada a una actividad y el resultado de su procesamiento es lo que da lugar interfaz de usuario.

Existen varios tipos de Layouts donde elegir su tipo dependerá del tipo de objetos a representar y su ubicación.

En cada una de las Layouts que se han tenido que definir se ha utilizado el mismo estilo (color de fondo, tipografía, etc.) para dar una sensación de un todo a medida que nos movemos por la app.

Buddy UC3M usa principalmente como objetos o widgets, botones e imágenes a las que se les puede asociar un comportamiento en particular. En nuestro caso al pulsar sobre cualquiera de esto widgets hace que se ejecuten otras actividades.

A continuación se procede a explicar los ficheros XML o Layouts y las interfaces gráficas más representativos implementados en la app.

- splash_screen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SplashScreenActivity" >

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:contentDescription="@string/app_name"
        android:scaleType="fitXY"
        android:src="@drawable/splash_screen" />

</FrameLayout>
```

Esta layout es la que conforma la pantalla de presentación o splash screen, se crea una imagen que será mostrada durante 4 segundos a pantalla completa cuando la app se inicia.


5:10
Buddy UC3M
BUDDY UC3M
SISTEMA DE UBICACIÓN UNIVERSITARIO
Universidad
Carlos III de Madrid
www.uc3m.es

- activity_welcome.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:background="#0099cc"
    tools:context=".WelcomeActivity">
    <TextView android:id="@+id/fullscreen_content" android:layout_width="match_parent"
        android:layout_height="match_parent" android:keepScreenOn="true"
        android:textStyle="bold"  android:gravity="center"
        android:text="Bienvenid@ a  Buddy UC3M"
        android:layout_gravity="top|center"
        android:background="#ff2a1b81"
        android:textSize="24dp"
        android:textColor="@android:color/white" />
    />
    <FrameLayout android:layout_width="match_parent" android:layout_height="match_parent"
        android:fitsSystemWindows="true">

        <LinearLayout android:id="@+id/fullscreen_content_controls"
style="?metaButtonBarStyle"
            android:layout_width="match_parent" android:layout_height="wrap_content"
            android:layout_gravity="bottom|center_horizontal"
            android:background="@color/black_overlay" android:orientation="horizontal"
            tools:ignore="UselessParent">

            <Button android:id="@+id/button_options" style="?metaButtonBarButtonStyle"
                android:layout_width="0dp" android:layout_height="wrap_content"
                android:layout_weight="0.14" android:text="@string/button_options"
                android:onClick="@string/title_activity_fullscreen_activity_o" />

        </LinearLayout>
    </FrameLayout>
</FrameLayout>
```

Tras mostrar la pantalla de presentación, este fichero XML es el que conforma la interfaz de la pantalla de bienvenida. Hay varios tipos de layouts anidadas unas dentro de otras. Se ha tomado esta opción para poder definir el botón con el texto comenzar.

Desde esta pantalla accedemos a los créditos del proyecto que se mostrarán en el siguiente apartado.


5:10
Buddy UC3M
Bienvenid@ a Buddy UC3M
COMENZAR

- activity_creditos.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".WelcomeActivity"
    android:background="@android:color/black">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/textBuddyUC3M"
        android:id="@+id/textBuddyUC3M"
        android:linksClickable="false"
        android:textSize="55dp"
        android:textAlignment="center"
        android:textStyle="bold"
        android:textColor="@android:color/holo_blue_dark"
        android:layout_above="@+id/textBuddy_2"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="51dp" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/textPFC"
        android:id="@+id/textPFC"
        android:linksClickable="false"
        android:textSize="23dp"
        android:textAlignment="center"
        android:textColor="@android:color/white"
        android:layout_above="@+id/textAutor"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_marginBottom="32dp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/textAutor"
        android:id="@+id/textAutor"
        android:linksClickable="false"
        android:textSize="20dp"
        android:textAlignment="center"
        android:textColor="@android:color/white"
        android:layout_above="@+id/textProfesor"
        android:layout_centerHorizontal="true" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/textProfesor"
        android:id="@+id/textProfesor"
        android:linksClickable="false"
        android:textSize="20dp"
        android:textAlignment="center"
        android:textColor="@android:color/white"
        android:layout_above="@+id/imageView"
        android:layout_alignLeft="@+id/textAutor"
        android:layout_alignStart="@+id/textAutor" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/textBuddyUC3M_2"
        android:id="@+id/textBuddy_2"
        android:textSize="18dp"
        android:textColor="@android:color/holo_blue_bright"
        android:textStyle="italic"
        android:layout_above="@+id/textPFC"
        android:layout_centerHorizontal="true"
```

```
        android:layout_marginBottom="55dp" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@drawable/logo_letras_negro"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

