Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una aplicación móvil Android para mejorar la integración de los estudiantes de intercambio en el aspecto de movilidad mediante el uso de mapas estáticos y Google Maps.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Cristian López Espejo

Tutores: Juan Carlos Ruiz García

David de Andrés Martínez

2013-2014

Resumen

En este proyecto final de grado se desarrolla una aplicación para la plataforma Android con la cual se pretende ayudar a la integración de los estudiantes de intercambio que van a estudiar a la Universitat Politècnica de València (UPV).

Para conseguir desarrollar esta aplicación primero se estudió el alcance que podríamos tener dependiendo el tiempo disponible y el necesario para implementar todas las funcionalidades. Se ha utilizado el IDE oficial de Android llamado Android Studio y se ha basado en la Guía de Intercambio que proporciona la UPV a los alumnos de intercambio.

Se ha conseguido una aplicación muy útil y fácil de utilizar y disponible en Google Play.

Palabras clave: Android, ERASMUS, aplicación, GoogleMaps, app.

Abstract

This project develops an Android application to welcome and support the integration of incoming students in the Universitat Politecnica de Valencia (UPV).

First, the time and extend of the project was carefully studied, in collaboration with the international relations office of the univeristy, in order to define the main features required from the app. Then, Android Studio, which is the facto standard for the development of Android apps, was used to implement the app, which is currently available at the Google Play Store.

Keywords: Android, ERASMUS, forum, Valencia, UPV, GCM, app.

Tabla de contenidos

1. Introducción 10
   1.1 Motivación 10
   1.2 Estructura del documento 11
2. Android 12
   2.1 Estructura del documento 12
   2.2 Versiones 12
   2.3 Arquitectura 13
3. Estado del arte 16
   3.1 Sistemas operativos móviles 16
   3.2 Aplicaciones similares 18
4. Objetivos 22
5. Especificación 23
   5.1 Análisis 23
      5.1.1 Requisitos de la aplicación 23
   5.2 Casos de uso 27
   5.3 Bocetos de la interfaz 28
   5.4 Modelo de datos 30
   5.5 Planificación 31
6. Implementación 32
   6.1 Inicio 37
   6.2 Calendario 38
      6.2.1 Diseño de interfaces 38
      6.2.2 Aspectos técnicos 44
   6.3 Información 49
      6.3.1 Diseño de interfaces 49
      6.3.2 Aspectos técnicos. 56
   6.4 Localización 63
      6.4.1 Diseño de interfaces 63
      6.4.2 Aspectos técnicos 72
   6.5 Opciones 77
      6.5.1 Diseño de interfaces 77

6.1.1 Aspectos técnicos 78
7. Pruebas 83
7.1 Pruebas de rendimiento 83
7.2 Pruebas de satisfacción del usuario 85
8 Conclusiones 89
9 Mejoras futuras 90
9.1 Inicio 90
9.2 Calendario 90
9.3 Información 90
9.4 Localización 90
Glosario 91
Bibliografía 93
Anexos 95

Tabla de figuras

Figura 1 Versiones de android a fecha de 12/08/2014 13
Figura 2 Arquitectura android 14
Figura 3 Cuota de mercado de sistemas operativos móviles 16
Figura 4 Mercado actual de sistemas operativos móviles 17
Figura 5 Aplicación UPV 18
Figura 6 Estudia UJA 18
Figura 7 Fallo en la aplicación estudia UJA 19
Figura 8 Universidad de Valencia 19
Figura 9 Harvard mobile 20
Figura 10 University of phoenix mobile 20
Figura 11 Interfaz de University of Phoenix 20
Figura 12 Herramientas de la especificación 23
Figura 13 Pantalla de carga 25
Figura 14 Casos de uso 27
Figura 15 Bocetos de interfaz del frawer e inicio 28
Figura 16 Bocetos de mapas e información 29
Figura 17 Modelo entidad relacion de la base de datos 30
Figura 18 Diagrama de gantt del proyecto 31
Figura 19 Navigation drawer 33
Figura 20 Elemento del drawer 33
Figura 21 Diseño de la pantalla principal 35
Figura 22 Pantalla de inicio 36
Figura 23 Pantalla tutorial 37
Figura 24 Carga de eventos del calendario 38
Figura 25 Vista personalizada del evento 39
Figura 26 Notificacion de alarma (izq.) y informacion de evento (Der.) 40
Figura 27 Filtro de eventos por asignatura 41
Figura 28 Filtro de eventos por Fecha 42
Figura 29 Filtro de eventos por Fecha 43
Figura 30 Quitar filtro 43
Figura 31 Pantalla de inicio de sesión 44
Figura 32 Proceso de login y obtención de calendarios 45
Figura 33 Array json con los calendarios del alumno 45
Figura 34 Creacion del calendario ical 46
Figura 35 Rellenado de la lista 46
Figura 36 Creación de una alarma 47
Figura 37 Clase alarmreceiver 47
Figura 38 Clase alarmmanager 48
Figura 39 Pantalla de información 50
Figura 40 Pantalla de información de la upv 51
Figura 41 Escuelas (DER.) e informacion sobre las ESCUELAS (izq) 52
Figura 42 Asignaturas (DER.) e información sobre las asignaturas (izq) 53

Figura 43 Información sobre valencia....54
Figura 44 Fragment avión (DER.) y fragment tren (IZQ)....55
Figura 45 Division de interfaz del menu de informacion....56
Figura 46 Distribucion de una lista desplegable....57
Figura 47 Autolink de la lista desplegable....58
Figura 48 Distribucion viewpager....59
Figura 49 Método de creación del viewpager....60
Figura 50 Archivos XML de información (seleccionados)....61
Figura 51 Llamada al parser de transportes....61
Figura 52 Parseador de transportes....61
Figura 53 Menú Localización....64
Figura 54 Mapa de la UPV (IZQ) e información de los edificios (DER)....65
Figura 55 Icono de realidad aumentada....66
Figura 56 Mapa Valenbisi (IZQ) e información sobre las estaciones (DER)....67
Figura 57 Mapa de metro (izq) e información sobre las estaciones (der)....68
Figura 58 Mapa de sitios de interés (izq) e información sobre ellos (der)....69
Figura 59 Mapa EMT (IZQ) e información sobre las estaciones (DER)....70
Figura 60 Edificios disponibles en Realidad aumentada....71
Figura 61 Actividad de realidad aumentada....72
Figura 62 Layout del menu de localización....73
Figura 63 Interfaz de mapa....74
Figura 64 Inicializacion de la actividad del mapa....74
Figura 65 Metodo que configura el mapa....75
Figura 66 Método que dibuja los marcadores en el mapa....75
Figura 67 Layout marcador edificio....76
Figura 68 Método que crea un bitmap de un View....76
Figura 69 Creación de un marcador con el icono generado....76
Figura 70 Fragment de opciones....77
Figura 71 pantalla de acerca de....78
Figura 72 Dialogo confirmacion de cambio de idioma....79
Figura 73 Cambio de idioma de la aplciacion....79
Figura 74 Pantalla de cerrar sesion (IZQ) y pantalla de opciones con sesión cerrada (DER)....80
Figura 75 Generar item de sesion....81
Figura 76 Comprobación de alerta....81
Figura 77 Proceso de activacion de alarmas....82
Figura 78 Cancelación de la alarma del calendario....82
Figura 79 Pregunta 1 de pruebas de satisfacción....85
Figura 80 Pregunta 2 de pruebas de satisfacción....85
Figura 81 Pregunta 3 de pruebas de satisfacción....86
Figura 82 Pregunta 4 de pruebas de satisfacción....87
Figura 83 Pregunta 5 de pruebas de satisfacción....88
Figura 84 Obtencion de una clave api mediante SHA1....96
Figura 85 Introduccion de apikey en el manifest....96

Tablas

Tabla 1 Comparación de funcionalidades entre aplicaciones 21
Tabla 2 Dispositivos utilizados en las pruebas 83
Tabla 3 Resultados pruebas de respuesta 84

1. Introducción

1.1 Motivación

En los últimos años el mundo de los Smartphone y tabletas se ha mostrado dominante frente al mundo de los ordenadores personales [8]. Este crecimiento ha permitido a sus usuarios disfrutar de contenido al momento y desde prácticamente cualquier parte.

Teniendo en cuenta este crecimiento de consumo de contenido desde dispositivos móviles, empresas de éxito de prensa escrita [4] o editoriales de libros [5] permiten a los usuarios tener el contenido que siempre se ha vendido en papel en sus dispositivos móviles.

De misma manera, la Universitat Politècnica de Valencia es consciente de esta situación y ha desarrollado aplicaciones móviles para el acceso a la información [6].

Si bien las aplicaciones desarrolladas hasta la fecha cumplen su función para los alumnos de la UPV, estas aplicaciones no son específicas para cada tipo de alumno si no que son más generales.

En este contexto, se veía necesario desarrollar una aplicación que permita localizar, integrar y hacer más cómoda la estancia en la universidad de los alumnos que más lo necesitan: los alumnos de intercambio.

Para este caso la UPV entrega una guía para los estudiantes de intercambio bastante completa, que si nos situamos en el contexto del crecimiento del consumo de contenido desde dispositivos móviles, tendría sentido adaptar a estos dispositivos.

Esta es la principal motivación para realizar la aplicación que se desarrolla en este proyecto. Al realizar esta aplicación se obtienen unas ventajas muy claras sobre la versión de la guía en papel: al utilizar esta aplicación como guía de intercambio se eliminaría el coste de la impresión de las guías que se entregan en papel; Al ser desarrollada en dispositivos móviles se puede ampliar la información estática en papel mediante el uso de información dinámica como puede ser mapas o características sociales.

1.2 Estructura del documento

Para facilitar la lectura de la memoria, la estructura del resto del documento se detalla a continuación:

1. Introducción:
Se realiza la introducción describiendo cual es la motivación del proyecto, sobre que trata, cual es el problema que se desea resolver y un breve desglose del resto del documento.

2. Android:

Se realiza una breve explicación básica sobre el sistema operativo Android para el cual se desarrolla este proyecto.

3. Estado del arte:
Se presenta una panorámica general sobre el estado actual de la cuestión que se trata en este proyecto, es decir, una base sobre la cual este trabajo se sustenta.

4. Objetivos:
Breve recopilación, a título general y específico, de los objetivos seguidos para este Trabajo de Fin de Grado.

5. Especificación:
En esta sección se detalla el estudio precio a la implementación de la aplicación y la planificación seguida.

6. Implementación:
Se describe el proceso de implementación y los aspectos técnicos de esta.

7. Pruebas:
Se enumeran las diversas pruebas que se han realizado para comprobar que la aplicación cumple los requisitos establecidos en la fase de especificación.

8. Conclusiones:
En esta sección se detallan las conclusiones obtenidas tras finalizar el proyecto.

9. Mejoras futuras:
A modo de continuación del apartado anterior y con fin de que este proyecto sirva como base a futuras extensiones que se le puedan realizar, se detallan posibles mejoras sobre el mismo.

2. Android

Para entender la aplicación desarrollada en este proyecto se necesita hacer una explicación sobre el sistema operativo Android y como funciona.

Android es una plataforma software con el objetivo de abstraer el hardware y facilitar el desarrollo de aplicaciones para dispositivos. Inicialmente fue una propuesta del OHA (Open Handset Alliance [9]) pero su expansión ha propiciado un uso masivo de los servicios propietarios de Google.

2.1 Estructura del documento

Android comenzó como una pequeña empresa fundada en 2003 por Andy Rubin, Rich Miner, Nick Sears y Chris White, todos ellos trabajadores de otras importantes empresas tecnologicas.

En julio de 2005 Google adquirió la pequeña compañía con la intención de entrar en el mercado de la telefonía móvil. Dicha empresa se llamaba Android Inc. y fue fundada en Palo Alto (California).

Una vez en Google, el equipo que lideraba Rubin desarrolló una plataforma basada en el núcleo de Linux para dispositivos móviles, con el objetivo de ofrecer un sistema flexible y actualizable. En ese momento Google ya había comenzado a mantener relaciones entre fabricantes hardware y software, abriendo su posible cooperación a operadores móviles.

En 2007 "InformationWeek" difundió un estudio reportando que Google había solicitado diverstas patentes dentro de la telefonía móvil [10] y fue ese mismo año cuando se fundó el consorcio Open Handset Alliance, y se estrenó la primera plataforma Android construida sobre el núcleo 2.6 de Linux.

Fue finalmente en septiembre de 2008 cuando se lanzó la versión 1.0 de Android integrada en el teléfono móvil "HTC Dream".

2.2 Versiones

A lo largo de la historia de Android se han publicada diversas versiones del sistema operativo. Para realizar este proyecto se ha tenido en consideración el porcentaje de uso de las diferentes versiones para establecer una versión mínima desde la cual puedas ejecutar la aplicación. Como podemos observar en la Figura 1 las versiones predominantes en el mercado son las mayores de la 4.0 que ocupan un 85.7%

mientras que versiones inferiores obtienen un 14.3 % de cuota de dispositivos Android [3].

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	13.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	10.6%
4.1.x	Jelly Bean	16	26.5%
4.2.x		17	19.8%
4.3		18	7.9%
4.4	KitKat	19	20.9%

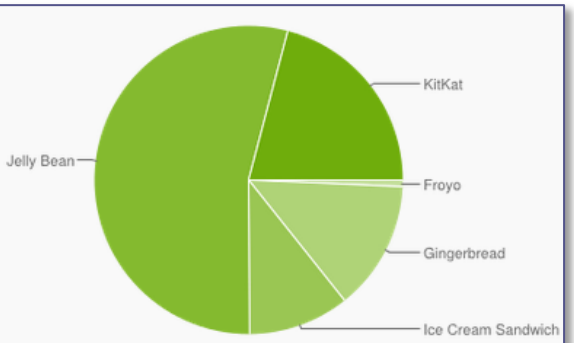

Figura 1 Versiones de Android a fecha de 12/08/2014

Es por esta razón que se ha optado por desarrollar la aplicación para dispositivos que usen una versión del sistema operativo mayor que la 4.0.

Las versiones más nuevas tienen distintas ventajas frente a las más antiguas:

- Mayor fluidez en la interfaz
- Mejor rendimiento de batería
- El SDK para esta versión incorpora nuevos elementos gráficos que realizan acciones más novedosas

Si se hubiese optado por aumentar la compatibilidad de dispositivos hasta la versión 2.3 Gingerbread, se hubieran desechado algunas funcionalidades y aspectos gráficos que se detallaban en los objetivos del proyecto. Por este motivo, se ha elegido disminuir la compatibilidad pero aumentar las funcionalidades y el rendimiento.

2.3 Arquitectura

Android, como todo sistema operativo, hace uso de una arquitectura que utiliza para su correcto funcionamiento.

Como se muestra en la Figura 2, la arquitectura de Android se compone de estas capas:

- Núcleo Linux
- Entorno en tiempo de ejecución
- Librerías nativas
- Entorno de la aplicación
- Aplicaciones

Figura 2 Arquitectura Android

Núcleo Linux

El núcleo de Android está formado por el sistema operativo Linux versión 2.6. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de controladores para dispositivos.

Esta capa del modelo actúa como capa de abstracción del hardware, por lo tanto, es la única que es dependiente del hardware.

Entorno en tiempo de ejecución

Está basado en el concepto de máquina virtual utilizado en Java pero dadas las limitaciones de los Android, Google tomó la decisión de crear una nueva, la máquina virtual Dalvik, que respondiera mejor a estas limitaciones.
La máquina virtual Dalvik esta optimizada para ahorrar la máxima memoria posible trabajando con registros y delegando operaciones al núcleo.
El entorno en tiempo de ejecución también contiene la mayoría de las librerías disponibles en el lenguaje Java.

Librerías nativas

Incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android que están compiladas en código nativo del procesador.

Entorno de aplicación

Proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones lo que permite a los desarrolladores implementar en sus aplicaciones funcionalidades como ubicación, notificaciones y el uso de todos los sensores.
El entorno de la aplicación contiene servicios que permiten a las aplicaciones ser ejecutadas y mostradas a los usuarios además de darle funcionalidades extra.

Los servicios más importantes que incluye son:

- **Views**: extenso conjunto de vistas, (parte visual de los componentes).
- **Resource Manager**: proporciona acceso a recursos que no son en código. Cualquier recurso (imagen, audio,…) que no se encuentre dentro de la aplicación necesita llamar a este manejador.
- **Activity Manager**: maneja el ciclo de vida de las aplicaciones y proporciona un sistema de navegación entre ellas.
- **Notification Manager**: permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.
- **Content Providers**: mecanismo sencillo para acceder a datos de otras aplicaciones (como los contactos).

Aplicaciones

Este nivel está formado por el conjunto de aplicaciones instaladas en una máquina Android. Todas las aplicaciones han de correr en la máquina virtual Dalvik para garantizar la seguridad del sistema.

Normalmente las aplicaciones Android están escritas en Java pero también existe otra opción consistente en desarrollar las aplicaciones utilizando C/C++.

3. Estado del arte

Este capítulo sirve como base teórica sobre la que se sustenta este Proyecto de Fin de Grado, y en el que se ofrece una visión general del área en el que se enmarca el mismo.

Se comienza hablando sobre los sistemas operativos existentes para móviles y su comparación. Después se analizarán distintas aplicaciones similares que hay en la tienda de aplicaciones que se asemejen a la que se desarrolla en este TFG. Finalmente se detallaran las herramientas utilizadas para el desarrollo de este proyecto.

3.1 Sistemas operativos móviles

Existen diferentes sistemas operativos móviles pero sin duda el más utilizado universalmente y en España es Android. Android es un sistema operativo con móviles asequible que permite tener cuota amplia de mercado. Como podemos apreciar en la Figura 3 [7] Android está por delante de competidores como iOS o Windows Phone.

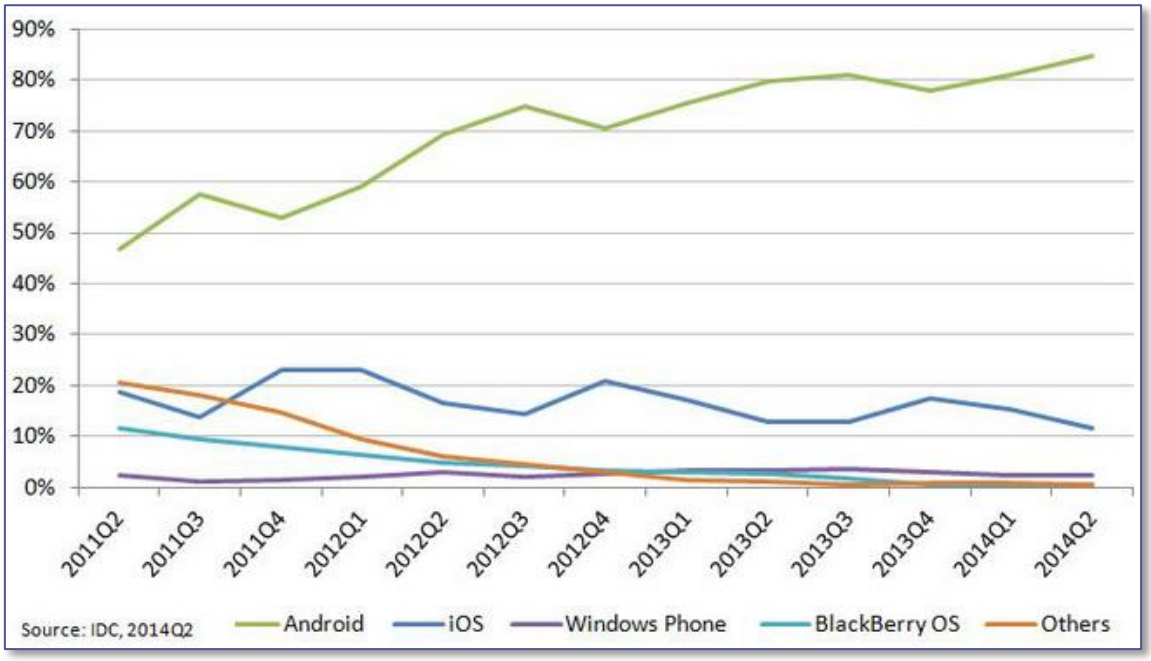

Figura 3 Cuota de mercado de sistemas operativos móviles

Como podemos apreciar, desde 2011 Android se situó como sistema operativo móvil más usado. Esta cuota no ha hecho sino que crecer más y el sistema operativo del Androide ha conseguido distanciarse hoy de sus competidores hasta el 70% de cuota de mercado.