</RelativeLayout>
```

Para esta layout se ha usado una layout de tipo relativa para poder colocar objetos de tipo texto e imágenes teniendo el siguiente resultado.


5:10
Buddy UC3M
Buddy UC3M
SISTEMA DE UBICACIÓN UNIVERSITARIO
Proyecto de Fin de Carrera - 2015
ALUMNO: Pablo Torrejón Sánchez
PROFESOR: Jesús García Herrero
Universidad
Carlos III de Madrid

- activity_options.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:background="#ff2a1b81"
    tools:context=".OptionActivity"
    android:orientation="vertical"
    android:id="@+id/layoutPrincipalOpciones"
    android:weightSum="1">

    <TextView
        android:id="@+id/fullscreen_content"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:keepScreenOn="true"
        android:textStyle="bold"
        android:gravity="center"
        android:text="@string/title_options"
        android:layout_gravity="center_horizontal|top"
        android:background="#ff2a1b81"
        android:textSize="24dp"
        android:textColor="@android:color/white"
        android:layout_weight="1" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:weightSum="1">
        <GridLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <ImageButton
                android:layout_width="190dp"
                android:layout_height="160dp"
                android:id="@+id/imageResidencia"
                android:src="@drawable/residencia"
                android:contentDescription="@string/residencia" />
            <ImageButton
                android:layout_width="190dp"
                android:layout_height="160dp"
                android:id="@+id/imageAulas"
                android:src="@drawable/aulas"
                android:contentDescription="@string/aulas"/>
        </GridLayout>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:weightSum="1">
        <GridLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <ImageButton
                android:layout_width="190dp"
                android:layout_height="160dp"
                android:id="@+id/imageSecretaria"
                android:src="@drawable/secretaria"
                android:contentDescription="@string/campus" />
            <ImageButton
                android:layout_width="190dp"
                android:layout_height="160dp"
                android:id="@+id/imageBiblioteca"
                android:src="@drawable/bibilioteca"
                android:contentDescription="@string/biblioteca"/>
        </GridLayout>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:weightSum="1">
        <GridLayout
```

```
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <ImageButton
                android:layout_width="190dp"
                android:layout_height="160dp"
                android:id="@+id/imageCampus"
                android:src="@drawable/mapa"
                android:contentDescription="@string/campus" />
            <ImageButton
                android:layout_width="190dp"
                android:layout_height="160dp"
                android:id="@+id/imageUbicacion"
                android:src="@drawable/ubicacion"
                android:contentDescription="@string/ubicacion"/>
        </GridLayout>
    </LinearLayout>
</LinearLayout>
```

Esta pantalla es la más compleja en cuanto a diseño en la aplicación. Hubo que diseñar los botones uno a uno utilizando imágenes de la universidad.

Tal y como se puede observar en el código una linear layout contiene tres linear layouts (una por cada fila de dos botones) y estas a su vez una grid layout donde se colocan cada pareja de botones.

Asimismo los botones tienen todos las misma dimensiones y se les ha procesado creando un biselado a su alrededor que refleja la imagen.

Se ha bloqueado la orientación de la pantalla para una correcta visualización y uso de la app.


5:10
Buddy UC3M
¿A dónde quieres ir?
RESIDENCIA
AULAS
SECRETARIA
BIBLIOTECA
MAPA
UBICACION

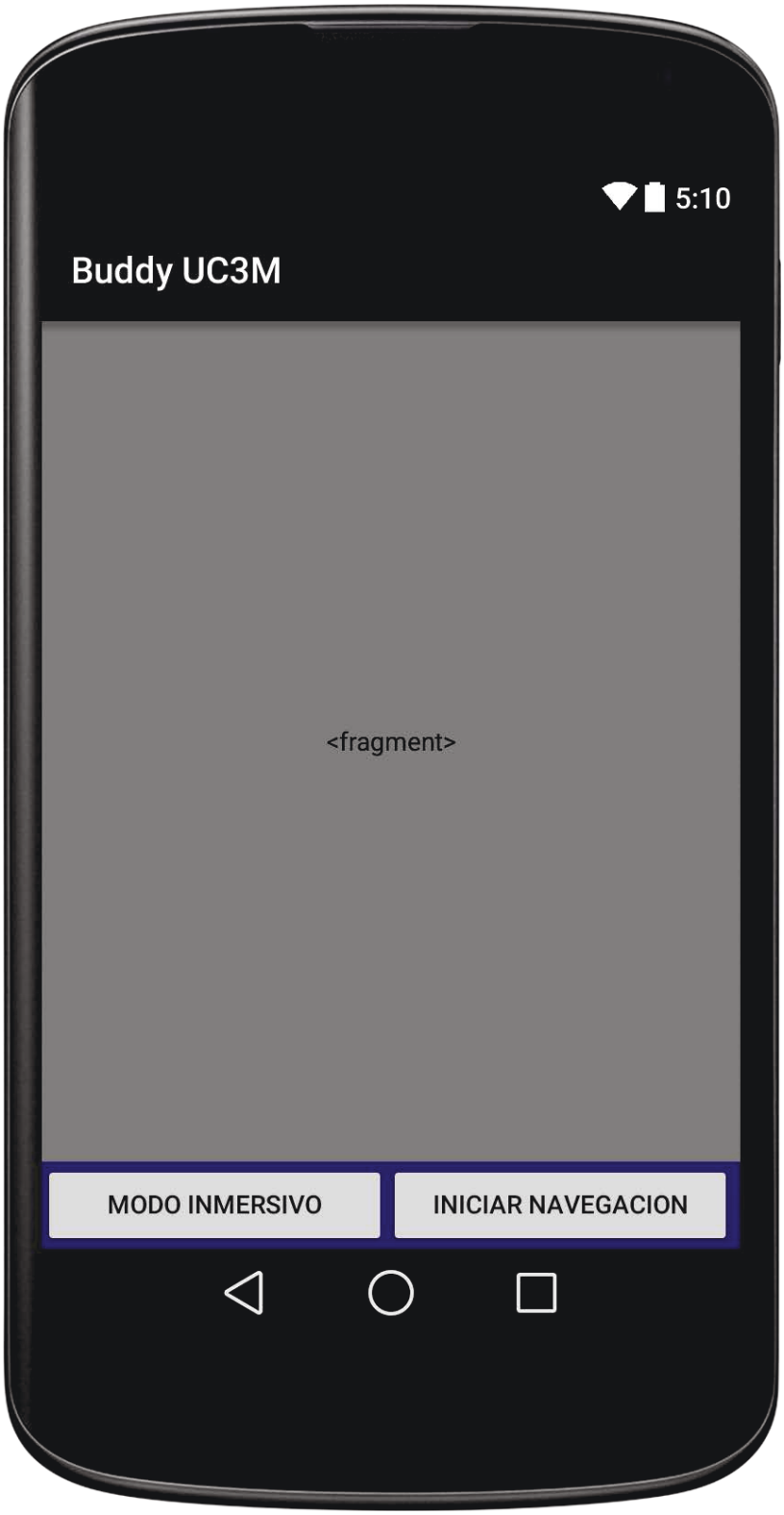
En las siguientes Layouts que se muestran es donde se hace el uso de Fragments como contenedores del objeto mapa Google. Dependiendo de la función seleccionada, la pantalla que se muestra difiere la una de la otra.

- activity_maps_route_b.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff2a1b81"
    tools:context=".MapsRouteB"
    android:orientation="vertical"
    android:id="@+id/layoutMapsRouteB"
    android:weightSum="1">
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1">
        <fragment xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+id/map"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:name="com.google.android.gms.maps.MapFragment" />
    </FrameLayout>
    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button android:id="@+id/button_inmersivo" style="?metaButtonBarButtonStyle"
                android:layout_width="190dp" android:layout_height="wrap_content"
                android:text="@string/button_inmersivo" />

            <Button android:id="@+id/button_navegacion" style="?metaButtonBarButtonStyle"
                android:layout_width="190dp" android:layout_height="wrap_content"
                android:text="@string/button_navegacion" />
        </TableRow>

    </TableLayout>
</LinearLayout>
```

Tal y como se puede apreciar, esta interfaz está contenida en una layout de tipo lineal. Para situar el fragment donde irá el mapa se ha utilizado una Frame Layout.

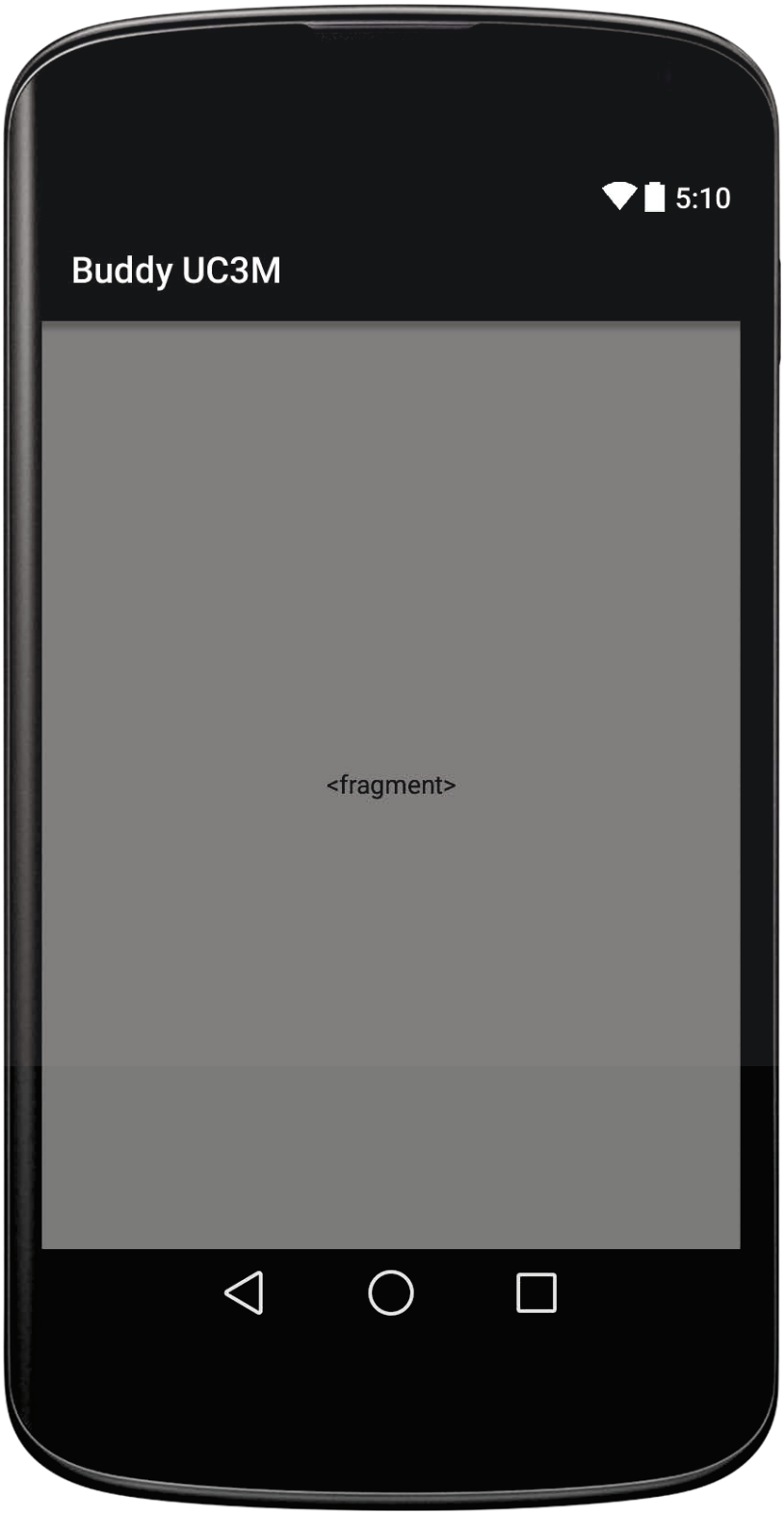
Asimismo a continuación de esa capa se ha puesto una layout en modo tabla donde se colocan los botnes de MODO INMERSIVO e INICIAR NAVEGACIÓN. Gracias a este tipo de implementación, si se gira el dispositivo todo redimensiona.


5:10
Buddy UC3M
<fragment>
MODO INMERSIVO
INICIAR NAVEGACION

- activity_universidad.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff2a1b81"
    tools:context=".MapaUniversidad"
    android:orientation="vertical"
    android:id="@+id/layoutUniversidad"
    android:weightSum="1">
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1">
        <fragment xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+id/map"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:name="com.google.android.gms.maps.MapFragment" />
    </FrameLayout>
</LinearLayout>
```