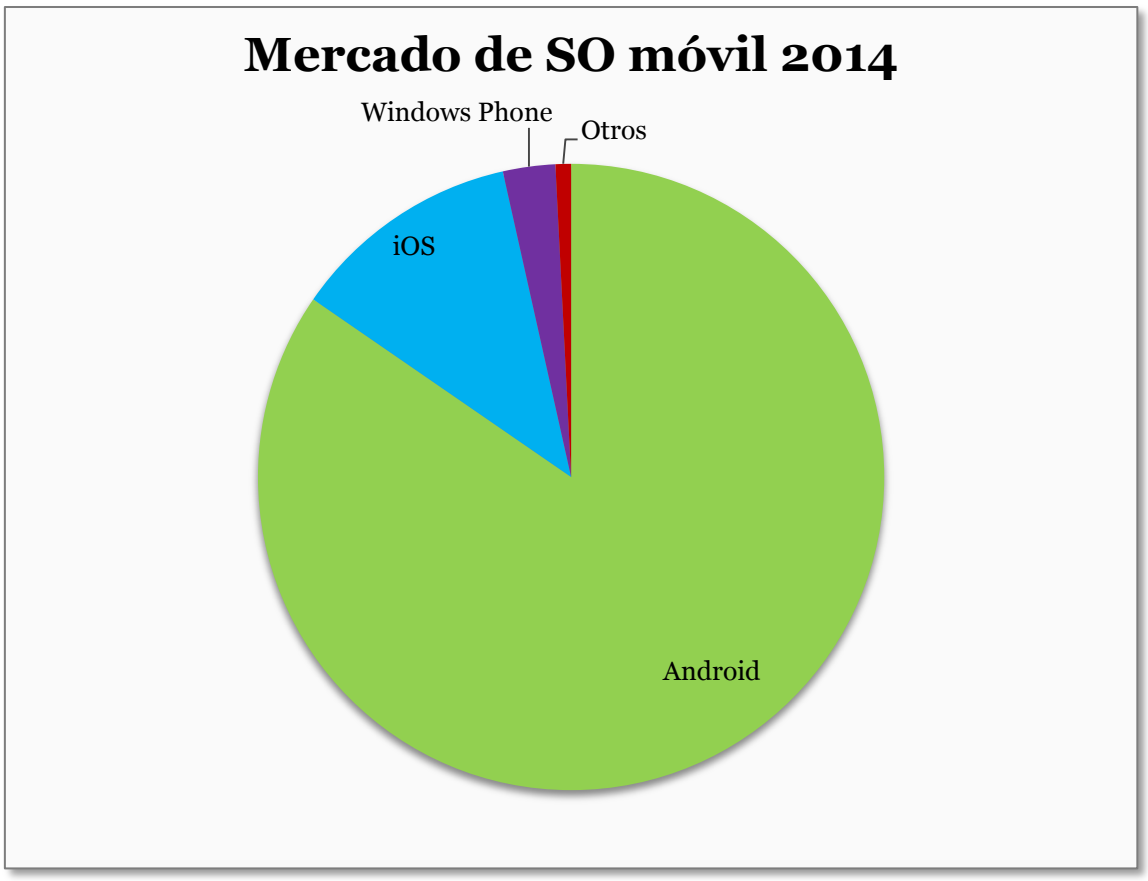

Figura 4 Mercado actual de sistemas operativos móviles

La Figura 4 muestra el estado actual de los sistemas operativos móviles. Se puede apreciar que Android supone casi el 85% del mercado mientras que los demás sistemas operativos conformarían un 15% donde iOS supone un 9.6% de cuota. Este predominio de Android sobre otros sistemas operativos moviles se basa en el hecho de que Android se ejecuta en muchos más móviles que la competencia ya que es un sistema de código abierto, libre de implementar por cualquier compañía de telefonía. iOS y Windows Phone sin embargo, son SOs privativos y solo se producen moviles con este software desde Apple y Microsoft.

3.2 Aplicaciones similares

Ya que, como hemos dicho antes, las aplicaciones móviles permiten que la información de las instituciones esté al alcance de todos, casi la totalidad de las universidades disponen de una aplicación móvil. Con solo buscar aplicaciones académicas en Google Play, se puede observar que hay muchas aplicaciones que podrían considerarse del mismo tipo que la que desarrollamos en este proyecto.

Instalamos diversas aplicaciones que podrían entrar en el mismo ámbito que esta aplicación. Vamos a proceder a analizar cada aplicación una por una:

Figura 5 Aplicación UPV

UPV – Universitat Politècnica de València.

Esta es la aplicación más parecida que se ha encontrado debido a que se aplica a la misma universidad.

Tiene cosas en común con la aplicación como los calendarios, los mapas de la universidad y la realidad aumentada. Sin embargo, le faltan muchas funcionalidades destinadas a alumnos de intercambio como pueden ser el traductor y los mapas de la ciudad.

Tampoco soporta el idioma inglés, algo necesario en una aplicación destinada a alumnos extranjeros. Por último, no se dispone de ningún método de comunicación entre alumnos.

Figura 6 Estudia UJA

Estudia UJA– Universidad de Jaén.

Welcome Incoming se basa en la manera de mostrar la información que tiene esta aplicación, pero también podemos ver que el diseño no es del todo adecuado y fácil para estudiantes que sean ajenos a la universidad.

La aplicación en general solo dispone de información estática y ninguna funcionalidad que permita al alumno acceder a su información relativa a la universidad como calendarios o exámenes.

Otro inconveniente es su fiabilidad. Falla con frecuencia y se reinicia la aplicación como se puede observar en la Figura 7.

Figura 7 Fallo en la aplicación estudia UJA

Figura 8 Universidad de Valencia

Universidad de Valencia – Universidad de Valencia.

Aplicación bastante completa que contiene muchas de las funcionalidades que se desarrollan en este proyecto.

La principal diferencia con esta aplicación es el público al cual va dirigida la aplicación. El público al que va dirigida la aplicación es a los estudiantes de la UV, y no a los estudiantes de intercambio. Si bien no está dirigida a este público, soporta el idioma ingles lo que puede resultar en una ayuda para este tipo de alumnos.

Se puede acceder a la intranet de la universidad dando así acceso a calendario y otras funcionalidades solo disponible para alumnos registrados.

No dispone de ningún tipo de herramienta de localización, por lo que un alumno de intercambio no se podrá localizar con facilidad dentro del recinto de la universidad. Esta funcionalidad es básica para una aplicación de estas características.Tampoco contiene una herramienta de comunicación entre alumnos ni de traducción.

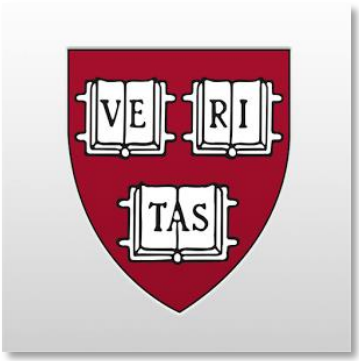

Figura 9 Harvard mobile

Harvard Mobile–Harvard University.

Nos basamos en la localización y la información que mostrar en los mapas.

Otra vez más, el diseño no parece el más adecuado buscando la simplicidad y la comodidad.

La aplicación no permite acceder a la intranet y solo dispone de información estática útil.

Se echa en falta algún tipo de conexión con los calendarios o información académica del alumno.

Figura 10 University of phoenix mobile

University of Phoenix Mobile– University of Phoenix.

Esta aplicación contiene todos los datos estáticos que dispone Welcome Incoming, conexión a la intranet de la universidad y contiene calendario y foro integrado por el que preguntar dudas a profesores y alumnos. Es la única aplicación estudiada en introducir un medio de comunicación entre alumnos.

Figura 11 Interfaz de University of Phoenix

En la Tabla 1 muestra una tabla comparativa en la que se enumeran las funcionalidades que la aplicación desarrollada en este proyecto contiene y si las aplicaciones que se han probado lo cumplen.

	UPV	UJA	UV	Harvard	Phoenix
Conexión a intranet					
Mapas de la universidad					
Mapas de la ciudad					
Herramienta de traducción					
Orientado a alumnos de intercambio					
Calendario integrado					
Foro para los alumnos					
Realidad aumentada					
Alertas para las clases					
Información estática útil					
Soporte de diferentes idiomas incluido Inglés					

Tabla 1 Comparación de funcionalidades entre aplicaciones

Es cierto que existen muchas aplicaciones de universidades parecidas, pero en todas estas aplicaciones se echa en falta una funcionalidad crucial: la integración de estudiantes de intercambio en la universidad. Por otra parte, Welcome Incoming dispone de unas funcionalidades que no se han visto en ninguna aplicación como la traducción o mapas de la ciudad.

El foro, que solo se ha visto algo parecido en la aplicación de la universidad de Phoenix, permite a los usuarios de la aplicación comunicarse y la traducción facilita la comunicación entre estudiantes de distintos países e idiomas.

Otras funcionalidades como el acceso interno a la intranet de la UPV, y datos internos de la web no está disponible al ser desarrollada de forma externa. Todos los calendarios se han extraído de los archivos disponibles para todos los alumnos desde la web.

4. Objetivos

El objetivo del proyecto es el desarrollo de una aplicación móvil para la plataforma Android que se distribuya mediante la tienda de aplicaciones Google Play de manera gratuita. La aplicación está destinada para estudiantes de intercambio de la UPV, tanto para alumnos extranjeros (ERASMUS y PROMOE) como para alumnos que provienen de provincias españolas (SICUE).

Los objetivos específicos de este proyecto son los siguientes:

- Mostrar de forma clara y concisa la información extraída de la Guía de Intercambio.
- Mantener localizado en todo momento al usuario mediante ubicación en el mapa y mediante el uso de la realidad aumentada.
- Disponer de información académica actualizada con las escuelas de la UPV con su correspondiente información y las asignaturas que se imparten en cada una de estas escuelas.
- Contener planos e información útil de Valencia y de la UPV.
- Notificar mediante el sistema de notificaciones de Android cuando una clase vaya a comenzar para mantener al alumno localizado y alerta con sus obligaciones en la UPV.
- Mantener un rendimiento alto y un tiempo de espera de menos de medio segundo para funcionalidades que no requieran obtener información. En caso de funcionalidades que requieran mayor tiempo, informar al usuario de la obtención de recursos.
- Cumplir con los requisitos de diseño tanto del uso de la marca de la UPV como de Android y sus patrones de diseño de interfaces.
- Hacer que el uso de la aplicación sea *responsive* y cómodo en todo momento.

Ya que estos objetivos los ha impuesto el cliente, la aplicación deberá cumplir cada uno de estos objetivos si se quiere tener éxito en el desarrollo.

5. Especificación

En este apartado se realiza en análisis de la aplicación, casos de uso, bocetos de interfaz y modelo de datos. Este proceso pre-implementación permite a los desarrolladores tener una base sólida donde construir su aplicación.

Las herramientas necesarias para realizar la especificación de esta aplicación son:

- Bouml para el diseño de los casos de uso [11].
- NinjaMock para la creación de los bocetos de la interfaz [14].
- DBVisualizer para crear el modelo de datos [13].

Figura 12 Herramientas de la especificación

5.1 Análisis

Esta fase se considera sumamente importante en el proceso de desarrollo de un sistema software, ya que sirve de base para comprender y concretar el sistema, y conocer los elementos que conforman el contexto del problema. Los productos del análisis a desarrollar serán los siguientes: Requisitos de la aplicación (requisitos funcionales y requisitos no funcionales) y casos de uso. Además se detallan los bocetos de interfaz y la planificación del desarrollo del proyecto.

5.1.1 Requisitos de la aplicación

En este grupo de requisitos se encuentran los definidos por el usuario, es decir, aquellas características demandadas por el cliente que tienen como objetivo plantear el problema y delimitar qué es lo que ha de cumplir la aplicación.

Requisitos funcionales

Los requisitos funcionales de la aplicación son:

- Guía de intercambio:
  - **Basarse en la 'Guía de Intercambio'**: Este es el principal requisito de la aplicación, ya que fue pensada como un sustituto de la Guía de Intercambio de la UPV. De ella se obtiene la información sobre transportes, Valencia, UPV y teléfonos de interés.

- Mapas:
  - **Mostrar un mapa de la UPV**: La aplicación debe mostrar un mapa con los edificios de la universidad, mostrando información de cada uno de estos edificios.
  - **Mostrar un mapa de Valencia**: A parte de la información correspondiente a la universidad, se mostrará también información relativa a la ciudad de Valencia. Para este fin se deben de mostrar diferentes mapas:
    - **Mapa de Valenbisi**: En este mapa se muestran las estaciones de Valenbisi e información relacionada con esta, tal como las plazas totales y las bicis disponibles.
    - **Mapa de EMT**: Se muestran las estaciones de bus de EMT de Valencia.
    - **Mapa de metro**: El mapa muestra las estaciones de metro de Valencia.
    - **Mapa de sitios de interés**: Los sitios más importantes de Valencia se muestran en el mapa.
  - **Localización**: Si el dispositivo es capaz de usar los servicios de localización se deberá mostrar en el mapa la ubicación actual del alumno.

- Realidad aumentada:
  - **Utilizar la realidad aumentada para localizar al alumno dentro del campus**. Esta funcionalidad permitirá usar la realidad aumentada para localizar edificios de forma interactiva.

- Información estática:
  - **Mostrar información académica:** La aplicación debe mostrar información actualizada de las asignaturas y escuelas de la UPV.
  - **Mostrar información sobre los transportes más utilizados de Valencia**: Se mostrarán información sobre tren, metro, bus, avión y taxi.

- Contacto:
  - Mostrar noticias de la OPII:
  - Mostrar enlaces a las principales redes sociales de la UPV:
  - Mostrar teléfonos importantes de la UPV:

- Otros:

  - Imagen de carga: Imagen necesaria para que el usuario sepa que la aplicación se está iniciando mientras se cargan los recursos necesarios.

Figura 13 Pantalla de carga

- Guardar preferencias: Se deberá guardar diversas preferencias dependiendo el usuario que utiliza la aplicación, para así mantener las configuraciones personalizadas de la aplicación.

- Localización y lenguaje: La aplicación dispondrá de dos idiomas que el usuario puede cambiar a su gusto: Español e Inglés

Requisitos no funcionales

Los requisitos no funcionales para la aplicación, es decir, los que no especifican el comportamiento del sistema, son:

- **Rendimiento**: La aplicación debe desempeñar su función de una manera fluida. Se debe buscar la experiencia de uso más agradable para el usuario.

- **Disponibilidad**: La aplicación debe estar disponible mediante la Google Play. La aplicación debe funcionar sin conexión, ya que los alumnos que vienen a estudiar no tienen conexión a internet inmediata.

- **Accesibilidad**: La aplicación debe ser legible y tiene que seguir los patrones de accesibilidad de Google.

- **Usabilidad**: Cualquier alumno extranjero debe ser capaz de utilizar la aplicación y acceder a toda la funcionalidad sin ningún tipo de restricción.

- **Estabilidad**: La aplicación debe ser capaz de manejar los errores ocurridos durante la ejecución de la misma y avisando a este de la naturaleza del error.

- **Mantenimiento**: La aplicación debe ser mantenida y actualizada, dando posibilidad a mejorar el rendimiento y la usabilidad en cualquier momento.

- **Interfaz**: Clara y concisa. No debe dar lugar a la confusión del usuario y debe seguir los estándares de diseño de interfaces de Google.

- **Integración**: La aplicación debe de integrarse con todo el sistema operativo de Android. Hacer uso de las aplicaciones nativas si se necesita y mantener un diseño acorde al sistema.

- **Optimización**: El consumo de batería y de datos debe ser adecuado, y nunca dejar procesos sueltos que consuman memoria y batería. El tiempo de ejecución debe ser mínimo, para mejorar los tiempos de respuesta y la experiencia de uso del usuario.

5.2 Casos de uso

El diagrama de casos de uso es una de las herramientas destinadas al análisis de un proyecto software. Es una técnica de escenarios en UML que describe la interacción entre actores y sistema. Debido a que en este TFG el único actor del sistema seria el usuario que utiliza la aplicación, la complejidad es baja.

El conjunto de casos de usos que se describen a continuación son:

- Inicio de sesión
- Consultar noticias
- Consultar información estática
- Obtener eventos de calendario
- Aplicar filtros al calendario
- Consultar mapas
- Acceder a la realidad aumentada
- Activar o desactivar alarmas
- Cerrar sesión
- Cambiar idioma de la aplicación
- Consultar acerca de

Estas actividades suelen componerse de más de un caso de uso, pero en este caso se ha considerado solo como un caso de uso:

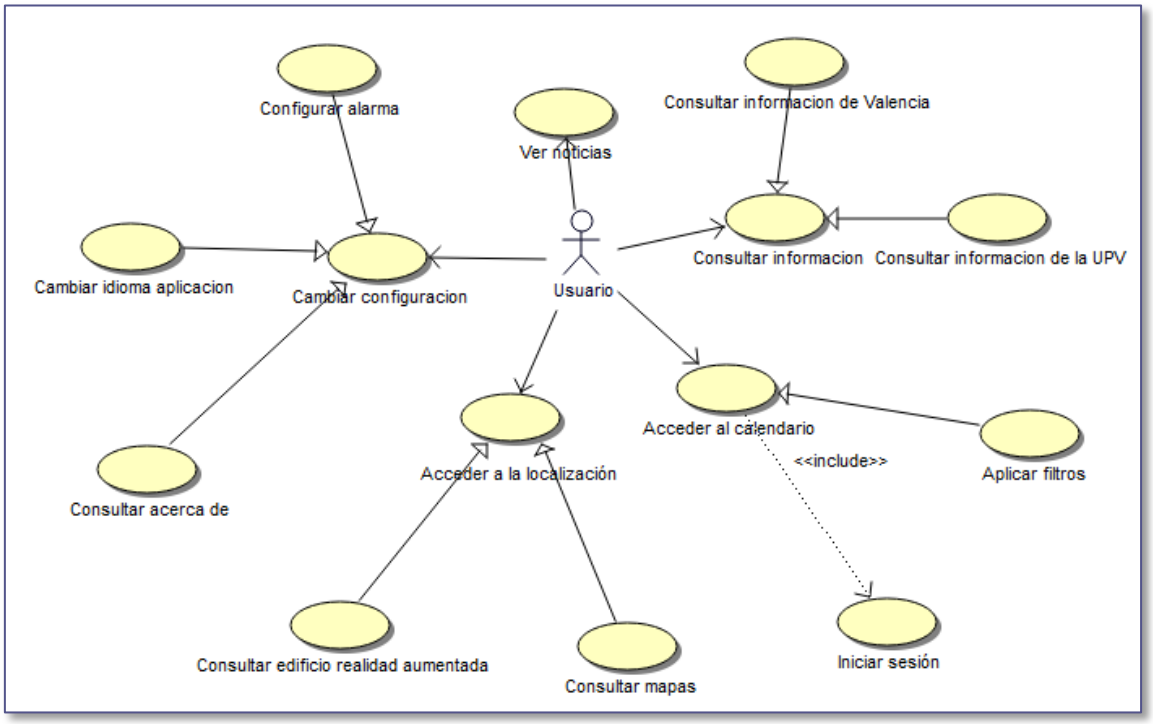

Figura 14 Casos de uso

Como se puede observar en la Figura 14, se contemplan todas las acciones que el usuario puede hacer en la aplicación. Desde ver las noticias hasta aplicar filtros a los eventos del calendario.

5.3 Bocetos de la interfaz

Para obtener una idea general sobre la aplicación a desarrollar y su versión final, se realizan bocetos de interfaz que permiten al cliente ver como se orienta su aplicación y su diseño.

Los bocetos que se muestran corresponden con la versión de la aplicación desarrollada en la asignatura SDM con posteriores modificaciones en la fase de especificación de este proyecto.