Esta pantalla sólo está conformada por un fragmente donde se alberga el Mapa del Campus de Colmenarejo ocupando la superficie máxima de pantalla disponible.


5:10
Buddy UC3M
<fragment>

- activity_maps_p.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff2a1b81"
    tools:context=".MapsActivityP"
    android:orientation="vertical"
    android:id="@+id/layoutMapsP"
    android:weightSum="1">
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1">
        <fragment xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+id/map"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:name="com.google.android.gms.maps.MapFragment" />
    </FrameLayout>

    <Button android:id="@+id/button_inmersivo" style="?metaButtonBarButtonStyle"
        android:layout_width="match_parent" android:layout_height="wrap_content"
        android:text="@string/button_inmersivo" />

</LinearLayout>
```

Este código corresponde a la interfaz de la app cuando el usuario quiere ver su ubicación en un mapa de satélite. Al final de la pantalla se le ha agregado la funcionalidad de acceder a la vista panorámica en 360 grados de donde se encuentre.

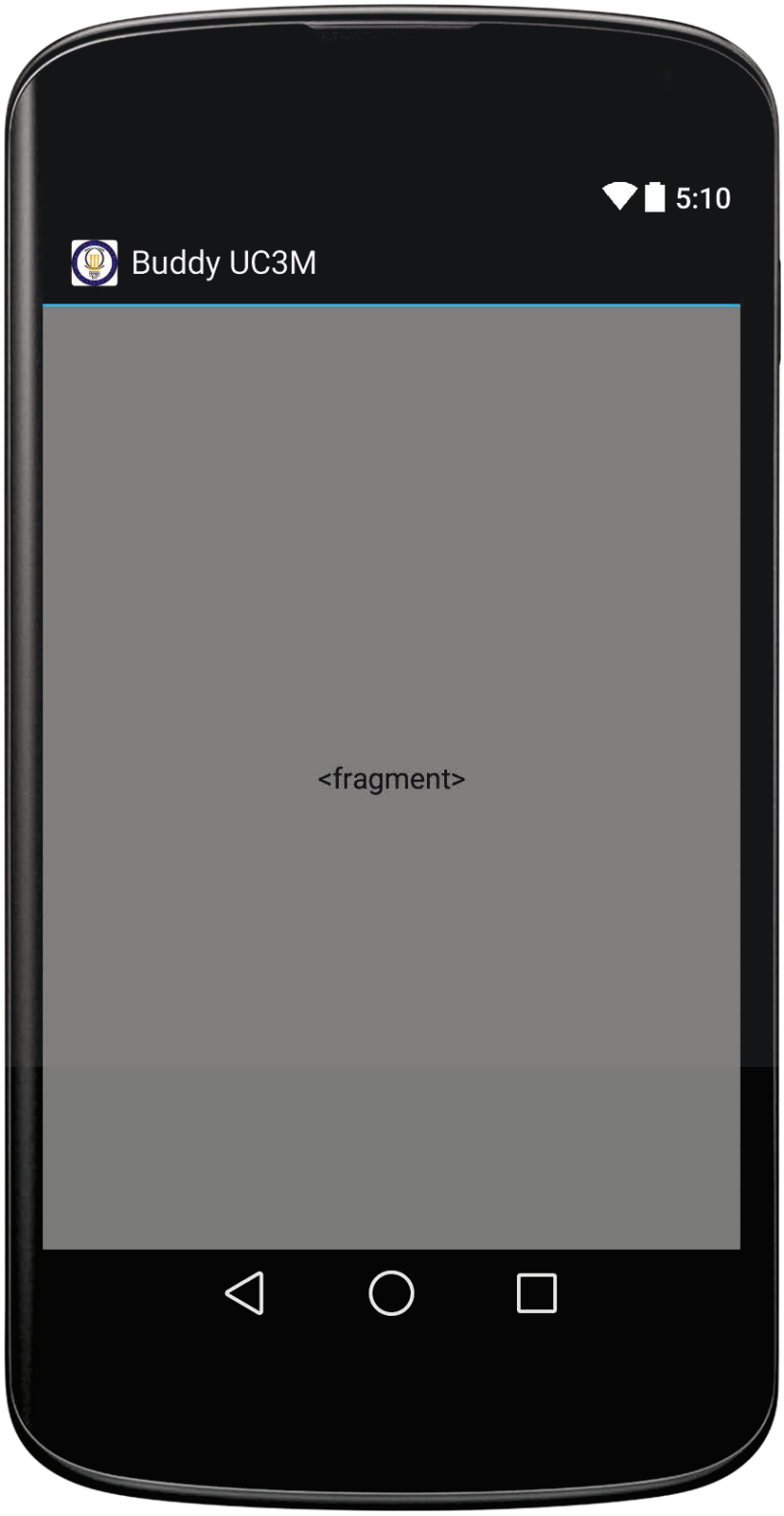
La app se ha modeliza para que aun cambiando de orientación el dispositivo la interacción que hayamos realizado sobre el mapa, no se reinicie y vuelva al punto origen de nuestra ubicación actual con los valores de mapa originales.


5:10
Buddy UC3M
<fragment>
MODO INMERSIVO

- activity_maps_gps.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#ff2a1b81"
tools:context=".MapsActivityGPS"
android:orientation="vertical"
android:id="@+id/layoutMapsGPS"
android:weightSum="1">
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1">
    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:name="com.google.android.gms.maps.StreetViewPanoramaFragment" />
</FrameLayout>
</LinearLayout>
```

Y ya acabamos el diseño de la interfaz gráfica con la pantalla que embebe un fragmento de tipo panorama para acoger la vista 3D. Se accede a esta pantalla siempre que el usuario pulsa sobre el botón MODO INMERSIVO.