Algunos de los bocetos de la interfaz son:

Figura 15 Bocetos de interfaz del drawer e inicio

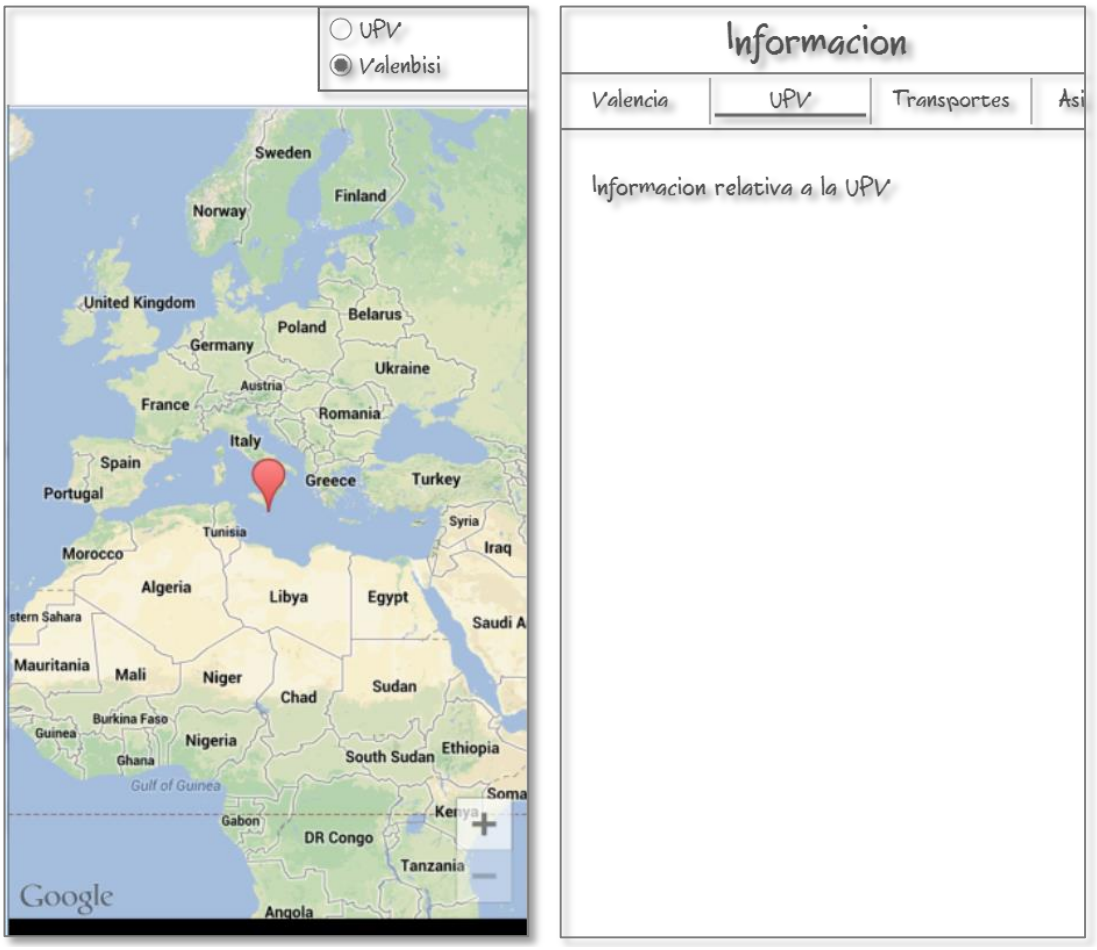

Figura 16 Bocetos de mapas e información

5.4 Modelo de datos

La aplicación desarrollada debería funcionar casi en su totalidad sin conexión a internet. Este requisito requería almacenar toda la información que contenía la aplicación localmente. Además de archivos XML que contuviesen la información, se pensó en incorporar una base de datos interna en el dispositivo móvil al instalar la aplicación para un fácil acceso a los datos.

Para la base de datos se creó un modelo relacional que soportaría todo el almacenamiento de la información. Se puede observar cómo se almacenan la información de los transportes, los eventos del calendario y los marcadores de los elementos del mapa.

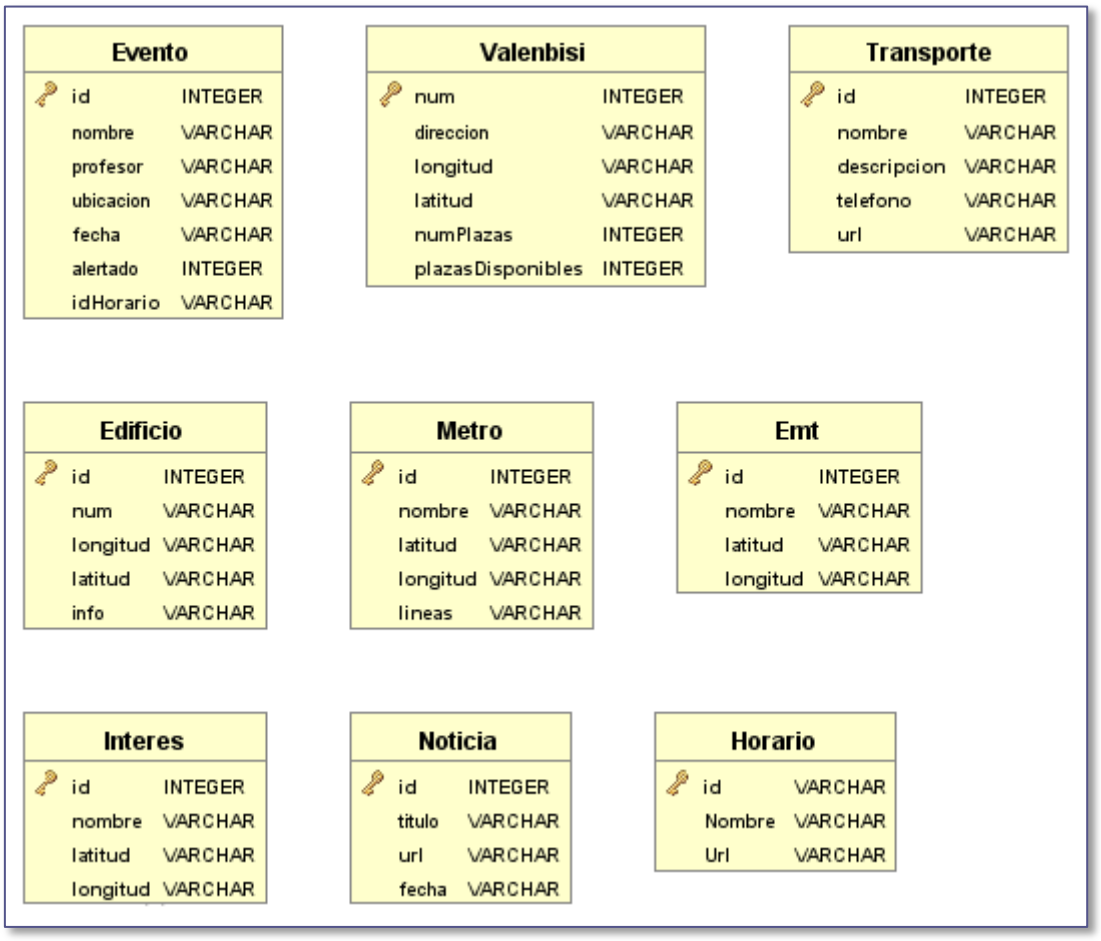

Figura 17 Modelo entidad relación de la base de datos

Como se puede observar, no se contempla ninguna relación entre las diferentes tablas de la base de datos debido a que no se requiere. Toda esta información está almacenada de forma individual y por tanto no necesita de conexión tabla a tabla.

5.5 Planificación

Desde el momento en que se define los requisitos y los casos de uso, el próximo proceso en el desarrollo del software es establecer una planificación de la implementación y pruebas.

Para la estructuración del proceso de implementación se ha utilizado un diagrama de Gantt. Un diagrama de Gantt es una herramienta muy útil que permite exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. En la Figura 18 se muestra el diagrama utilizado para este proyecto.

Como se puede apreciar, se ha dividido el trabajo en tres Sprint diferenciados:

- El primer sprint: Debido a que este proyecto es una continuación de otra aplicación desarrollada anteriormente, este sprint se basa en inspeccionar lo obtenido e implementar mejoras de funcionalidades y de interfaz a la aplicación existente.
- El segundo sprint: En este sprint se produce los cambios a la aplicación. Se incorporan las funcionalidades requeridas por el cliente y se sigue mejorando la interfaz.
- El sprint final: Este es el sprint de las pruebas y el acabado de la aplicación. Se corrigen errores y se añaden funcionalidades extra que se creen posibles de introducir.

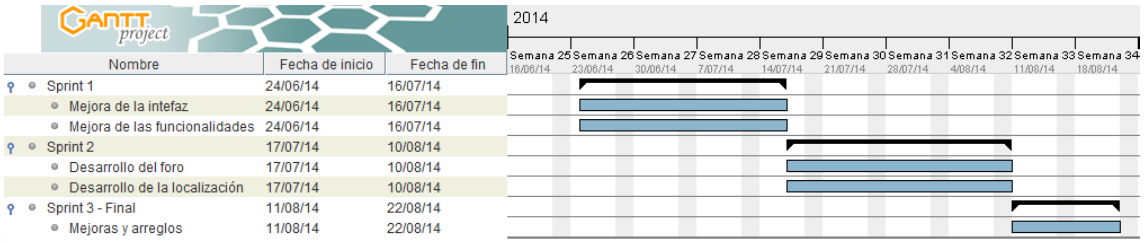

Figura 18 Diagrama de Gantt del proyecto

Cabe decir que el proyecto es una continuación de una aplicación desarrollada para la asignatura de SDM y que ya disponía de desarrollo anterior. Si bien eso ha facilitado el objetivo de este proyecto, también ha supuesto un mayor esfuerzo por mejorar las funcionalidades ya existentes. También ha entorpecido el desarrollo el hecho de que la anterior aplicación se había implementado por personas ajenas a este proyecto y mucho tiempo se ha dedicado a entender código y refactorizar.

6. Implementación

Se ha utilizado Java como lenguaje de programación y Android Studio como entorno de desarrollo debido a que es el actual IDE respaldado por Google y el que dispone de mejores herramientas para desarrollar aplicaciones para la plataforma Android.

Las herramientas necesarias para desarrollar utilizadas en esta aplicación son:

- Android Studio 0.8.2 [2].
- Samsung Galaxy S3, Samsung Galaxy Grand 2 con Android 4.3.
- Gimp 2.8 para el diseño de iconos y edición de imágenes utilizadas en la aplicación [13].

Para el desarrollo de la aplicación se ha tenido en cuenta que debe ser una aplicación intuitiva y fácil de usar. Para este objetivo se ha divido la aplicación en bloques de información:

- Inicio
- Calendario
- Información
- Traducción
- Localización
- Foro
- Opciones

Con el fin de conseguir una aplicación intuitiva se ha implementado el patrón de diseño con Navigation Drawer (Figura 19), un panel lateral que aconseja Google tener en todas las aplicaciones con muchos bloques de información.

El navigation drawer es un View o vista que actúa como menú para cambiar entre las distintos Fragments o pantallas de la aplicación. El drawer se oculta automáticamente cuando el usuario este usando la aplicación, por lo que permite a los Fragments ocupar el mayor tamaño de pantalla posible.

Como se puede apreciar en la Figura 20, cada elemento del drawer contiene un icono y un texto al lado correspondiente al bloque de funcionalidades. Estos ítems se han personalizado para poder tener diferentes elementos visuales en una misma fila.

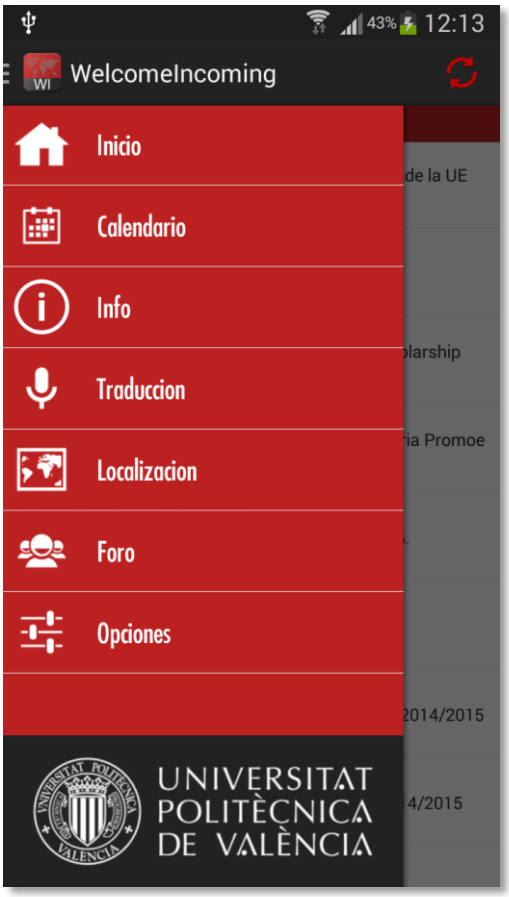

Figura 19 Navigation drawer

El drawer por defecto, dispone de un adaptador que te permite mostrar los elementos pero solo con texto, por lo tanto tuvimos que crear nuevas vistas personalizadas como se muestra en la Figura 20.

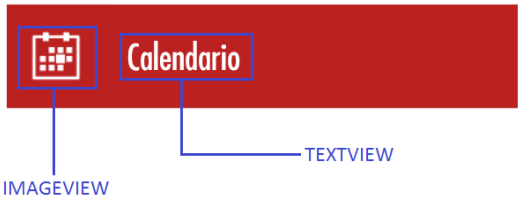

Figura 20 Elemento del drawer

Como hemos dicho antes, esta aplicación se basa en el patrón de diseño de Drawer + Fragments. Para comprender el

comportamiento principal de la aplicación es necesaria la definición de Fragment que nos proporciona el mismo Google Developers:

> *Un Fragment representa un comportamiento o una porción de interfaz de usuario en una actividad. Puede combinar múltiples fragmentos en una sola actividad para construir una interfaz de usuario multi-panel y reutilizar un fragmento en múltiples actividades.*
>
> *Se puede entender un fragment como una sección modular de una actividad, que tiene su propio ciclo de vida, recibe sus propios eventos de entrada, y que se pueden agregar o quitar mientras que la actividad está en marcha (algo así como una "sub actividad" que pueda reutilizar en diferentes actividades).*

Por tanto, todo fragment necesita una actividad y una vista donde implementarse (Contenedor fragment). Este contenedor solo puede mostrar un fragment a la vez, pudiendo cambiar en tiempo de ejecución el fragment que se muestra.

Al hacer clic en un ítem del drawer, el fragment principal de la aplicación se cambia por el correspondiente fragment creado para cada uno de los ítems del drawer.

Para entender mejor el comportamiento de la actividad principal de la aplicación 'Welcome Incoming' mostramos cuales serían los pasos para mostrar cualquier ítem del drawer:

1. **Diseño de actividad**: En la Figura 21 se muestra como seria la actividad sin ningún fragment creado. Cabe destacar que en esta imagen el drawer está creado pero oculto. La parte destacada en rosa es el contenedor de fragments donde se va a posicionar el fragment de cada elemento del drawer.

Figura 21 Diseño de la pantalla principal

2. **Apertura del drawer y pulsación sobre el ítem deseado**: Al deslizar de izquierda a derecha de la pantalla se abre el drawer (También podemos abrirlo haciendo clic sobre el icono en la parte superior izquierda de la activity). Cada elemento del drawer que vemos en la Figura 22 tiene un fragment creado y diseñado de forma diferente a cada uno de los demás bloques de información.
   Podemos ver en la imagen como el ítem "Inicio" esta pulsado, lo que conllevara que el fragment que se cargue en el contenedor de fragments sea el "Fragment_Inicio".

3. **Carga del fragment**: El 'Fragment_Inicio" contiene una lista (ListView) que muestra el RSS de las noticias de la Oficina internacional de la OPII. Este es el fragment que se muestra en el contenedor de fragment como podemos ver en la Figura 22. Esto es debido a que hemos pulsado sobre el ítem 'Inicio' del drawer.

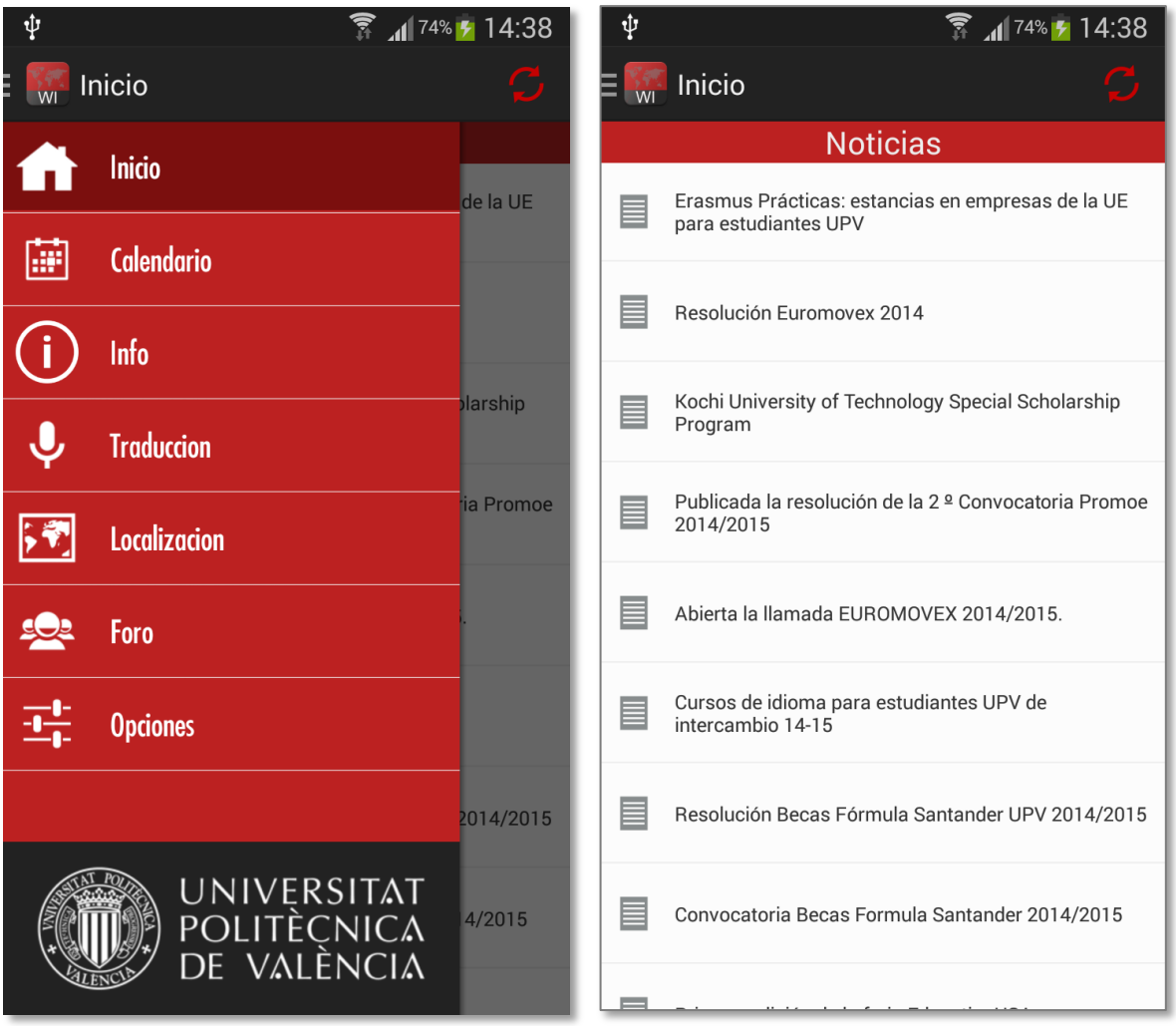

Figura 22 Pantalla de inicio

6.1 Inicio

Como hemos comentado anteriormente, la primera pantalla al iniciar la aplicación es 'Inicio'. El fragment inicio se carga automáticamente la primera vez que inicias WelcomeIncoming.

Al ser la primera vez que inicias la aplicación, se muestra una pantalla a modo de 'Tutorial' que contiene instrucciones del uso de la aplicación como podemos ver en la Figura 23.

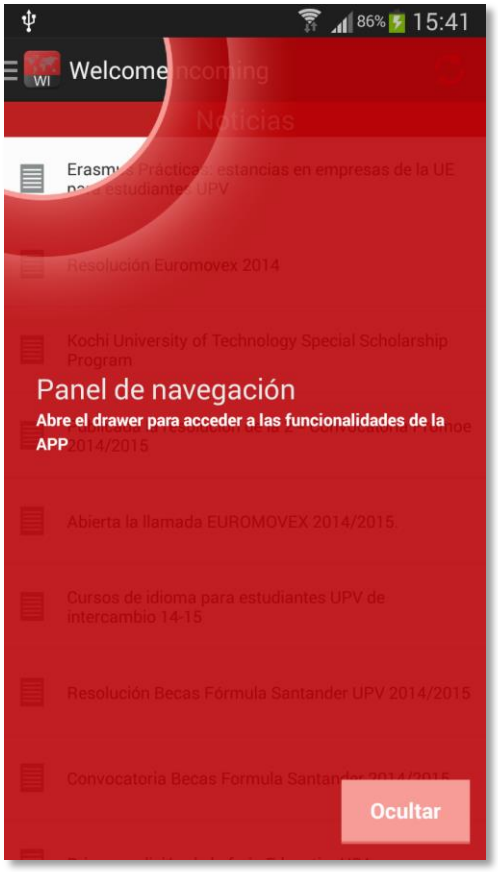

Figura 23 Pantalla tutorial

El fragment inicio contiene las últimas noticias de la OPII (Oficina internacional de Programas de Intercambio) obtenidas del RSS oficial. Al pulsar sobre cualquier noticia el usuario será redirigido hacia la página web de la noticia que ha pulsado.

También dispone de la funcionalidad de actualizar las noticias, para poder acceder a las noticias más recientes.

6.2 Calendario

El segundo ítem en el drawer es el Calendario. Este ítem contiene el horario de clases del usuario. Para poder obtener el horario de clases se tiene que acceder a la UPV y a su intranet. Para ello, si no estás registrado anteriormente, se mostrara una ventana de inicio de sesión. Si el inicio es correcto se mostraran los eventos del calendario.

6.2.1 Diseño de interfaces

Los eventos que se muestran en este fragment se descargan de la intranet de la UPV en formato iCAL y se convierten para que sean utilizables por la aplicación. Como la descarga de todos los eventos suele ser costosa, solo se realiza obligatoriamente la primera vez que accedes al calendario. Cualquier acceso posterior al calendario no necesitará de descarga de eventos, ya que estos se guardan en la base de datos interna de la aplicación.

Si por alguna razón los calendarios del usuario han sido cambiados, el alumno puede actualizar los eventos y volverlos a descargar dándole al icono de actualizar, situado en la parte superior derecha de la pantalla. Esto sobrescribirá los eventos y las alarmas que haya podido poner.

Los eventos del calendario se muestran en una lista (ListView) ordenados según fecha como se muestra en la Figura 24. Solo se muestran los eventos posteriores al día actual.

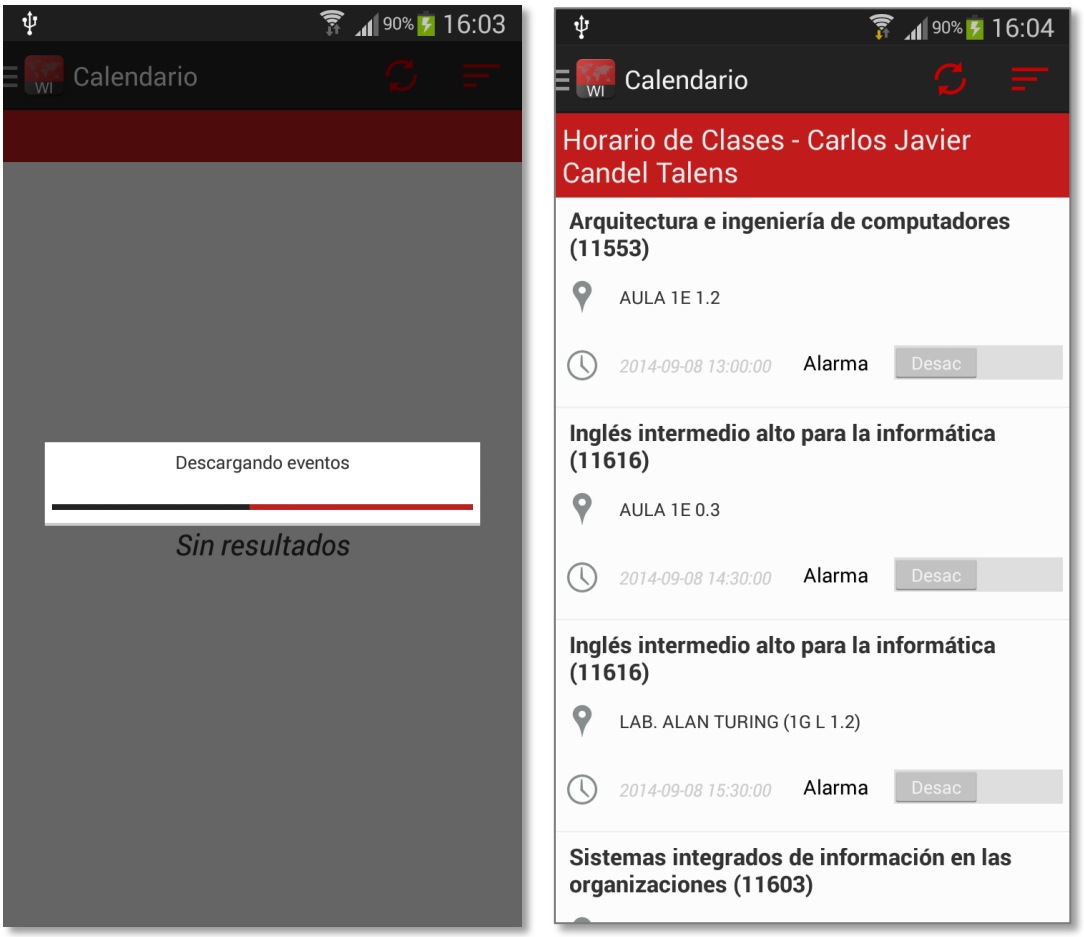

Figura 24 Carga de eventos del calendario

Evento

El calendario se compone de eventos. Estos eventos son mostrados en una lista (ListView) de una forma determinada. Ya que como hemos explicado anteriormente la lista por defecto de Android solo permite mostrar en sus elementos Strings, hemos optado por crear una vista para los elementos de la lista que permita mostrar más detalle acerca del evento que estamos visualizando.

Como podemos observar en la Figura 25, los elementos de la lista muestran la información del evento de forma estructurada y clara.

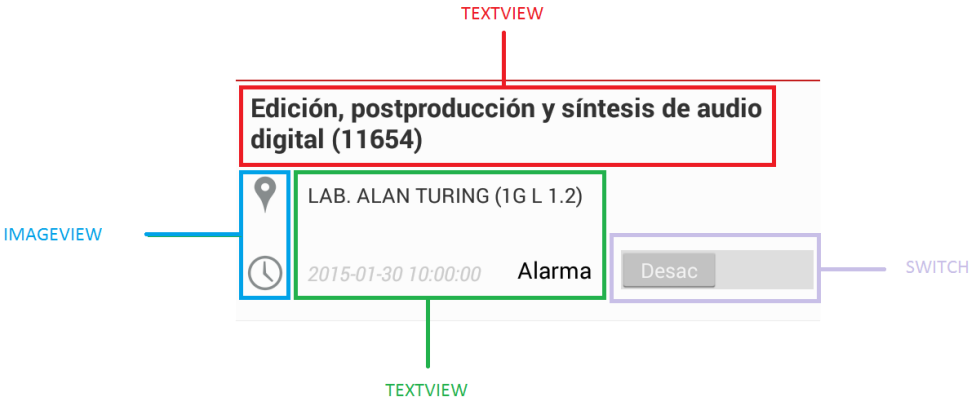

Figura 25 Vista personalizada del evento

Por una parte tenemos el título del evento, que corresponde al nombre de la asignatura.

En segundo lugar tenemos la información del lugar y la fecha del evento. Ambas informaciones vienen acompañadas de iconos que indican que es cada elemento para facilitar la lectura de eventos.

Por ultimo tenemos un Switch, es decir, un 'interruptor' que nos permitirá definir alarmas para los eventos. Cada uno de los eventos tendrá un interruptor ya que, en caso de querer poner alarma para más de un evento, se podrá desde la misma lista programar la notificación.

Alarmas

Una funcionalidad que distingue el calendario de Welcome Incoming del calendario que dispone la aplicación oficial de la UPV es que este calendario te permite definir alarmas para los eventos de modo que te avise cuando una clase comienza.

Al encender la alarma de un evento, se envía una notificación pasándole la fecha y hora a la que es el evento y el usuario recibirá la notificación 10 minutos antes de que el evento programado ocurra como se puede ver en la Figura 26.

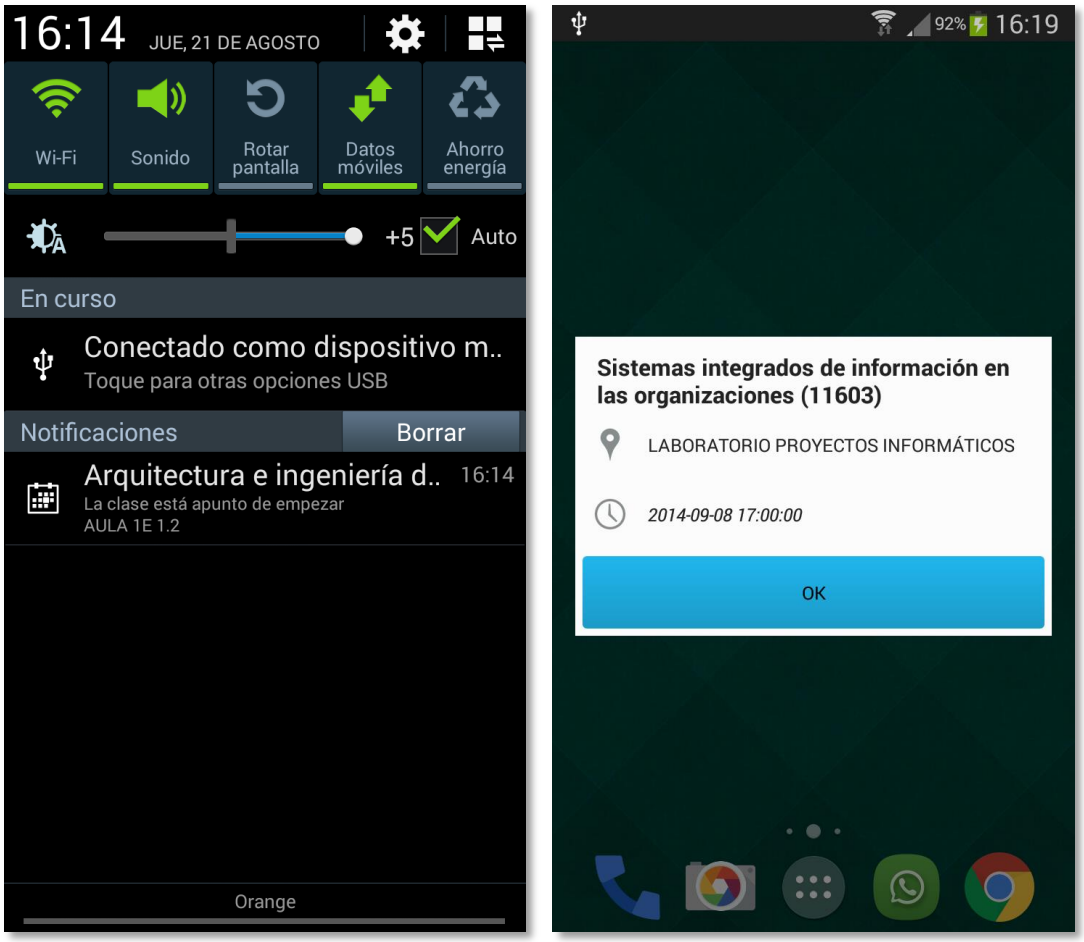

Figura 26 Notificacion de alarma (izq.) y informacion de evento (Der.)

La notificación muestra el nombre de la asignatura del evento además de un texto informando de que la clase va a comenzar y en qué edificio y sala se va a impartir esta. Si pulsamos sobre esta notificación se mostrara toda la información detallada sobre el evento como podemos apreciar en la Figura 26 (Derecha).

Filtros de eventos

Debido a que se muestran en una lista, buscar un evento que ocurrirá meses desde el día actual puede llegar a ser muy incómodo. Por este motivo se han añadido filtros para seleccionar eventos dependiendo de la fecha o de las asignaturas.

Filtro por asignaturas

Al filtrar por asignaturas, aparece una actividad en forma de dialogo que carga las asignaturas que tiene el usuario actualmente (Figura 27). Al seleccionar una asignatura y aceptar en la pantalla de eventos se mostraran los eventos de esa asignatura. Si desea quitar este filtro una vez encontrado el evento, solo tiene que pulsar en la opción quitar filtros que se muestra en el menú desplegable y volverán a aparecer los eventos tal y como se mostraban anteriormente.

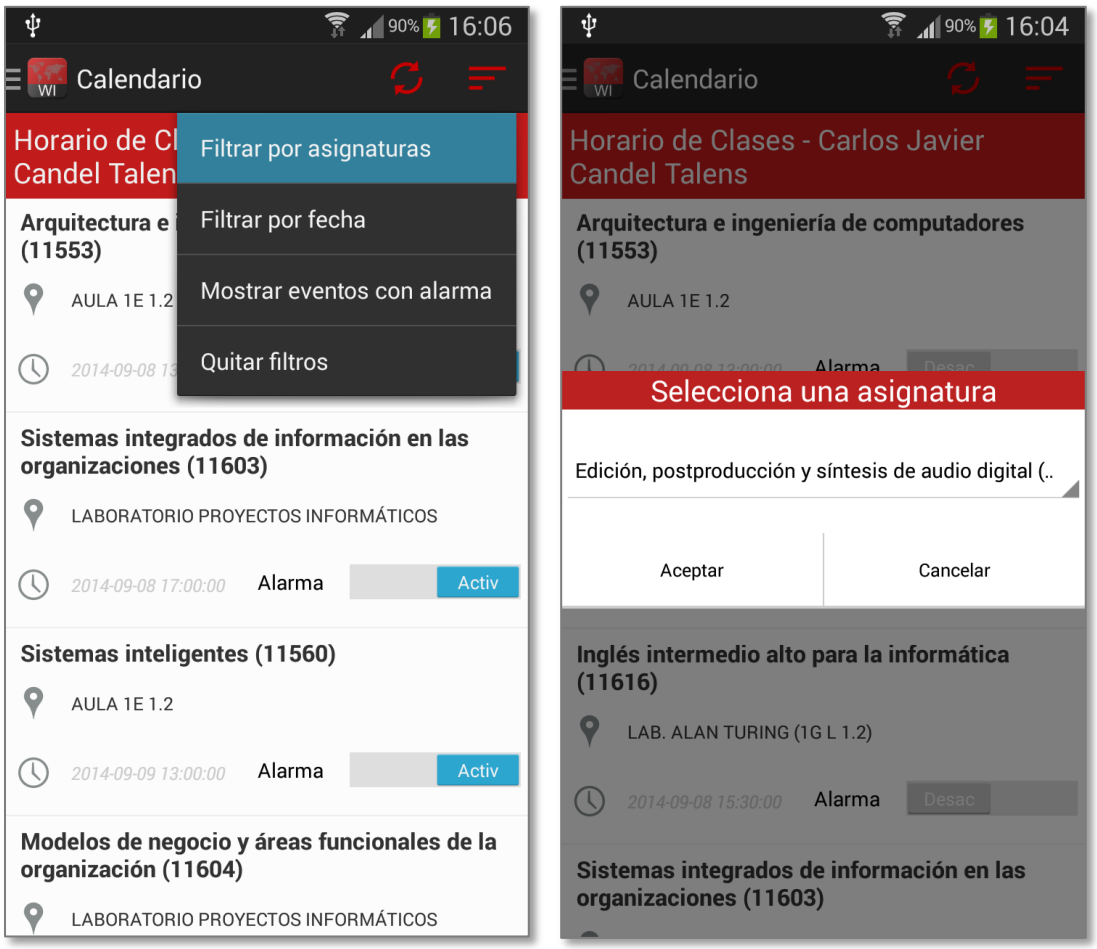

Figura 27 Filtro de eventos por asignatura

Filtro por fecha

Si un usuario de la aplicación intenta buscar los eventos de un día concreto en una lista de eventos larga, puede ofuscarse o incluso no encontrar ese día. Para evitar esto, se ha añadido otro filtro que se basa en buscar los eventos de un día en concreto.

El dialogo para filtrar fecha (Figura 28) contiene un DatePicker, un vista que te permite seleccionar día, mes y año. Al marcar la fecha deseada y pulsar el botón aceptar, la lista de eventos muestra los eventos de ese día si existen y si no mostrará una lista vacía informando que no hay eventos que correspondan con ese día.

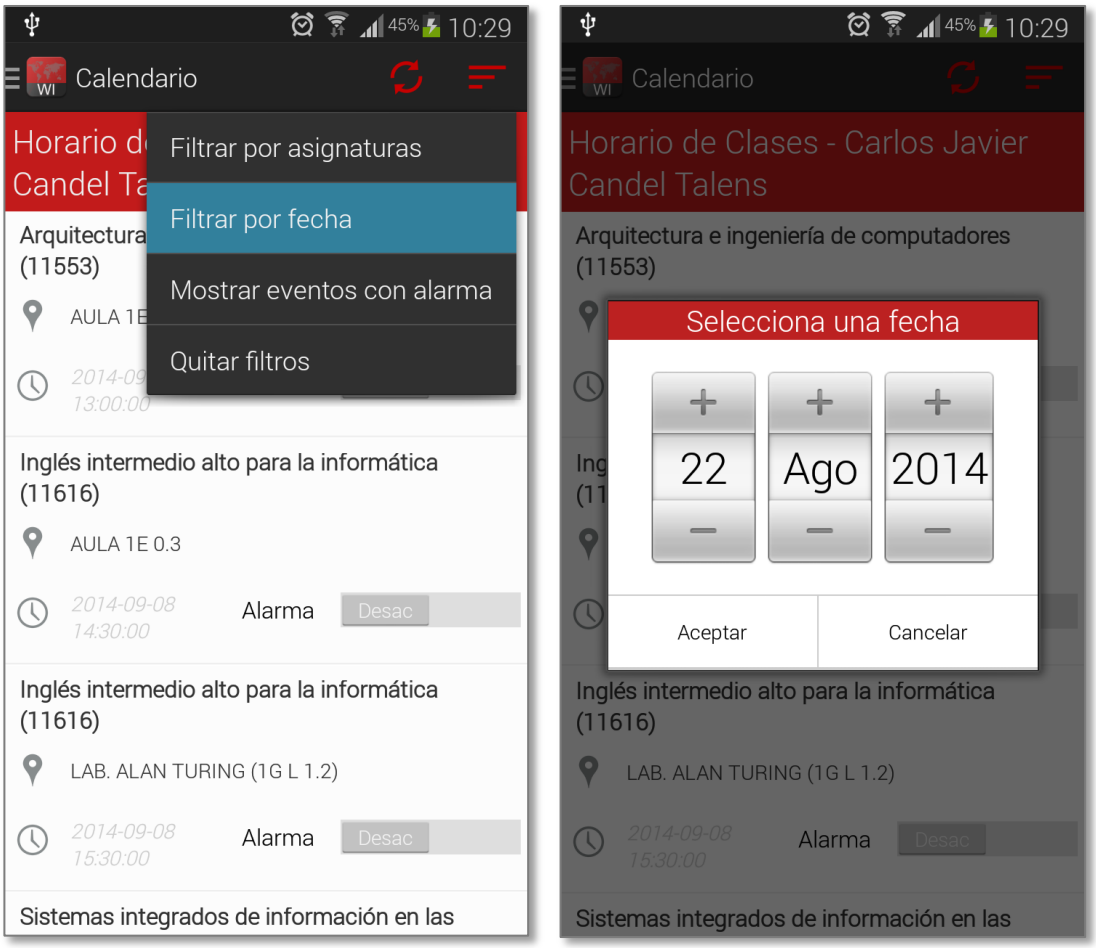

Figura 28 Filtro de eventos por Fecha

Filtro de eventos con alarma

Ya que se implementa la funcionalidad de aplicar alarmas a los eventos, es interesante introducir un filtro para mostrar únicamente los eventos que tienen alarma definida. Esta funcionalidad permite al usuario observar las alarmas que tiene definidas y poner anularlas. Como podemos ver en la Figura 29, los eventos con alarma se muestran al pulsar la opción 'Mostrar eventos con alarma' del menú desplegable.

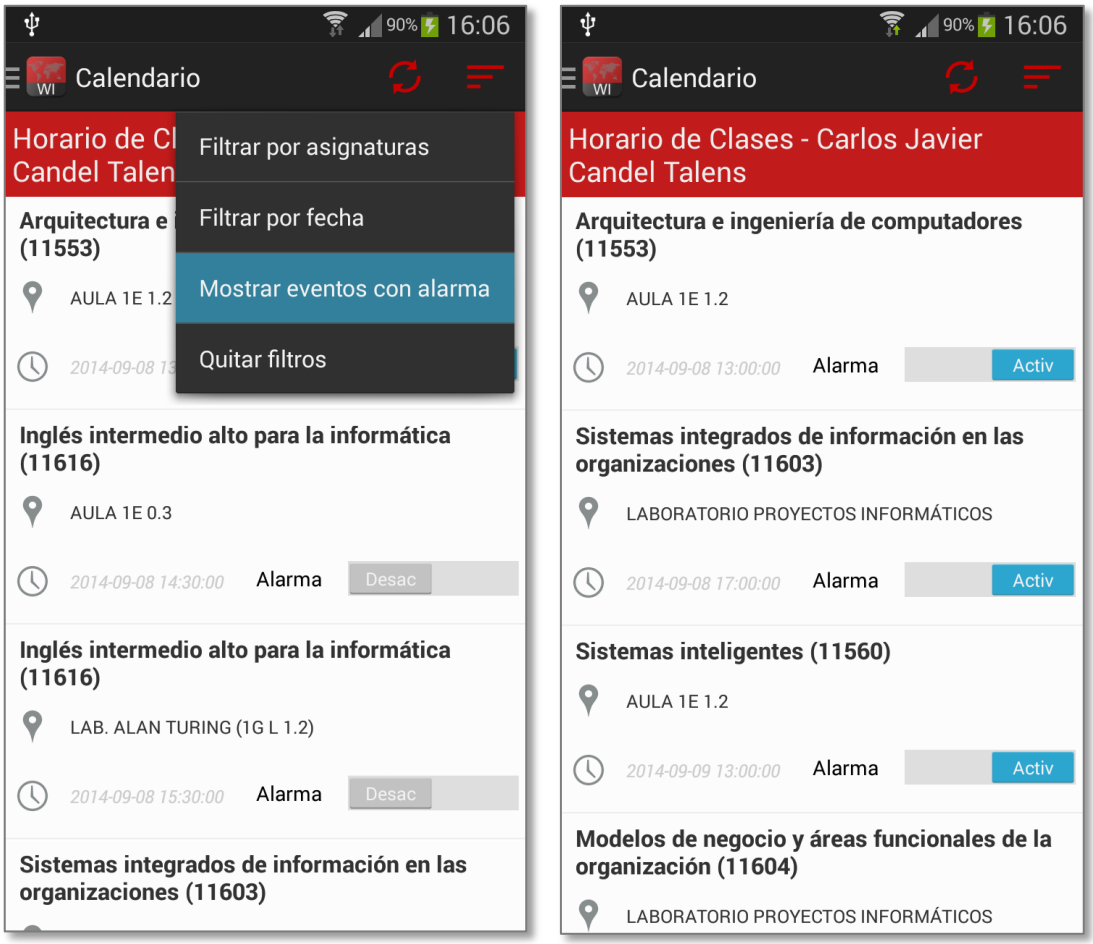

Figura 29 Filtro de eventos por Fecha

Como hemos comentado antes, si se quieren eliminar los filtros y volver a mostrar los eventos sin filtros solo habría que pulsar la opción 'Quitar filtros' en el panel desplegable (Figura 30)

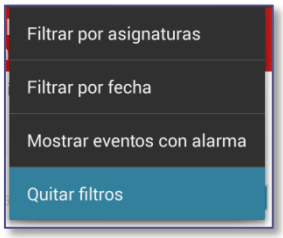

Figura 30 Quitar filtro

6.2.2 Aspectos técnicos

Al desarrollar el calendario, al no disponer de acceso interno a la UPV y su servicios, se ha tenido que buscar alternativas para obtener los eventos de calendario de los alumnos.