5:10
Buddy UC3M
<fragment>

6) Pruebas

La parte de realización de pruebas fue un proceso continuado a lo largo de la realización de todo proyecto. Podría decirse que las pruebas podrían dividirse en dos tipos: las orientadas al funcionamiento correcto de la app y las orientadas al ajuste de la información proporcionada por el receptor GPS del dispositivo.

Para las pruebas relativas a la app se ha utilizado la herramienta ADB de Android con un dispositivo real conectado al equipo de desarrollo. Las siglas ADB significan Android Debug Bridge y es una herramienta que nos permite controlar el estado de nuestro Smartphone Android.

Se puede decir que en ADB nos encontramos con una colección de herramientas y comandos útiles que nos ayudarán a realizar diversas acciones como por ejemplo, actualizar el sistema, ejecutar comandos shell, administrar el direccionamiento de puertos, copiar archivos, acceder al modo recovery, etc.

Para la realización de todas estas acciones es necesario que el Smartphone se encuentre conectado vía USB al ordenador donde se encuentra instalado el entorno de desarrollo Android Studio.

El principal motivo por lo que se decidió no utilizar el emulador que posee el entorno de desarrollo, fue porque a la hora de obtener una ubicación había que crear ubicaciones simuladas y parece ser que según diferentes expertos esto no daba muy buenos resultados.

En la realización de estas pruebas se ha hecho mucho hincapié en que el GPS se encuentre activado para utilizar la app. Para ello se han implementado mecanismos de control para que en caso de que el usuario no tenga el GPS activado le aparezca un mensaje emergente donde si lo desea le lleva directamente a las opciones del sistema para activarlo. Asimismo se han incluido notificaciones Toast para que en caso de que el usuario desactive el GPS durante el empleo de Buddy UC3M le avise.

El motivo de tener tanto control sobre los servicios del sistema GPS es que la aplicación necesita en todo momento cuando se están utilizando la funcionalidad del cálculo de rutas, ubicación actual, etc. una ubicación con la latitud y longitud informadas. En caso contrario la app se cierra lanzando una excepción de puntero nulo.

Asimismo otra parte importante de las pruebas es la que tienen que ver con el comportamiento de la interfaz. En esta aplicación se ha buscado simplificar al máximo su uso. No existen parámetros a configurar y el acceder a cualquier de las 3 opciones disponibles se hace fácilmente con pulsar un botón.

Se ha tratado de que la app sea usable para cualquier dimensión de pantalla del dispositivo, no obstante por razones obvias no se ha podido probar en dispositivos con diferentes configuraciones de pantalla.

Una característica que se ha implementado en la app es que cuando el usuario realiza el giro del dispositivo para cambiar su orientación y con ello obtener una mejor visualización, las características de la función y la vista se conservan. Si no se ha trabajado en habilitar y configurar esta característica, la actividad se reiniciaría en cada cambio de orientación.

La siguiente parte de las pruebas fueron las realizadas para ajustar el funcionamiento del receptor GPS y su representación en la app y para ello se han hecho pruebas en espacios abiertos con la aplicación instalada en varios dispositivos.

Principalmente se ha trabajado con el tiempo de actualización de la posición GPS, después de muchas pruebas inclusive en vehículos en movimientos. Se determinó que la mejor configuración de las actualizaciones de datos de posición para obtener un equilibrio entre precisión y consumo de batería fue la obtener la nueva posición GPS cada vez que el usuario se desplazara a 10 metros de su última ubicación o cada 10 segundos.

Para comprobar hasta qué punto el proceso de cálculo de rutas soporta gran volumen de cálculos, se solicitó que se calculara la ruta a cualquiera de los cuatro puntos de interés configurados desde 600 km de distancia. El resultado fue el esperado y no se apreciaron mayores tiempos de cálculo que los que se tienen cuando se calcula una ruta de tan sólo 40 kilometros.

Por último indicar que el Smartphone que ha sido principalmente utilizado para la realización de las pruebas ha sido un Google Nexus 4. Para verificar el correcto funcionamiento de la app, se exportó a un empaquetado de tipo APK sin firmar digitalmente y se instaló en diversos dispositivos. El comportamiento y funcionamiento mostrado en las pruebas fue el esperado.

6.1. Casos de prueba

A continuación se procede a realizar la explicación formal de los casos de prueba efectuados.

Cada caso de prueba efetuado se especificará con el siguiente formato.

- **ID**: Clave unívoca que idéntica el caso de prueba.

- **Mecánica**: Explicación de los pasos de la prueba realizada.

ID	P-01
Mecánica	Inicio de la aplicación. Pulsar sobre botón **COMENZAR**. Seleccionar cualquiera de los cuatro botones hacia donde se desea calcular la ruta. Se muestra en pantalla la ruta calculada desde nuestra ubicación.