Debido a que no hay constancia de manera externa de si existen servidores con acceso a esos datos, la forma que se ha utilizado para obtener los calendarios es mediante HTTP.

Cuando un alumno accede a su intranet desde el navegador tiene acceso a su horario personal versión web y también tiene la posibilidad de obtener el horario personal en los formatos más utilizados por los calendarios nativos de los diferentes dispositivos móviles. De esta forma, los estudiantes pueden obtener los archivos .iCal de sus horarios.

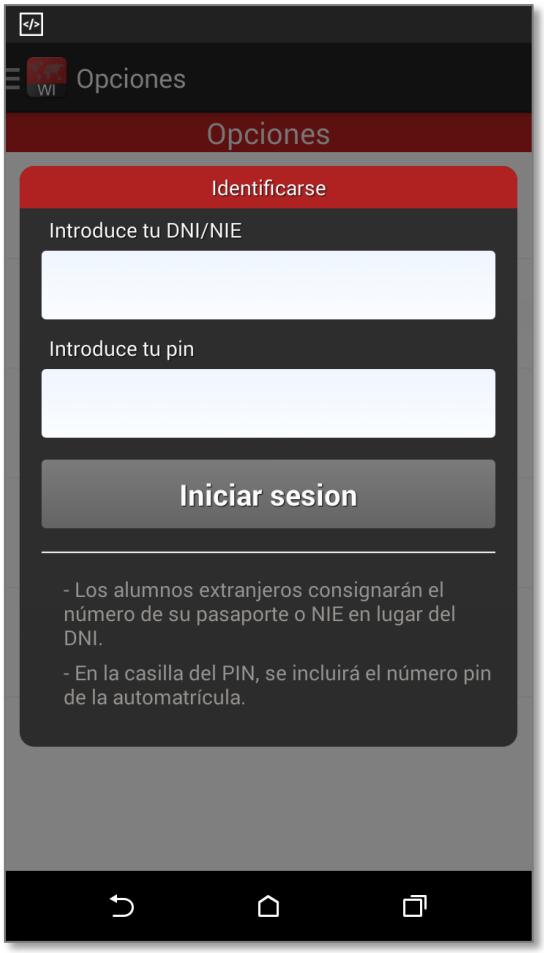

Figura 31 Pantalla de inicio de sesión

Para poder acceder a estos horarios primero es necesario iniciar sesión en la intranet de la UPV, que se realiza mediante observadores y peticiones HTTPS por lo tanto la conexión es segura y no hay ningún tipo de peligro de seguridad. Hay una jerarquía de hilos en la aplicación que realizan la tarea de conectar al usuario a la Intranet y traer los calendarios. La Figura 31 muestra el proceso de una forma simplificada.

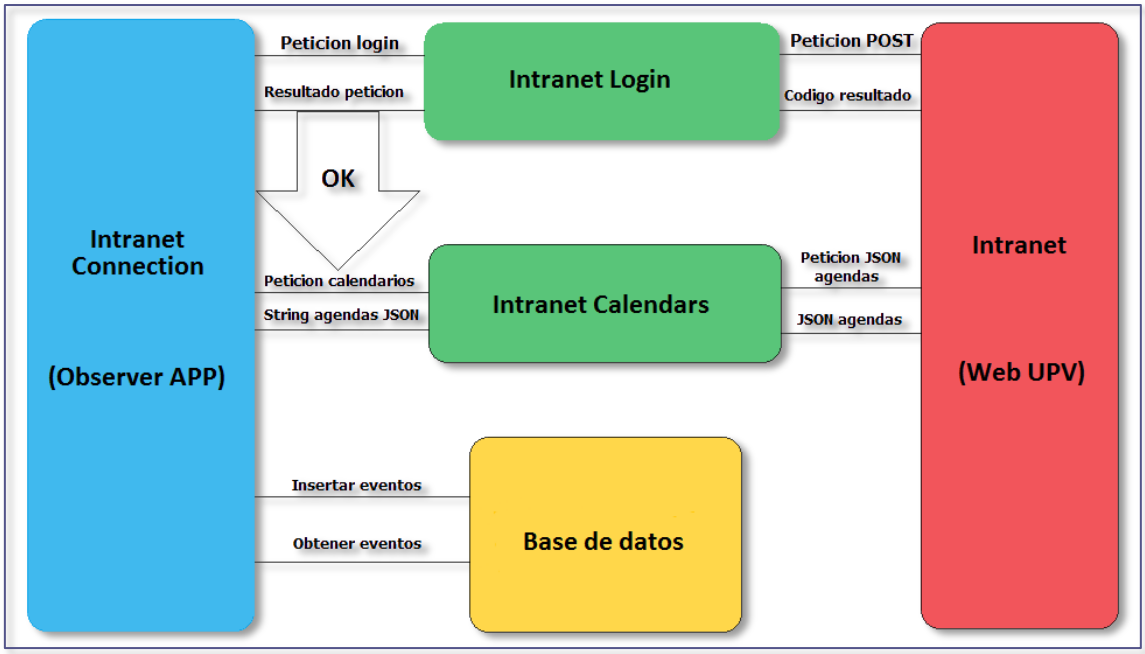

Figura 32 Proceso de inicio de sesión y obtención de calendarios

En la aplicación se emula una petición http GET para obtener el JSON de los calendarios.


```
"version" :  "1.0",
"username" :  "López Espejo, Cristian",
"agendas" : [
{
"uid" : "ICALUPVFEED29",
"nombre" : "Agenda UPV",
"grupo" : "publico",
"url" : "http://www.upv.es/ical/1DB54EC646B0BBD877456A8836A34C1FE8A2F498557FC3CBC7D07B76D1459EE0.ics"
}
,
{
"uid" : "ICALUPVFEED74",
"nombre" : "Horario de Clases - Cristian López Espejo",
"grupo" : "personal",
"url" : "http://www.upv.es/ical/BBD9F16FF3BA51AE2EA64949499C1E0FEAC6A372E1CDA9A450801A1677E9AE4096535A0FEE86A909.ics"
}
,
```


Figura 33 Array JSON con los calendarios del alumno

Una vez obtenido el array de horarios en formato JSON, accedemos al enlace del archivo ICS que nos facilita. ICS es un formato universal de calendario que podemos abrir con numerosos clientes de calendario.

Cuando obtenemos este archivo, el archivo es convertido a una variable String que podemos utilizar y maneja a nuestro antojo y así crear un objeto de la clase Calendar de la librería ical4j. Esta librería permite manejar archivos iCal en java.

```
CompatibilityHints.setHintEnabled(CompatibilityHints.KEY_RELAXED_PARSING, true);
StringReader sin = new StringReader(outPutParamsIntranetConnection.getIcsString());
CalendarBuilder builder = new CalendarBuilder();
Calendar calendar = builder.build(sin);
CalendarICAL calendarICAL = new CalendarICAL(calendar, this.calendario.getUid());
```

Figura 34 Creación del calendario iCal

Una vez obtenemos el objeto CalendarICAL podemos acceder a todos los eventos de este calendario, ya que este objeto se compone de un ID y un ArrayList de Eventos de calendario.

De esta forma, podemos rellenar el array de eventos del calendario (En este caso el calendario de horario de clases) y añadirlos al adaptador de la lista del fragment de calendario y poblar la lista.

```
eventos = obtenerEventosDB(db);
Collections.sort(eventos, new ComparadorEventos());
this.adapter = new ArrayAdapterCalendarDiaryItemList(this.getActivity(), eventos);
this.setListAdapter(adapter);
textViewName.setText(getHorario());
```

Figura 35 Rellenado de la lista

Al obtener los eventos los metemos en la base de datos interna de la aplicación y así poder acceder rápidamente cada vez que se abra el calendario.

Si se desea actualizar la lista de eventos, al pulsar el botón de actualizar se volvería a realizar todo este proceso.

Alarmas

Para las alarmas se ha creado un servicio de Android para manejarlas. En concreto se han utilizado dos clases: AlarmReceiver y AlarmIntentService.

Primero se llama a la clase AlarmReceiver al crear la alarma en el calendario moviendo de lado el interruptor del evento.

```
Intent intent = new Intent("alarmReceiver");
if (b) {
    actual.setAlertado(1);
    eventos.set(position, actual);
    mCheckedState[position] = true;
    setUniqueID(actual);
    int id = getUniqueID(actual);
    intent.putExtra("nombre", actual.getNombre());
    intent.putExtra("edificio", actual.getUbicacion());
    intent.putExtra("hora", actual.getFecha());
    intent.putExtra("id", id);
    PendingIntent pendingIntent = PendingIntent.getBroadcast(getActivity(), id, intent, 0);
    Date date = null;
    try {
        date = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.ENGLISH).parse(actual.getFecha());
    } catch (ParseException e) {
        e.printStackTrace();
    }
    long when = date.getTime();
    AlarmManager alarmManager = (AlarmManager) getActivity().getSystemService(Context.ALARM_SERVICE);
    alarmManager.set(AlarmManager.RTC, when, pendingIntent);
}
```

Figura 36 Creación de una alarma

Como podemos ver, se crea un intent con todos los datos y se crea otro PendingIntent que de este. Este intent es una referencia al primer intent creado, y que puede ser utilizado por otras aplicaciones tales como un servicio. Por último se crea un AlarmManager que permite ejecutar una aplicación en un cierto momento en el futuro.

La primera clase extiende BroadcastReceiver y su función es la de recibir los intent que llegan cuando se crea una notificación. Esta clase recibe el evento y se lo pasa a la clase servicio. Como podemos ver, el Intent contiene los datos del evento del que se programa la alarma. Estos datos también son pasados al servicio.

```
public class AlarmReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Intent service1 = new Intent(context, AlarmIntentService.class);

        Log.e("nombre", intent.getStringExtra("nombre"));
        service1.putExtra("nombre", intent.getStringExtra("nombre"));
        service1.putExtra("id", intent.getIntExtra("id", 0));
        service1.putExtra("edificio", intent.getStringExtra("edificio"));
        service1.putExtra("hora", intent.getStringExtra("hora"));
        context.startService(service1);
    }
}
```

Figura 37 Clase AlarmReceiver

Una vez creado el intent con todos sus datos, se pasa al servicio, que recibe los datos y los trata para crear la notificación que saldrá en el panel de notificaciones.

```
if (intent != null) {
    int id = intent.getIntExtra("id", 0);
    String nombre = intent.getStringExtra("nombre");
    String edificio = intent.getStringExtra("edificio");
    String hora = intent.getStringExtra("hora");

    intent1.putExtra("nombre", intent.getStringExtra("nombre"));
    intent1.putExtra("edificio", intent.getStringExtra("edificio"));
    intent1.putExtra("hora", intent.getStringExtra("hora"));

    PendingIntent pendingNotificationIntent = PendingIntent.
            getActivity(this.getApplicationContext(), id, intent1, PendingIntent.FLAG_UPDATE_CURRENT);
    Log.e("nombre", nombre);
    NotificationCompat.Builder mBuilder =
            new NotificationCompat.Builder(this)
                    .setSmallIcon(R.drawable.ic_launcher)
                    .setContentTitle(nombre)
                    .setAutoCancel(true)
                    .setSound(RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION))
                    .setSubText(edificio)
                    .setContentText("The class is about to start");


    mBuilder.setContentIntent(pendingNotificationIntent);
    mManager.notify(id, mBuilder.build());
```

Figura 38 Clase AlarmIntentService

Finalmente, cuando llegue el instante de tiempo que se ha establecido para el AlarmManager, se creara la notificación.

6.3 Información

El tercer ítem del panel de navegación es la información estática. Este apartado se basa en la Guía de Intercambio que proporciona la UPV.

La información que se muestra es la siguiente:

- **Información**: Se muestra información general sobre información de la UPV y de Valencia. Esta información básica contiene teléfonos de interés de Valencia y de la UPV, así como direcciones importantes y sus respectivos teléfonos.

- **Asignaturas**: Se muestra información sobre las asignaturas impartidas en cada Escuela e información relativas a estas. Esta información se ha extraído directamente de la Guía de Intercambio de la universidad.

- **Escuelas**: Se muestra información sobre las escuelas de la UPV e información importante como teléfonos, correos electrónicos o direcciones. Esta información se ha extraído directamente de la Guía de Intercambio de la universidad.

- **Transportes**: Se muestra información relativa los transportes de Valencia, donde también muestran teléfonos de interés y enlaces a los mapas y direcciones web de cada uno de los transportes.

6.3.1 Diseño de interfaces

Al pulsar sobre el elemento 'Info' del drawer se muestra una pantalla donde se da a elegir entre la diferente información disponible para consultar.

El menú está dividido en dos secciones: UPV y Valencia como podemos observar en la Figura 39.

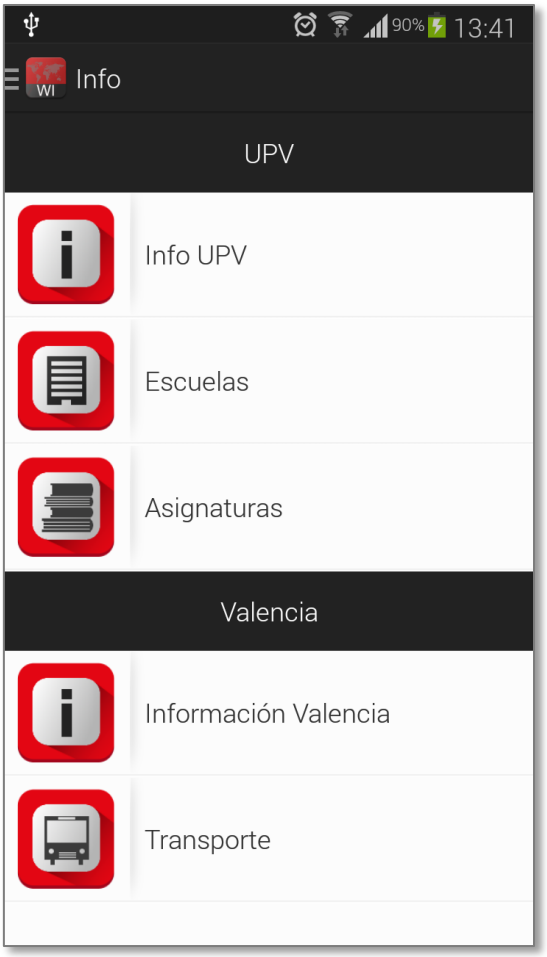

Figura 39 Pantalla de información

UPV

Dentro del apartado de la UPV podemos a acceder a la información sobre asignaturas y escuelas, además de la información de interés.

1. Información UPV

Al pulsar sobre 'Información UPV' se accede a una pantalla (Figura 40) sobre información de contacto de la UPV. Al pulsar cualquier elemento que este subrayado se abrirá la aplicación que se necesite para manejarlo.

Por ejemplo, si pulsamos sobre el teléfono se nos redirigirá a la aplicación 'Teléfono' del móvil Android para realizar la llamada. Si se pulsa sobre la web, se abrirá el navegador web y se mostrara la página. De misma manera si pulsamos sobre el email, se abrirá la aplicación de correo que se elija para proceder a enviar un correo a esa dirección.

Figura 40 Pantalla de información de la upv

2. Escuelas

En este apartado se muestran las escuelas que componen la Universitat Politècnica y su información más importante.

Esta pantalla se compone de una lista expandible (ExpandableListView) que muestra los nombres de las escuelas, y una vez expandida una escuela, se puede ver la información asociada a esta. Como se ha comentado en el apartado de la información, cada enlace a teléfonos, webs, emails, etc… abre su aplicación correspondiente en Android.

En esta información que se muestra podemos distinguir:

- URL de la página de la escuela
- URL de la docencia
- Correo electrónico de la escuela
- Fax
- Información del coordinador
- Información del técnico

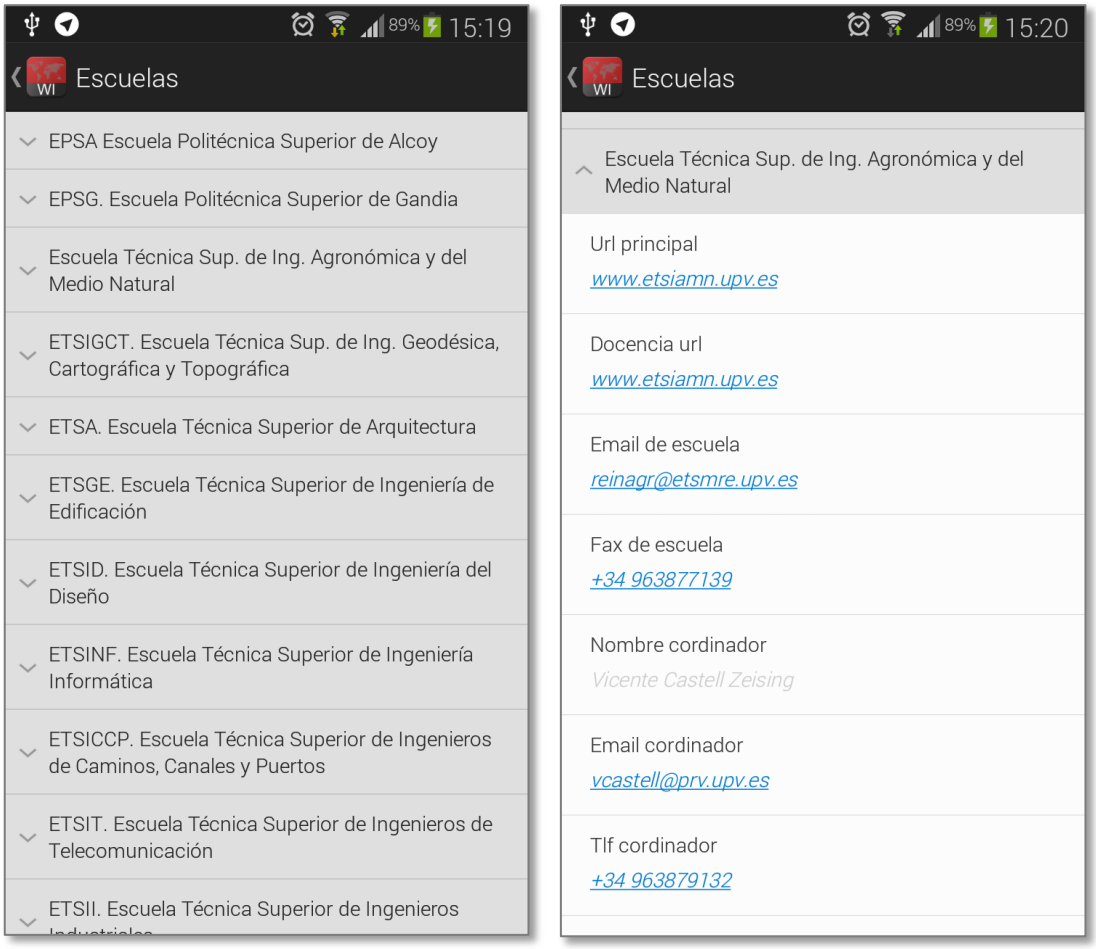

Figura 41 Escuelas (DER.) e informacion sobre las ESCUELAS (izq)

3. Asignaturas

En este apartado se pueden consultar las asignaturas que se imparten en las principales escuelas de la UPV.

Si bien es cierto que no están contempladas todas las escuelas y todas las asignaturas, se quiere introducir toda la información relativa a todas las escuelas en un futuro próximo debido a que las asignaturas de las escuelas del curso 2014/2015 no se han publicado a fecha de la redacción de esta memoria.

Esta vista también se compone de una lista expandible (MultiExpandableList) que muestra los nombres de las escuelas a consultar. Cuando una escuela se expande se muestran las asignaturas que se imparten en esta e información relativa a estas.

Si nos fijamos en las asignaturas podemos distinguir diversa información:

- Titulo
- Código de asignatura
- Semestre en el que se imparte
- Créditos
- Enlace a la web desde el icono situado a la parte superior derecha del elemento.

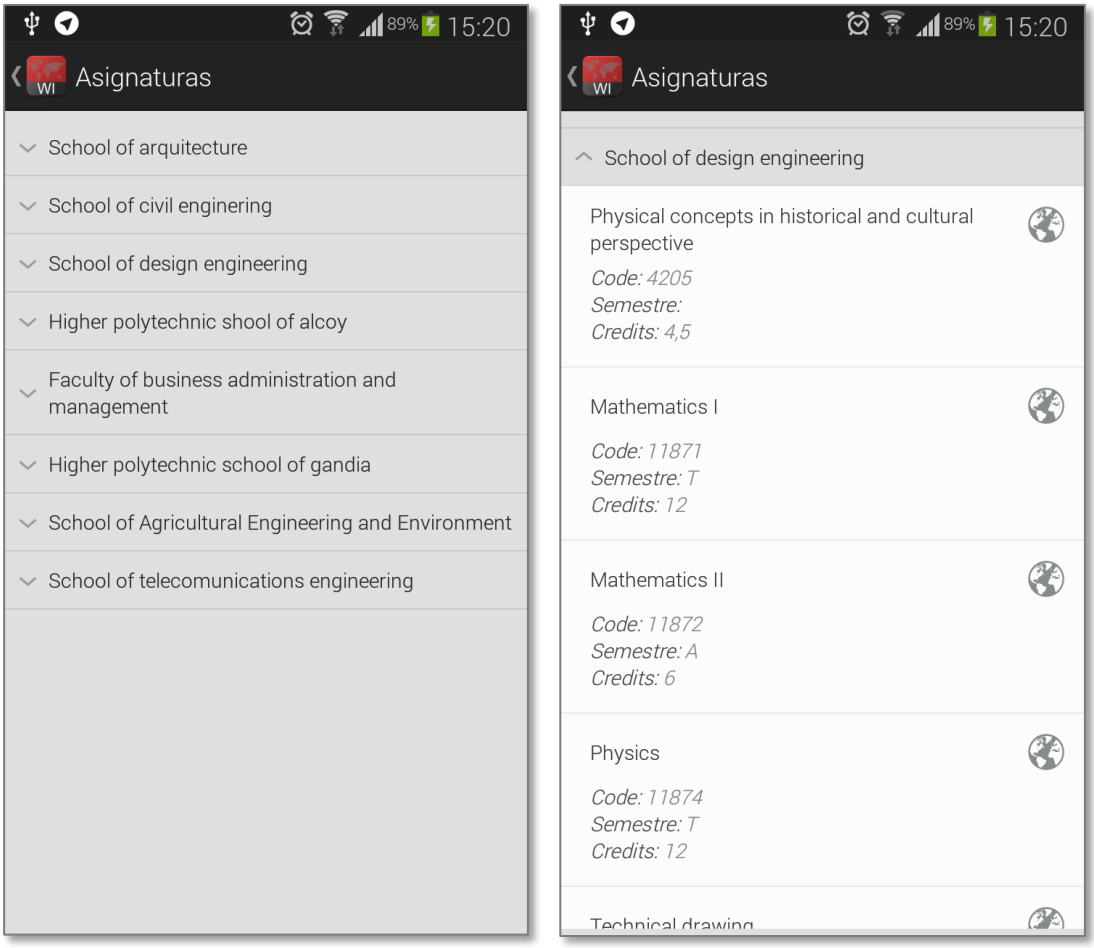

Figura 42 Asignaturas (DER.) e información sobre las asignaturas (izq)

Valencia

En esta parte del menú se muestra información necesaria sobre la ciudad de Valencia.

Si bien en este apartado se podría haber incluido la información relevante en cuanto a fiestas de Valencia, embajadas, lugares de encuentro, etc… se ha creído necesario únicamente estas dos opciones debido a que esta aplicación no está orientada a ser una guía turística si no una herramienta de localización e integración para el alumno de intercambio que acaba de llegar a la UPV.

El menú da dos opciones:

- Información Valencia
- Transporte

1. Información Valencia

Esta pantalla es muy parecida a la información de la UPV. Muestra la información más importante sobre el ayuntamiento, como es:

- Información sobre el ayuntamiento de Valencia
- Información sobre la Policía Local de Valencia
- Información sobre los Bomberos de Valencia

La Figura 43 muestra como se muestra la información en esta actividad.

Figura 43 Información sobre valencia

2. Transportes

Esta pantalla es la más elaborada de la funcionalidad 'información' ya que los estudiantes al venir de fuera requieren más información sobre cómo moverse por Valencia o simplemente como llegar a la UPV de la forma más sencilla y cómoda.

Los transportes que se han añadido a la información de la aplicación son los siguientes:

- Metro
- Autobús
- Avión
- Tren
- Taxi

Las pantallas de cada transporte tienen un diseño parecido. Cada pantalla de transporte es un fragment que se implementa en un ViewPager. Un ViewPager permite cargar fragments de forma que se puedan ir cambiado simplemente deslizando el dedo por la pantalla, aspecto que aumenta la facilidad ya que requiere muchas menos pulsaciones que una interfaz con botones.

El diseño de los fragment de transporte como podéis ver en la Figura 44 se compone de un texto con una descripción sobre el transporte, acompañado de unos botones que permiten al usuario ver un mapa del transporte, llamar a la empresa del transporte o visitar su web.

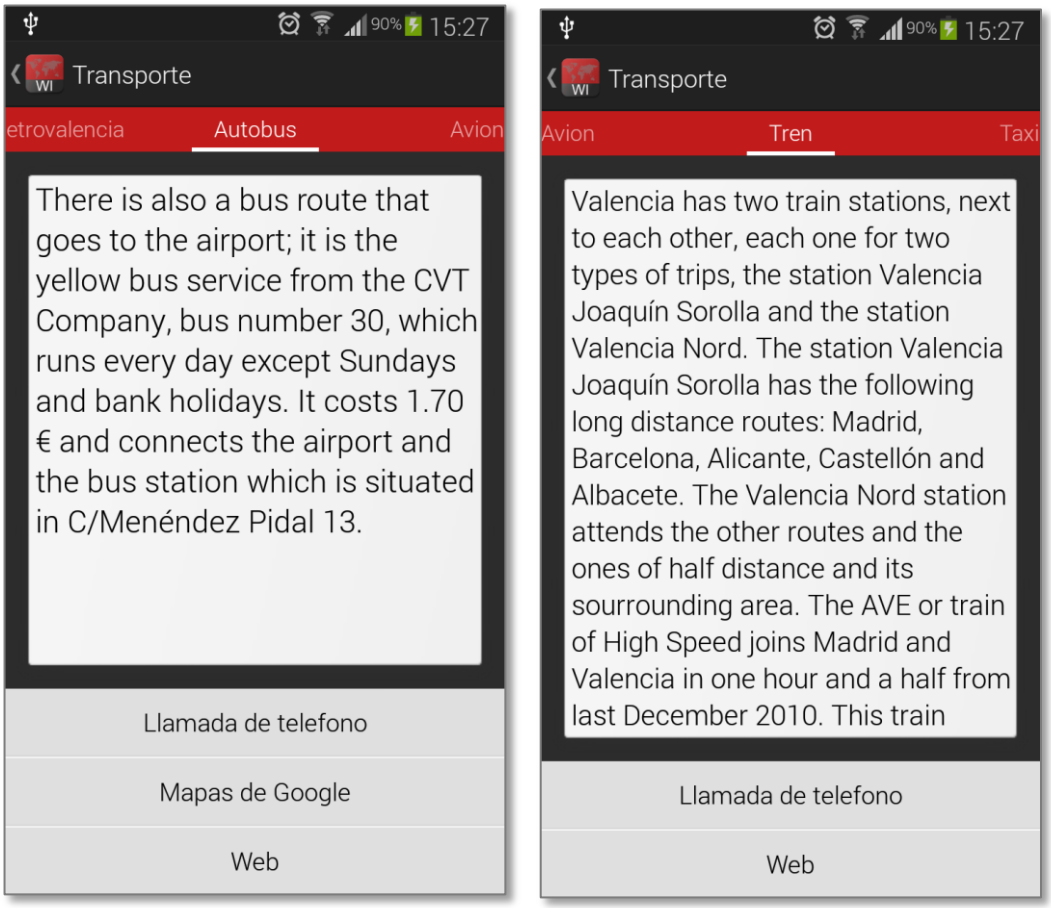

Figura 44 Fragment avión (DER.) y fragment tren (IZQ)

6.3.2 Aspectos técnicos.

Diseño de interfaces.

En esta funcionalidad de la aplicación se ha intentado diseñar una interfaz sencilla de utilizar y de fácil acceso. El menú es una interfaz tipo DashBoard dividido de una forma clara entre los dos tipos de información que puedes consultar.

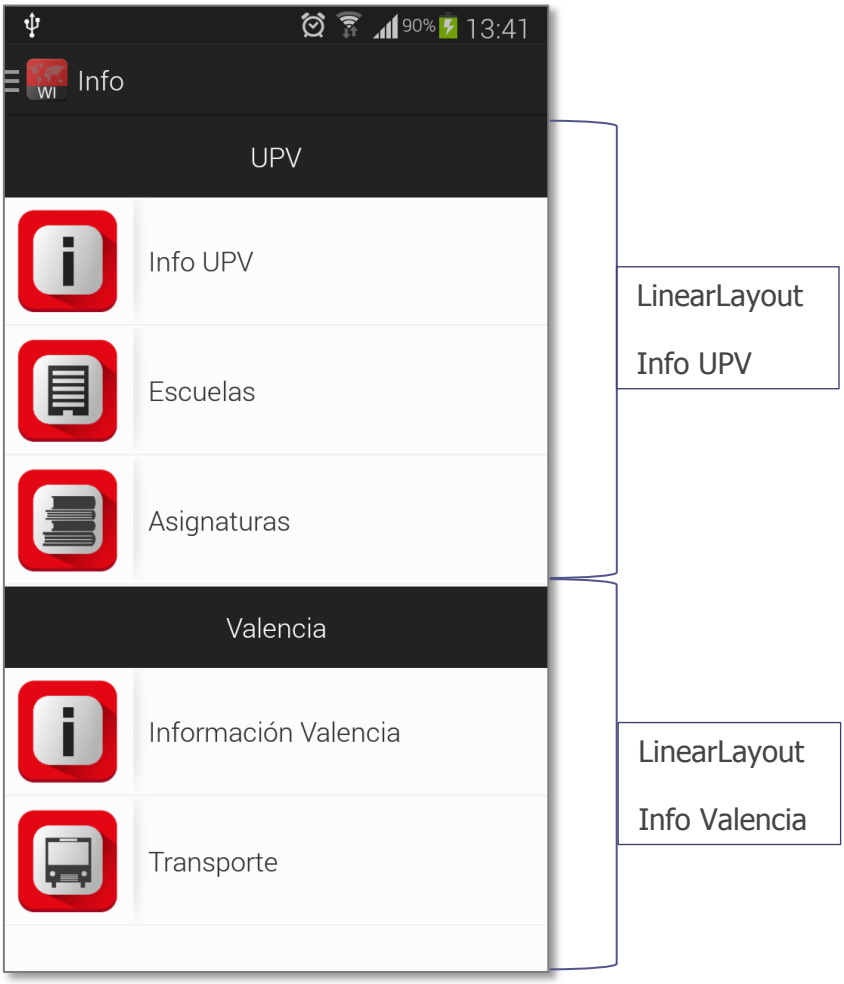

Figura 45 División de interfaz del menú de información

Las interfaces de las pantallas de información, tanto la de la UPV como la de Valencia, disponen del mismo Layout pero con colores diferenciados e imagen diferente para saber en todo momento en que información estás.

Las pantallas de Escuelas y Asignaturas también son parecidas en cuanto a que contienen ambas una lista expandible. Si bien es cierto que no tienen el mismo adaptador de lista, se parecen mucho en la forma en mostrar la información.

Una lista expandible en Android se compone de una vista del elemento padre (Header) y elementos hijo (Child). Los elementos padre son los elementos que se muestran en la lista como si de una lista normal se tratase mientras que los elementos hijo son aquellos que se muestran cuando se despliega el elemento padre. Para mostrar cómo funciona este tipo de lista, en la Figura 46 se muestra como se divide en las vista de escuelas.

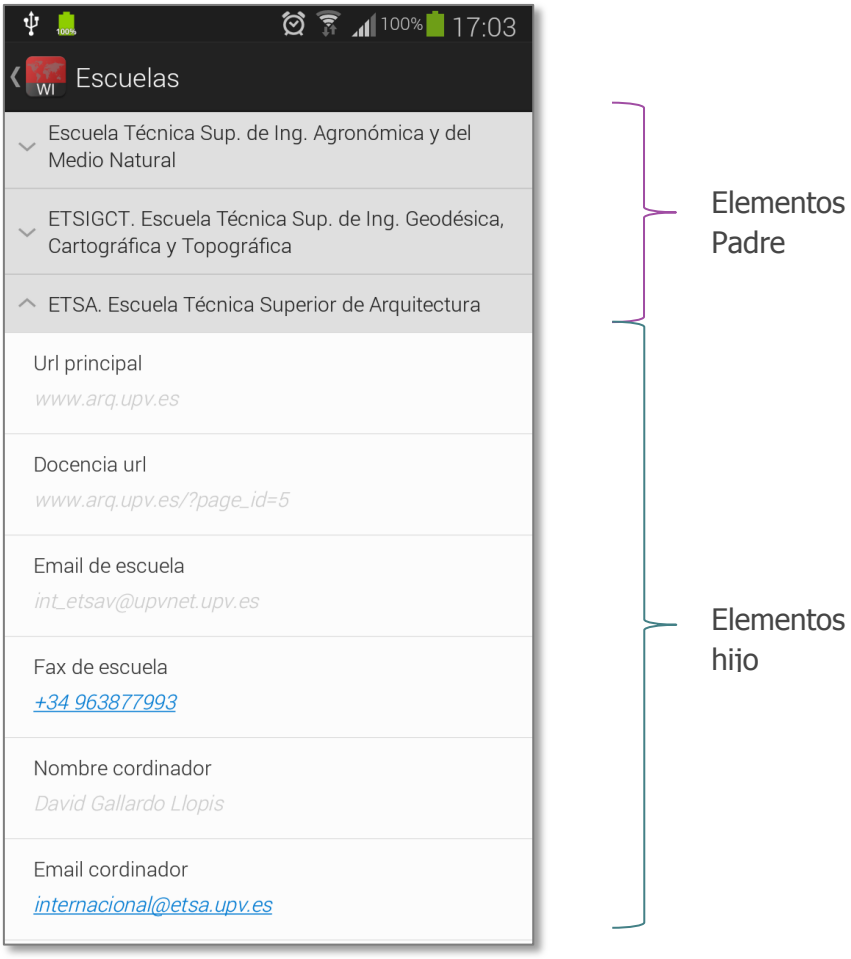

Figura 46 Distribución de una lista desplegable

Los hijos son elementos los elementos que se han personalizado para la aplicación. Se ha creado la vista del elemento hijo con un texto donde se escribe el título y otro texto debajo de este que almacena el valor.

Para que los elementos puedan ser pulsados y abrir el teléfono, cliente de correo, etc… se mira si el valor que se va a almacenar en el texto (TextView) es un valor telefónico, email o web y se asigna la acción al texto con el método 'setAutoLinkMask' que asigna un link a un TextView para que cuando se pulse se produzca una acción.

```
switch (childPosition) {
    default:
        break;
    case 0:
        tv_descripcion.setAutoLinkMask(Linkify.WEB_URLS);
        tv_descripcion.setMovementMethod(new ScrollingMovementMethod());
        break;
    case 1:
        tv_descripcion.setAutoLinkMask(Linkify.WEB_URLS);
        tv_descripcion.setMovementMethod(new ScrollingMovementMethod());
        break;
    case 2:
        tv_descripcion.setAutoLinkMask(Linkify.EMAIL_ADDRESSES);
        tv_descripcion.setMovementMethod(new ScrollingMovementMethod());
        break;
    case 3:
        tv_descripcion.setAutoLinkMask(Linkify.PHONE_NUMBERS);
        tv_descripcion.setMovementMethod(new ScrollingMovementMethod());
        break;
    case 5:
        tv_descripcion.setAutoLinkMask(Linkify.EMAIL_ADDRESSES);
        tv_descripcion.setMovementMethod(new ScrollingMovementMethod());
        break;
    case 6:
        tv_descripcion.setAutoLinkMask(Linkify.PHONE_NUMBERS);
        tv_descripcion.setMovementMethod(new ScrollingMovementMethod());
        break;
```

Figura 47 Auto link de la lista desplegable

La interfaz de transportes es un caso especial ya que usa un ViewPager para intercambiar los fragments de cada transporte.

Un ViewPager permite al usuario de la aplicación pasar entre pantallas con un simple movimiento de dedo, lo que en ingles se denomina *swipe* (Deslizar en español). Esto le da un toque de sencillez a la aplicación, además de darle un toque moderno y actual.

Un ViewPager dispone de diferentes páginas en las que podemos cargar nuestras vistas, y en las que en este caso nosotros cargamos los fragments de cada uno de los transportes.

La vista que utilizamos para el pager se compone de una barra superior (PagerTabStrip) que te indica que fragment estas utilizando mostrando el título del este.

Figura 48 Distribución ViewPager

Cuando se desliza el dedo hacia la derecha, el fragment de la izquierda pasa a ser el fragment visible, y a la misma vez, el pagertabstrip cambia de posición al título de la izquierda.

Para que los fragment no tengan que cargarse cada vez que deslizas, se crea un adaptador para el ViewPager que permite añadir fragments. Al añadir un fragment se le otorga una posición para seguir la jerarquía de páginas.

```
ViewPager pager = null;
PagerTabStrip tabStrip;
FragmentPagerAdapter_Info pagerAdapter;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.fragment_info_transportes);

    this.pager = (ViewPager) findViewById(R.id.pager);
    this.tabStrip = (PagerTabStrip) findViewById(R.id.pager_tab_strip);
    pager.setPageTransformer(false, (page, position) -> {
            final float normalizedposition = Math.abs(Math.abs(position) - 1);
            page.setAlpha(normalizedposition);
    });
    pagerAdapter = new FragmentPagerAdapter_Info(getSupportFragmentManager(), this);
    pagerAdapter.addFragment(new Fragment_Metro());
    pagerAdapter.addFragment(new Fragment_EMT());
    pagerAdapter.addFragment(new Fragment_Avion());
    pagerAdapter.addFragment(new Fragment_Tren());
    pagerAdapter.addFragment(new Fragment_Taxi());
    pager.setAdapter(pagerAdapter);
}
```

Figura 49 Método de creación del ViewPager

Finalmente, cuando ya se han añadido todos los fragments al adaptador, se le asigna el adaptador al pager para hacer efectivos los cambios.

Obtención de la información

En esta funcionalidad se hace uso de información estática tales como las asignaturas, escuelas, transportes…

Para almacenar toda esta información extraída de forma manual de la Guía de Intercambio de la UPV se ha utilizado archivos XML. Esta información se guarda en la aplicación al instalarse y se accede desde las actividades mediante unos parsers XML creados a medida para estos documentos.

Se han creado varios archivos XML como se muestra en la Figura 50 para almacenar toda la información relativa a esta funcionalidad.