Prueba 01: Cálculo de ruta.

ID	P-02
Mecánica	Inicio de la aplicación. Pulsar sobre botón **COMENZAR**. Seleccionar el botón **MAPA**. Se muestra en pantalla el mapa interactivo del Campus de Colmenarejo.

Prueba 02: Mapa del Campus.

ID	P-03
Mecánica	Inicio de la aplicación. Pulsar sobre botón **COMENZAR**. Seleccionar el botón **UBICACION**. Se muestra en pantalla el mapa interactivo vía satélite dela ubicación del usuario.

Prueba 03: Ubicación usuario mapa satélite.

ID	P-04
Mecánica	Inicio de la aplicación. Pulsar sobre botón **COMENZAR**. Seleccionar cualquiera de los cuatro botones hacia donde se desea calcular la ruta. Pulsar sobre el botón **INICIAR NAVEGACION**. Se ejecuta por fuera de nuestra app la aplicación Google Maps y comienza de modo automático la navegación guiada hasta el punto destino seleccionado.

Prueba 04: Navegación hasta un punto predefinido.

ID	P-05
Mecánica	Inicio de la aplicación. Pulsar sobre botón **COMENZAR**. Seleccionar cualquiera de los cuatro botones hacia donde se desea calcular la ruta. Pulsar sobre el botón **MODO INMERSIVO**. Se muestra una vista interactiva de Google Street View en nuestra ubicación actual embebida en la app.

Prueba 05: Mostrar entorno con Google Street View.

ID	P-05B
Mecánica	Inicio de la aplicación. Pulsar sobre botón **COMENZAR**. Seleccionar el botón **UBICACION**. Pulsar sobre el botón **MODO INMERSIVO.** Se muestra una vista interactiva de Google Street View en nuestra ubicación actual embebida en la app.

Prueba 05 B: Mostrar entorno con Google Street View.

ID	P-06
Mecánica	Inicio de la aplicación. Pulsar sobre el desplegable de la esquina superior derecha. Seleccionar **Acerca de**. Se muestra en pantalla los créditos relativos a Buddy UC3M.

Prueba 06: Créditos de aplicación.

ID	P-07
Mecánica	Inicio de la aplicación con los servicios GPS deshabilitados. La aplicación informa al usuario mediante una alerta que Buddy UC3M necesita tener el GPS activo. Se dan dos opciones: - **NO, FINALIZAR** - **ACTIVAR GPS** Si se selecciona la primera opción la app finaliza. Si selecciona activar el GPS, se redirige al usuario a las opciones del sistema para que lleve a cabo la acción. Se muestra notificación Toast del estado del GPS. Al volver atrás se retorna a la pantalla de bienvenida de Buddy UC3M.

Prueba 07: GPS Habilitado.

ID	P-08
Mecánica	Verificar que nos encontramos ante un terminal con sistema Operativo Android instalado. Si es así se puede proceder a la instalación de Buddy UC3M aceptando los permisos requeridos.

Prueba 08: Sistema Operativo.

ID	P-09
Mecánica	Verificar que tenemos acceso a Internet. Para ello se ha de revisar si los datos móviles se encuentran activados. En caso contrario habilitar.

Prueba 09: Acceso a Internet.

ID	P-10
Mecánica	Inicio de la aplicación sin Servicios Google Play instalados. La aplicación informa al usuario mediante una alerta que Buddy UC3M necesita tener instalados los servicios Google Play. Se da la opción al usuario de visitar la tienda de Google mediante la ejecución de la aplicación Play Store para así poder descargarlos y proceder a su instalación. Una vez instalados el inicio de Buddy UC3M puede continuar con normalidad.

Prueba 10: Servicios Google Play.

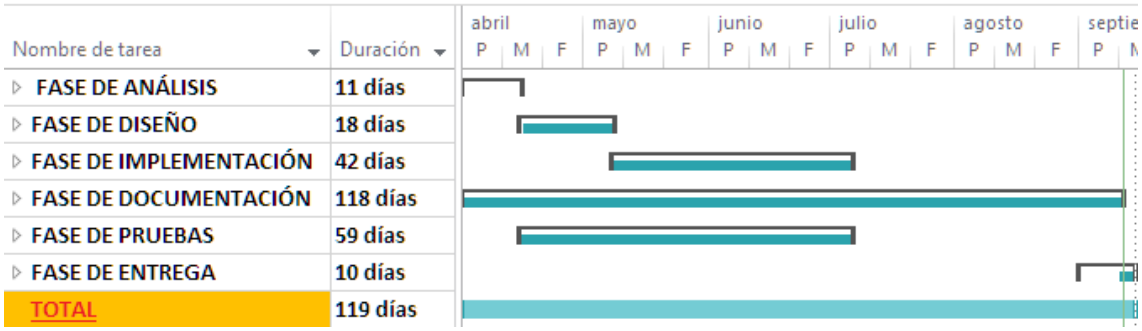
ID	P-11
Mecánica	Inicio de la aplicación sin la aplicación Google Maps instalada. La aplicación informa al usuario mediante una alerta que Buddy UC3M necesita tener instalada la app Google Maps. Se da la opción al usuario de visitar la tienda de Google mediante la ejecución de la aplicación Play Store para así poder descargarla y proceder a su instalación. Una vez instalada el inicio de Buddy UC3M puede continuar con normalidad.

Prueba 11: App Google Maps.