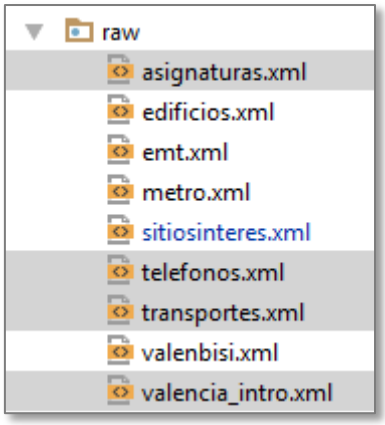

Figura 50 Archivos XML de información (seleccionados)

El parser creado permite extraer la información de acuerdo al tipo de datos que se quiere extraer. Por ejemplo, si deseamos extraer la información del XML de transportes, deberíamos ejecutar este comando:

```
InputStream fichero = getResources().openRawResource(R.raw.transportes);
Parser_XML parseador = new Parser_XML("transportes");
List<Transporte> listaTransportes = parseador.parseando(fichero);
```

Figura 51 Llamada al parseador de transportes

Al recibir el fichero que se quiere parsear, el parseador ejecuta el siguiente código:

```
else if (tipo_xml.equals("transportes")) {

/*Vamos a devolver un ArrayList de objetos transporte*/
    resultado = new ArrayList<Transporte>();

    while (tipoevento != XmlPullParser.END_DOCUMENT) {

        if ((tipoevento == XmlPullParser.START_TAG) && !(parseado.getName().equals("transportes"))) {

            Transporte nuevo = new Transporte();
            nuevo.setNombre(parseado.getName());
            nuevo.setUrl(parseado.getAttributeValue(null, "url"));
            nuevo.setDescripcion(parseado.getAttributeValue(null, "descripcion"));
            nuevo.setTelefono(parseado.getAttributeValue(null, "telefono"));
            resultado.add(nuevo);
            parseado.next();
            tipoevento = parseado.getEventType();
        } else {
            parseado.next();
            tipoevento = parseado.getEventType();
        }


    }

}
```

Figura 52 Parseador de transportes

Y se recibe una lista de transportes con todos los transportes. Cabe decir que este código no se ejecuta la abrir ninguna actividad del apartado 'Información' si no que se realiza en la pantalla de carga, donde una vez obtenidas las listas de transportes se almacena en la base de datos.

En las actividades de los transportes, cada vez que se llama a un transporte en concreto se realiza una consulta a la base de datos y se obtienen la lista del transporte que se quiere obtener.

Con esta lista de transportes que se obtiene de la base de datos, se obtienen los datos y se muestran en la pantalla.

6.4 Localización

Al plantear los objetivos de esta aplicación, un requerimiento esencial era la posibilidad de que el usuario se ubicase dentro de la UPV con facilidad. También era necesario que el estudiante de intercambio supiese, apoyándose en la aplicación, moverse por Valencia cómodamente, mediante transporte público o a pie.

Otras aplicaciones disponibles permiten al usuario obtener mapas de Valencia, de la UPV e incluso del transporte público. Al ser una aplicación destinada a ofrecer toda esta información, se pretendía que la información se obtuviese de una forma que la distinga de otras alternativas y en especial de la aplicación oficial de la UPV.

Para este fin se han proporcionado varias funcionalidades a la aplicación que permitirán que el estudiante se sienta cómodo en las inmediaciones de la Universitat Politècnica y en Valencia.

6.4.1 Diseño de interfaces

El menú de este apartado se compone de dos partes diferenciadas:

- Mapas estáticos:
    - Mapa de edificios de la UPV
    - Mapa Valenbisi
    - Mapa de paradas de Metro
    - Mapa de sitios de interés
    - Mapa de paradas de EMT
- Realidad aumentada.

Como se puede observar en la Figura 53, el menú tiene dos vistas que se diferencian. Por una parte tenemos la parte superior que se corresponde a los mapas y otra parte inferior donde se puede utilizar la realidad aumentada.

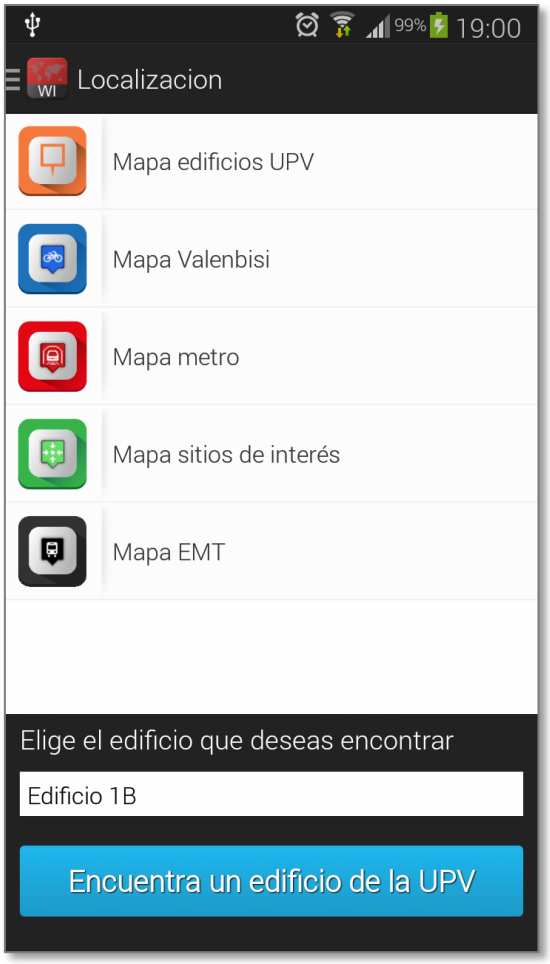

Figura 53 Menú Localización

Mapas estáticos

Una funcionalidad esencial para ayudar a la localización del usuario en Valencia y en la UPV es la disposición de mapas que le ayuden a ubicarse. En esta aplicación se ha hecho uso de los mapas de Google para crear mapas estáticos que no requieren conexión a internet, permitiendo así a los usuarios extranjeros que no disponen de conexión de datos en sus teléfonos la ocasión de posicionarse en la ciudad.

- **UPV**

En primer lugar tenemos el mapa de la UPV. Este mapa se basa en el mapa interactivo que proporciona la UPV que contiene edificios e información relacionada a estos.

En la Figura 54 tenemos reflejada la interfaz del mapa. Como se puede observar el mapa ocupa toda la pantalla disponible para poder ver de forma clara y cómoda toda la información.

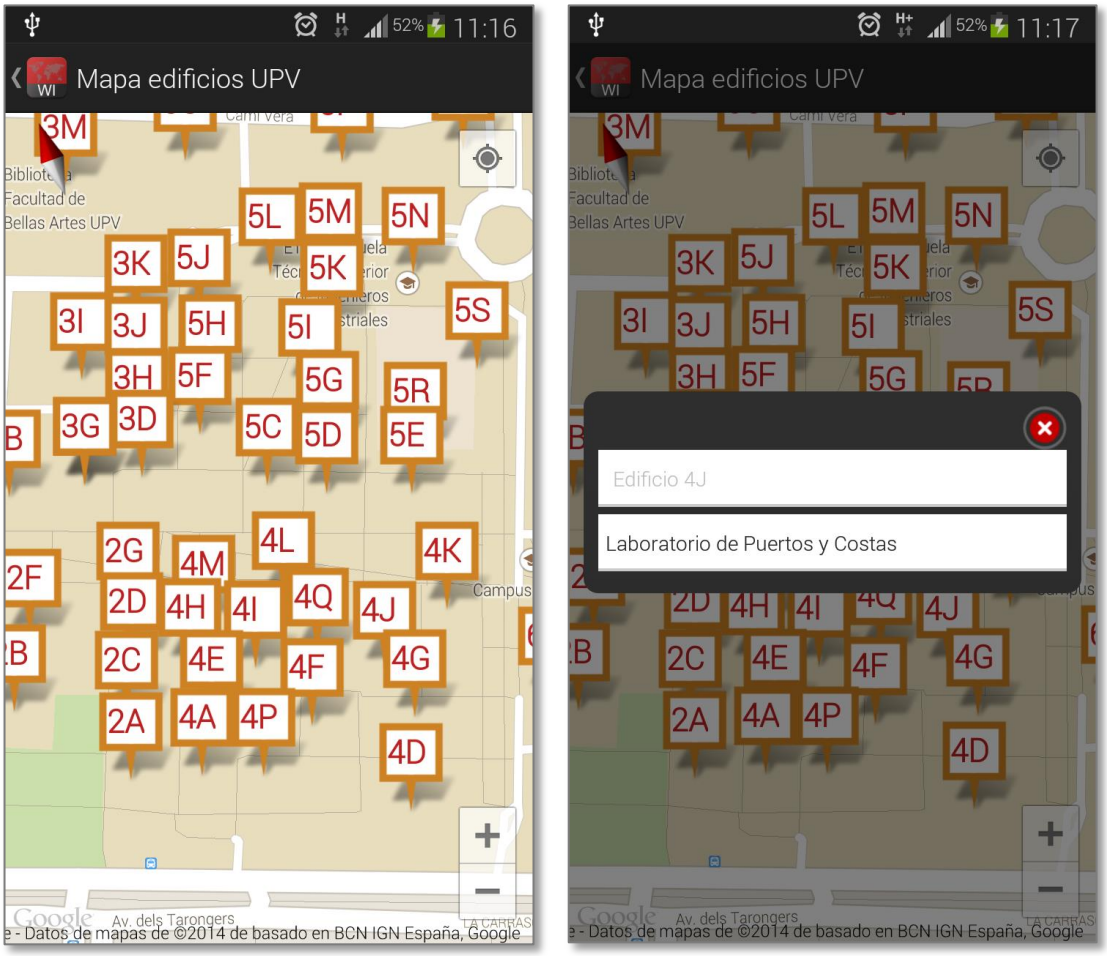

Figura 54 Mapa de la UPV (IZQ) e información de los edificios (DER)

Cabe destacar que la pantalla de mapas se puede ver tanto con la pantalla vertical como horizontal.

Si nos fijamos cada edificio tiene su propio marcador con su número de edificio para que se pueda acceder directamente al edificio que se quiere encontrar y no tener que buscar por los diferentes marcadores.

Cuando se pulsa uno de estos marcadores, una pantalla de información en forma de dialogo aparece y se muestra el nombre del edificio y toda la información que existe dentro de ese edificio.

Si pulsamos sobre un edificio donde existe un punto POI (Point of interest) que nos permita tener llegar hacia el mediante realidad aumentada, un icono al lado del nombre del edificio aparecerá y podremos guiarnos mediante la realidad aumentada (Figura 55).

Figura 55 Icono de realidad aumentada

- **Valenbisi**

El segundo mapa de la lista es el de Valenbisi. Este mapa se ha considerado necesario totalmente por el uso de la bicis que se da dentro de la universidad y fuera de ella. Una gran cantidad de estudiantes de la UPV utiliza este transporte público para moverse con facilidad entre la universidad y sus domicilios.

En este mapa se muestran todos los puntos Valenbisi de Valencia posicionados también mediante marcadores en el mapa de Google como se puede observar en la Figura 56.

En esta vista también se ha implementado una vista secundaria que se muestra al pulsar sobre un marcador en la cual se muestra el nombre de la estación de bicis y también información sobre cuantas bicis en total tiene y cuantas bicis de estas están disponibles en la estación. Esta información es obtenida en tiempo real, por lo que si se desea tener esta funcionalidad se debe tener internet en el dispositivo móvil. De

otra forma este contenido aparecerá en blanco, o si se ha consultado antes, los últimos datos obtenidos.

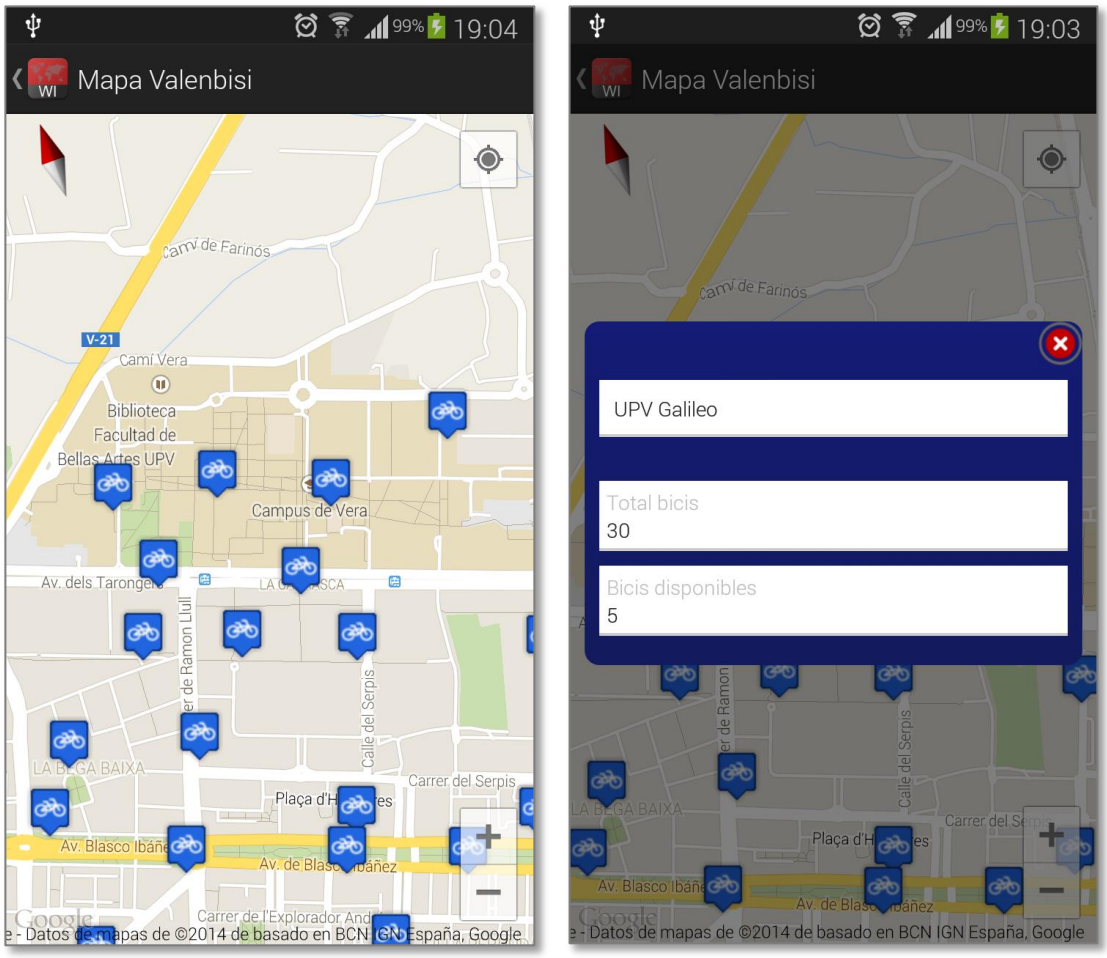

Figura 56 Mapa Valenbisi (IZQ) e información sobre las estaciones (DER)

Hay que destacar que, basados en la implementación de la aplicación, cualquier estación añadida o cualquier cambio de bicis totales en cualquier estación no será contemplada en la aplicación pero es una funcionalidad fácil de implementar y que se espera implementar en el futuro. Si no se ha mirado por implementar esta funcionalidad ha sido porque el objetivo de esta aplicación es el de presentar información de forma estática y que no precise de conexión a internet para funcionar.

- **Metro y tranvía**

El tercer elemento del menú de mapas es el mapa de metro. Este es también un mapa esencial para los estudiantes que vienen de otras partes del país o del mundo debido a que el metro facilita moverse por Valencia. También es necesario si vienes por el centro de Valencia y tienes que usar el transporte público para ir a la universidad. Por otra parte se muestran las estaciones de tranvía ya que son cruciales para llegar a la UPV.

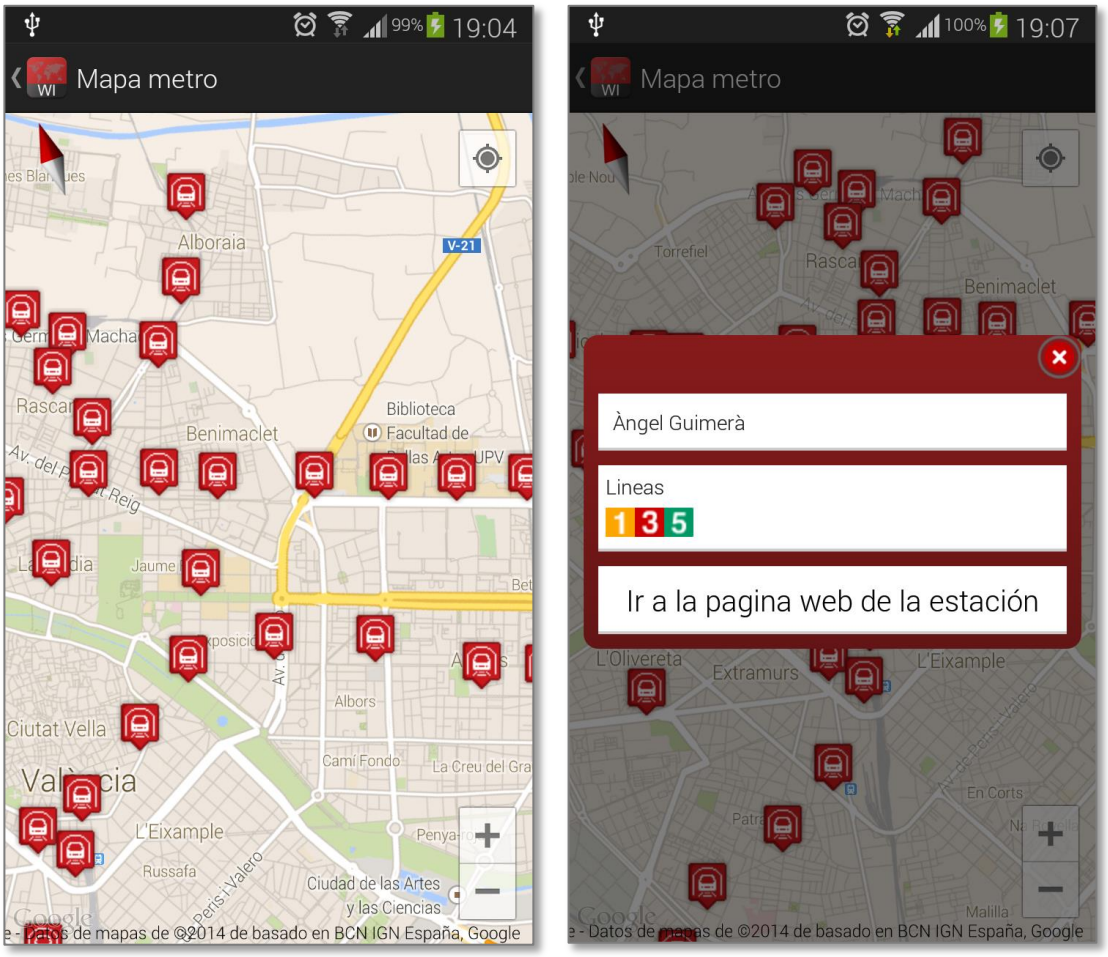

Figura 57 Mapa de metro (izq) e información sobre las estaciones (der)

Como se puede observar en la Figura 57, se muestran todas las paradas de metro y tranvía de Valencia. Al pulsar sobre una de estas se muestra una pantalla en forma de dialogo. Esta vista dispone de la siguiente información:

- Nombre de la estación
- Líneas que pasan por esa estación
- Enlace a la web de la estación para ver horarios

- **Sitios de interés**

El cuarto mapa disponible es el de sitios de interés. Este mapa es el que engloba todo tipo de sitios de interés a lo largo de Valencia.

Al pulsar sobre un marcador, muestra el nombre del sitio de interés que se ha pulsado.

Los sitios de interés introducidos corresponden a las zonas más transitadas de Valencia, pero se desean introducir más elementos de interés en versiones posteriores de la aplicación.