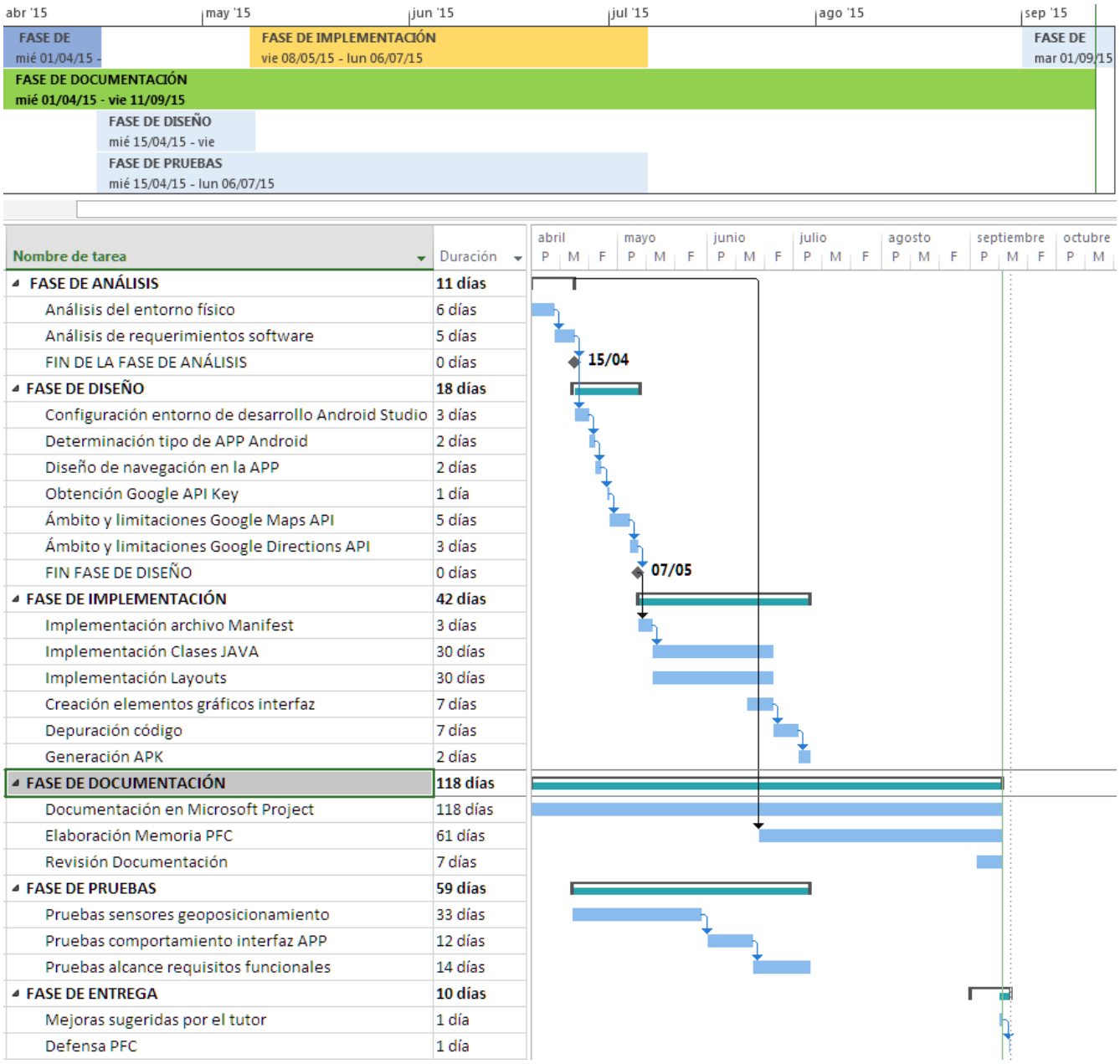
7) Presupuesto

Para calcular el presupuesto aproximado del desarrollo de la app habría que tener en cuenta los costes humanos y materiales.

En el diagrama de Gant resumiedo que se aprecia a continuación se puede observar que en el tiempo de desarrollo aproximado para la app se han invertido unos 120 días. Debido a mis circunstancias actuales, el tiempo de dedicación diario habrá sido de unas 4 horas.



7.1. Diagrama de Gant



Es por ello por lo que si se calculan las horas invertidas obtenemos 120*4 se obtiene un total de 480 horas. Si se tiene en cuenta un desempeño de 8 horas por jornada laboral, para la realización de la app se habrán invertido unos 60 días o lo que viene a ser lo mismo unos 3 meses de jornada laboral estándar.

7.2. Tabla de costes

Actualmente por desgracia, el sueldo medio de un Ingeniero Técnico Informático con poca experiencia laboral ronda los 1400 euros netos mensuales.

Es por ello por lo que si consideramos que un mes tiene 20 días laborables podría decirse que la app hubiera costado 3 meses de desarrollo lo que ascendería a un coste de unos 4200 euros con un ingeniero dedicado en su totalidad a ella.

A este coste de recursos humanos, habría que añadirle el coste de los equipos empleados. En mi caso los equipos que he usado son: Ordenador de sobremesa de alta potencia montado por mi valorado en 1300 euros y teléfono móvil Google Nexus 4 valorado en 350 euros, es decir un total de 1650 euros.

A continuación se procede a detallar los costes inherentes al proyecto a través de la siguiente hoja de cálculo.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:

Pablo Torrejón Sánchez

2.- Departamento:

Departamento de Informática

3.- Descripción del Proyecto:

- Titulo	**Buddy UC3M**
- Duración (meses)	**3**
Tasa de costes Indirectos:	**20%**

4.- Presupuesto total del Proyecto (valores en Euros):

Euros	5139

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)
Torrejón Sánchez, Pablo		Ingeniero Técnico	3	1.400,00	4.200,00
		Hombres mes	**3**	**Total**	4.200,00

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Equipo Informático	1.300,00	100	3	60	65,00
Teléfono Google Nexus 4	350,00	100	3	60	17,50
				Total	82,50

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Software Android Studio*	Google	0,00
Windows 7*	Microsoft	0,00
Office 365*	Microsoft	0,00
*Software freeware o con licencia de estudiante.	**Total**	0,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	4.200
Amortización	83
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	857
Total	5.139

Finalmente y tras analizar los costes en los que se incurre el proyecto, despreciando el precio de los suministros (luz) y de alquiler de una oficina, el presupuesto estimado sería de unos **5139 euros.** Hay que tener en cuenta que este precio sería si la persona que desarrolla, como en mi caso, ha tenido que aprender a desarrollar para Android desde el principio.

8) Herramientas software

Para la realización de este proyecto se han empleado varias herramientas que pueden distinguirse fácilmente según el ámbito que se hayan empleado.

En la parte de la confección de la aplicación se ha empleado el entorno de desarrollo Android Studio. Es una herramienta gratuita proporcionada por la empresa Google.

Google Android Studio - www

Para realizar el modelado UML integrado se ha empleado el plugin integrado en Android Studio llamado SimpleUML.

SimpleUML - www

Una vez codificado el proyecto se ha utilizado la herramienta Gradle integrada en el entorno de desarrollo Android que básicamente sirve para automatizar la construcción de nuestros proyectos, como por ejemplo las tareas de compilación, testing, empaquetado y el despliegue de los mismos.

Gradle - www

Relativo a la documentación de este proyecto se han utilizado básicamente dos herramientas de Microsoft.

Para poder llevar a cabo la planificación adecuada del proyecto estableciendo ventanas con tiempo y plazos se ha empleado la herramienta MS Project.

Y posteriormente para la redacción de la memoria se ha utilizado el MS Word. Ambas herramientas pertenecen al paquete ofimático Microsoft Office disponiendo de una licencia de estudiante para su uso.

Microsoft Office - www

Por último para la entrega de la memoria se ha optado por suministrarla en formato PDF. Con ello se asegura que el documento no podrá ser editado además de ser uno de los formatos más extendidos en el mundo de la documentación.

9) Bibliografía

Para el desarrollo de este PFC la mayoría de los recursos que se han utilizados se encontraban disponibles públicamente de forma electrónica en Internet.

No obstante para llegar a las conclusiones y acciones que se han llevado a cabo en este desarrollo se ha recurrido a numerosa documentación, publicaciones, foros, blogs, etc.

Dentro de la variedad de información consultada, los recursos más empleados fueron los siguientes que se encuentran de manera gratuita:

9.1. Libros

Carmen de Pablos Heredero, Victor Loyola Izquierdo, José Jooaquín López - Hermoso Agius (2006) "**Dirección y gestión de los sistemas de información en la empresa**" pp. 144-170

9.2. Recursos electrónicos

Estos se encuentran ordenados a medida que fueron requeridos y para algunos hay referencias a ellos durante el texto:

- **Entorno de desarrollo Android Studio**
https://developer.android.com/sdk/index.html

- **Localización GSM**
https://es.wikipedia.org/wiki/Localizaci%C3%B3n_GSM

- **Localización GPS**
https://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global

- **Términos de servicio API Google Map y Google Earth.**
https://developers.google.com/maps/terms#10-license-restrictions

- **Plano Campus de Colmenarejo**
http://www.uc3m.es/ss/Satellite/UC3MInstitucional/es/TextoMixta/1371206708023/

- **Google Maps API**
https://developers.google.com/maps/documentation/android/

- **Foros de opinión de donde se solucionaron y obtuvieron parte de las ideas implementadas.**
http://stackoverflow.com/
http://www.lawebdelprogramador.com/foros/Android/index1.html
http://www.sgoliver.net/foro/viewforum.php?f=3
http://www.androidpit.es/foro/foro-de-desarrolladores-android

10) Conclusión y futuribles

Tras finalizar el desarrollo de la app Buddy UC3M se puede decir que se ha conseguido alcanzar los objetivos planteados y crear valor añadido a la visita de un nuevo estudiante al Campus de Colmenarejo.

Para ello se ha abordado el proyecto con la intención de explotar las características y limitaciones del geoposicionamiento en los dispositivos móviles. Además se ha explotado las características de la API de Google Maps en la gestión, interacción y presentación de mapas.

También se ha conseguido integrar en la aplicación el cálculo de rutas y representación sobre un mapa que no se podía realizar nativamente con las características ofrecidas por el servicio Google Maps. Para ello se combinó junto con Google Directions.

Asimismo creo que para estudiantes veteranos les puede ayudar de una manera rápida a ubicar un aula desconocida en periodo de exámenes simplemente con mostrar el mapa del Campus e interactuando sobre el edificio Miguel de Unamuno, mostrar su ubicación satélite en cualquier otro lugar o curiosear con la vista en tres dimensiones que provee Google Street View.

En cuanto a las aplicaciones futuras de mi aplicación, creo que se podría extender a otro Campus de la Universidad Carlos III o incluso hacia otras instituciones académicas. Esto sería posible gracias a la manera en la que se ha llevado a cabo implementación de la app, ya que simplemente con modificar la interfaz y los puntos destino, los algoritmos y funciones existentes siguen siendo válidas.

Por último indicar que Buddy UC3M se ha desarrollado por y para el Campus de Colmenarejo ya que ha sido el lugar donde he realizado la mayor parte de mis actividades académicas y he considerado que era buen homenaje hacia la institución.

En Madrid, a 12 de Septiembre de 2015.
Pablo Torrejón Sánchez.