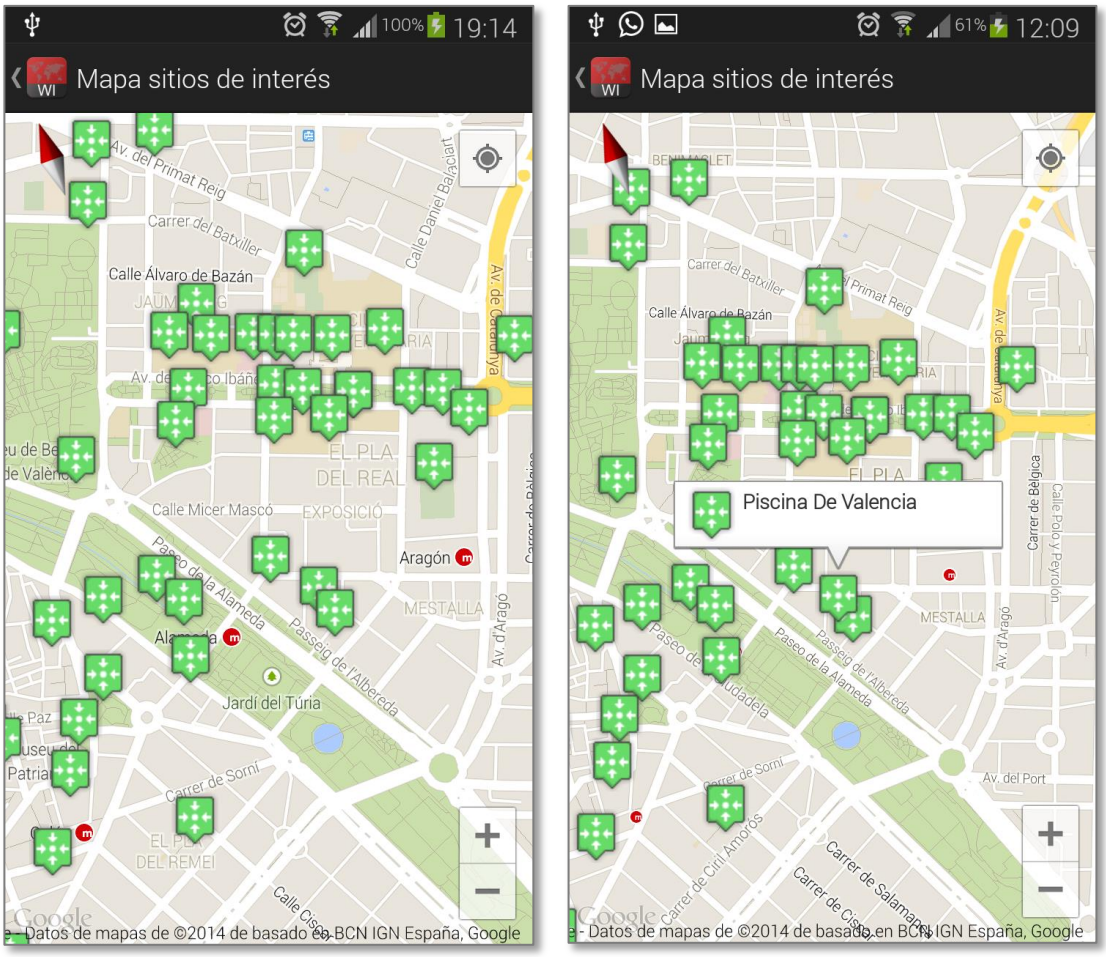

Figura 58 Mapa de sitios de interés (izq) e información sobre ellos (der)

- **EMT**

El último mapa disponible del apartado 'Localización' es el mapa de buses EMT. Este mapa presenta las estaciones de buses de Valencia más cercanas a las bocas de metro.

Aunque no contempla todas las estaciones EMT de Valencia, proporciona suficiente información para llegar a las zonas más importantes de la ciudad.

Al igual que en el mapa de sitios de interés, al pulsar sobre un marcador de estación se mostrara una vista que mostrara el nombre de la estación, que normalmente es la calle en la que se encuentra.

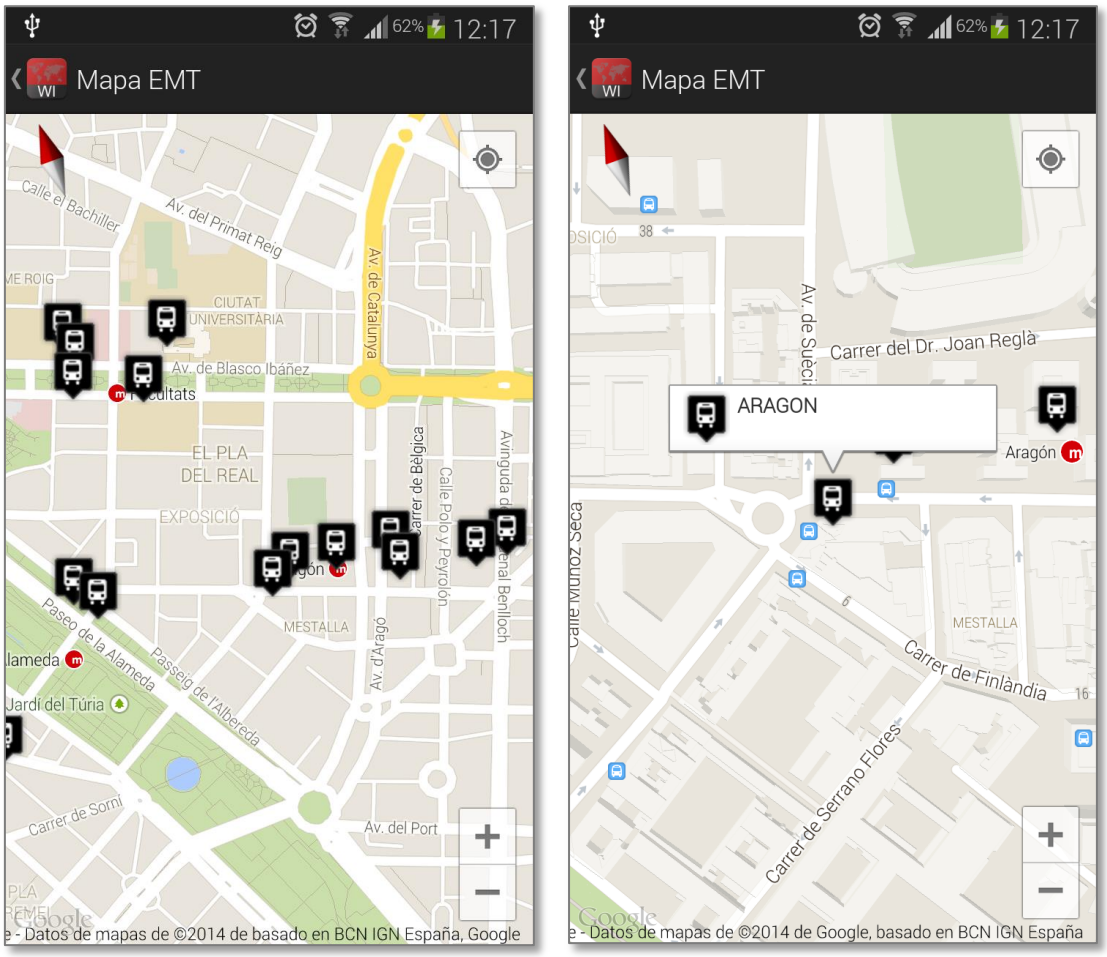

Figura 59 Mapa EMT (IZQ) e información sobre las estaciones (DER)

Como se puede observar en la Figura 59, las estaciones añadidas son las cercanas a sitios importantes y bocas de metro, ya que, como también se puede ver en la misma figura, si acercamos la vista en el mapa de Google se puede tener acceso a las estaciones de bus sin necesidad de marcadores. Por este mismo motivo no se ha visto necesario añadir todas las estaciones EMT de Valencia.

Realidad aumentada

La aplicación dispone de una funcionalidad que se ha añadido para apoyar a los mapas estáticos y que da un valor añadido muy importante. Esta funcionalidad es la realidad aumentada, cada vez más presente en dispositivos móviles.

Esta funcionalidad se basa en la premisa de que el estudiante que hace uso de ella se encuentra en la UPV y que desea buscar un edificio de la lista de edificios disponibles.

Figura 60 Edificios disponibles en Realidad aumentada

Cuando se escoge uno de los edificios disponibles de la lista y se pulsa el botón de 'Encontrar un edificio de la UPV' (Figura 60), se abre una nueva actividad que muestra una pantalla donde se puede encontrar el edificio que se ha seleccionado. Para esta actividad se ha utilizado una librería externa llamada Wikitude, que proporciona un SDK para Android que permite utilizar la realidad aumentada a gusto del desarrollador.

Para mostrarlo gráficamente se muestra en la Figura 61.

Si se observa con atención, se puede observar que la pantalla contiene 2 elementos:

- Radar: Se dispone de un elemento para posicionarte en el mapa y ver a que distancia y dirección está el edificio.
- Etiqueta del edificio: Se muestra el número del edificio y la escuela que imparte en ese mismo edificio.

Dependiendo a donde orientes el dispositivo, el edificio se mostrará en pantalla o no.

Figura 61 Actividad de realidad aumentada

6.4.2 Aspectos técnicos

Diseño de interfaces

Menú

El menú es un Dashboard simple formado por dos layouts principales: El layout correspondiente a la parte de la pantalla donde se posicionan los botones de los mapas y el layout correspondiente a la realidad aumentada.

Los iconos que se utilizan en este menú son personalizados. Se han creado utilizando el estilo de los marcadores que se usan en los mapas.

En el layout de realidad aumentada se ha utilizado un Spinner para mostrar los edificios. Este Spinner se rellena mediante un array de String que contiene los nombres de los edificios.

Figura 62 Layout del menú de localización

Mapas

Los mapas implementados en esta aplicación comparten la misma interfaz pero distinto contenido.

Las vistas se componen de un layout que engloba un fragment llamado 'com.google.android.gms.maps.MapFragment' (Figura 62). Este fragment se encarga de mostrar el mapa de Google sin ningún tipo de información, únicamente el mapa.

Figura 63 Interfaz de mapa

Como hemos podido observar en la Figura 54 el mapa contiene marcadores que se han añadido para facilitar la localización de los elementos.

A continuación se va a detallar el proceso para crear los marcadores y ubicarlos en el mapa de los diferentes mapas.

El proceso se va a realizar con el Mapa de la UPV como ejemplo, pero este proceso es el seguido en todos los mapas que contiene la aplicación.

Para mostrar los mapas se tiene cargar el mapa y posicionar los marcadores:

```
UpvMarkersHashMap = new HashMap<Marker, MarcadorEdificio>();
ArrayList<MarcadorEdificio> upvMarkersArray = obtenerMarcadoresEdificios(db);
setUpMap();
plotMarkers(upvMarkersArray);
```

Figura 64 Inicializacion de la actividad del mapa

Como podemos ver, primero se crea el HashMap donde se van a almacenar los edificios obtenidos de la base de datos. Después se creara el array de marcadores que contendrá todos los objetos de tipo MarcadorEdificio.

Después de cargar la información, se pasara a preparar el mapa:

```
private void setUpMap() {
    // Do a null check to confirm that we have not already instantiated the map.
    if (mMap == null) {
        // Try to obtain the map from the SupportMapFragment.
        mMap = ((MapFragment) getFragmentManager().findFragmentById(R.id.map_view)).getMap();
        LatLng posicionUPV = new LatLng(39.4810811, -0.3421079);
        CameraUpdate cameraUpdate = CameraUpdateFactory.newCameraPosition(new CameraPosition(posicionUPV, 15, 0, 20));
        mMap.animateCamera(cameraUpdate);

        mMap.setMyLocationEnabled(true);
        // Check if we were successful in obtaining the map.

        if (mMap != null) {
            mMap.setOnMarkerClickListener((marker) -> {
                    final String edificio = UpvMarkersHashMap.get(marker).getNumero();
                    final String[] info = UpvMarkersHashMap.get(marker).getInformacion().split(";");
                    Intent intent = new Intent(getApplicationContext(), Activity_Info_UPV.class);
                    intent.putExtra("edificio", edificio);
                    intent.putExtra("info", info);
                    startActivity(intent);
                    return true;
                });
        } else
            Toast.makeText(getApplicationContext(), "Unable to create Maps", Toast.LENGTH_SHORT).show();
    }
}
```

Figura 65 Método que configura el mapa

Una vez preparado el mapa, se preparan los marcadores:

```
private void plotMarkers(ArrayList<MarcadorEdificio> markers) {
    if (markers.size() > 0) {
        for (MarcadorEdificio myMarker : markers) {
            View view = ((LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE))
                    .inflate(R.layout.custom_marker_edificios, null);
            TextView numTxt = (TextView) view.findViewById(R.id.num_txt);
            numTxt.setText(myMarker.getNumero());
            LatLng pos = new LatLng(Double.parseDouble(myMarker.getLongitud()),
                    Double.parseDouble(myMarker.getLatitud()));
            Marker marker = mMap.addMarker(new MarkerOptions()
                    .position(pos)
                    .title(myMarker.getNumero())
                    .icon(BitmapDescriptorFactory.fromBitmap(createDrawableFromView(this, view))));
            UpvMarkersHashMap.put(marker, myMarker);
        }
    }
}
```

Figura 66 Método que dibuja los marcadores en el mapa

Para el mapa de la Universitat Politècnica de València se han creado unos marcadores especiales donde se muestra el número de edificio.

Para crear estos marcadores se ha utilizado un layout de un ImageView con un TextView que crea la vista. Esta vista después se transforma a bitmap (Imagen de bits) que es lo que se muestra en el proceso final. Para no dar lugar a dudas, se explica el proceso a continuación.

- Paso 1: Se crea una vista con la imagen del marcador donde se introducirá el número del edificio:

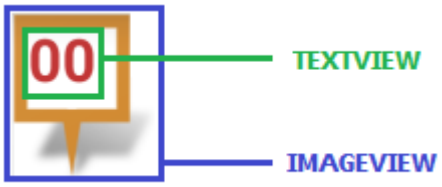

Figura 67 Layout marcador edificio

- Paso 2: Se infla la vista, se le asigna el valor del número del edificio a tratar y se convierte a imagen:

```
public static Bitmap createDrawableFromView(Context context, View view) {
    DisplayMetrics displayMetrics = new DisplayMetrics();
    ((Activity) context).getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
    view.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
    view.measure(displayMetrics.widthPixels, displayMetrics.heightPixels);
    view.layout(0, 0, displayMetrics.widthPixels, displayMetrics.heightPixels);
    view.buildDrawingCache();
    Bitmap bitmap = Bitmap.createBitmap(view.getMeasuredWidth(),
            view.getMeasuredHeight(), Bitmap.Config.ARGB_8888);

    Canvas canvas = new Canvas(bitmap);
    view.draw(canvas);

    return bitmap;
}
```

Figura 68 Método que crea un bitmap de un View

- Paso 3: Se obtiene la imagen bitmap y se asigna al marcador.

```
Marker marker = mMap.addMarker(new MarkerOptions()
        .position(pos)
        .title(myMarker.getNumero())
        .icon(BitmapDescriptorFactory.fromBitmap(createDrawableFromView(this, view))));
```

Figura 69 Creación de un marcador con el icono generado

6.5 Opciones

6.5.1 Diseño de interfaces

La última pantalla de la aplicación es la correspondiente a las opciones. Desde esta pantalla el usuario puede acceder a las configuraciones de la aplicación.

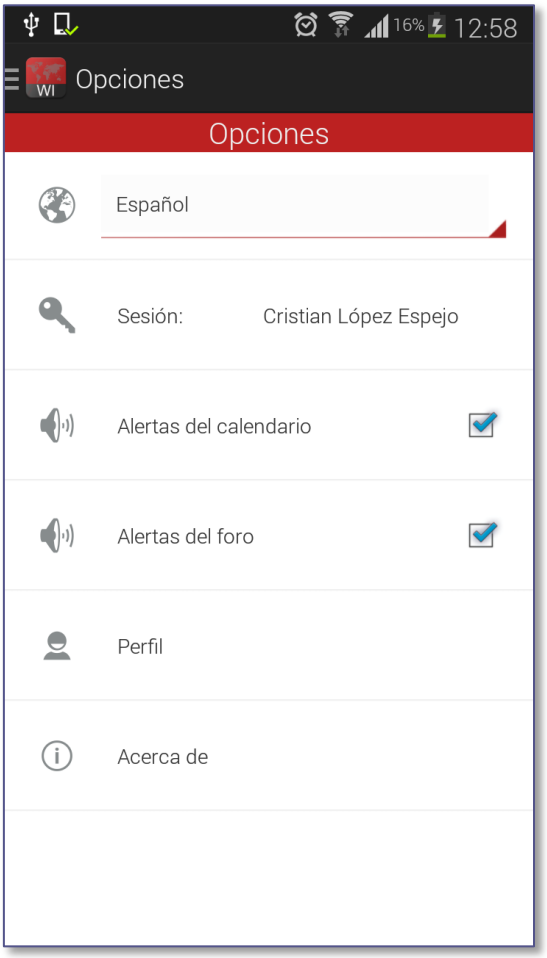

Figura 70 Fragment de opciones

Como podemos observar, desde la pantalla de opciones se puede cambiar el idioma de la aplicación, teniendo dos opciones:

- Español
- English

También tenemos un elemento de sesión. Este ítem permite tanto cerrar sesión como iniciar sesión. Si cerramos sesión se borran todos los datos guardados en la aplicación siendo ese usuario.

Desde opciones también podemos activar o desactivar las alarmas tanto del foro como del calendario.

Tenemos otro ítem que es perfil, la función de la cual es cambiar la foto de perfil para el foro y ver la actividad que has tenido en el foro. Esta funcionalidad es parte de la segunda parte del proyecto conjunto.

En acerca de accedemos a información relativa a los creadores de la aplicación.

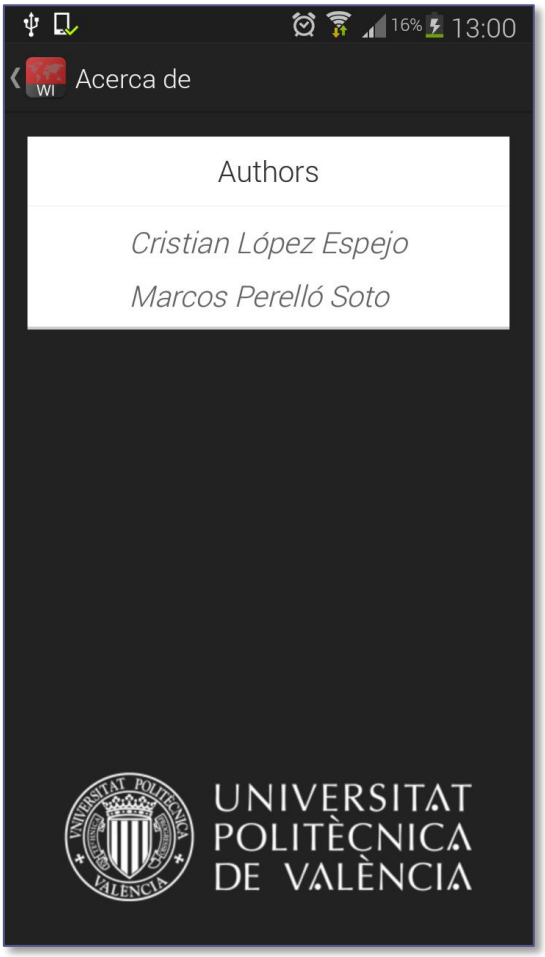

Figura 71 pantalla de acerca de

6.1.1 Aspectos técnicos

Idioma

Para el cambio de idioma, al seleccionar correctamente el idioma que se quiere seleccionar, la actividad se reinicia teniendo como idioma por defecto el ítem seleccionado en el spinner de idioma.

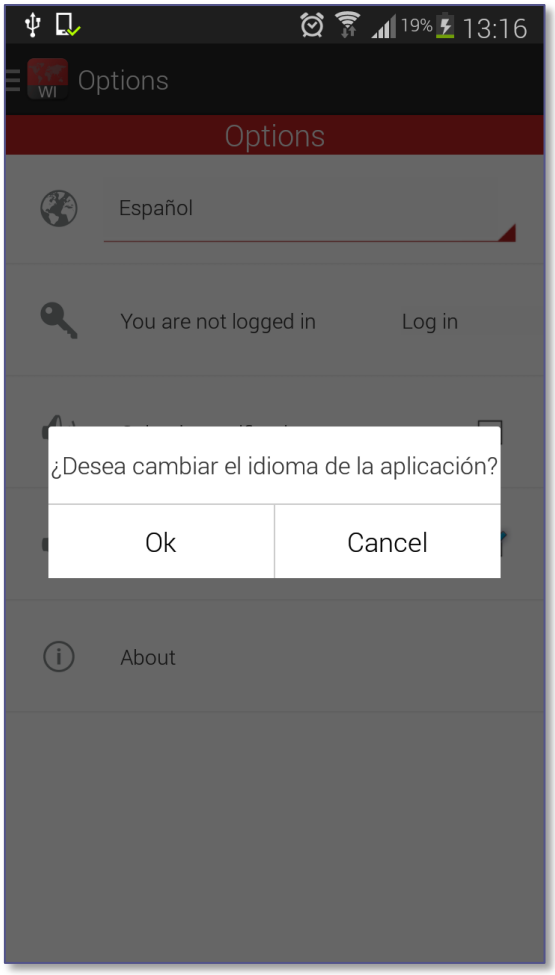

Figura 72 Dialogo confirmación de cambio de idioma

Para cambiar el idioma, se recrea la actividad como se muestra a continuación:

```
dialog.setOkListener((view) -> {
        if (spinner.getSelectedItemPosition() == 0) {
            Locale locale = new Locale("en_US");
            Locale.setDefault(locale);
            Configuration config = new Configuration();
            config.locale = locale;
            getActivity().getApplicationContext().getResources().updateConfiguration(config, null);
            getActivity().recreate();
        } else if (spinner.getSelectedItemPosition() == 1) {
            Locale locale = new Locale("es_ES");
            Locale.setDefault(locale);
            Configuration config = new Configuration();
            config.locale = locale;
            getActivity().getApplicationContext().getResources().updateConfiguration(config, null);
            getActivity().recreate();
        }
        dialog.dismiss();
});
```

Figura 73 Cambio de idioma de la aplicación

Sesión

El elemento de sesión cambia de apariencia según hayas iniciado sesión o no. Si has iniciado sesión, al pulsarlo se cerrara se sesión, y si no has iniciado sesión, al pulsarlo aparecerá el dialogo de inicio de sesión (Figura 31).

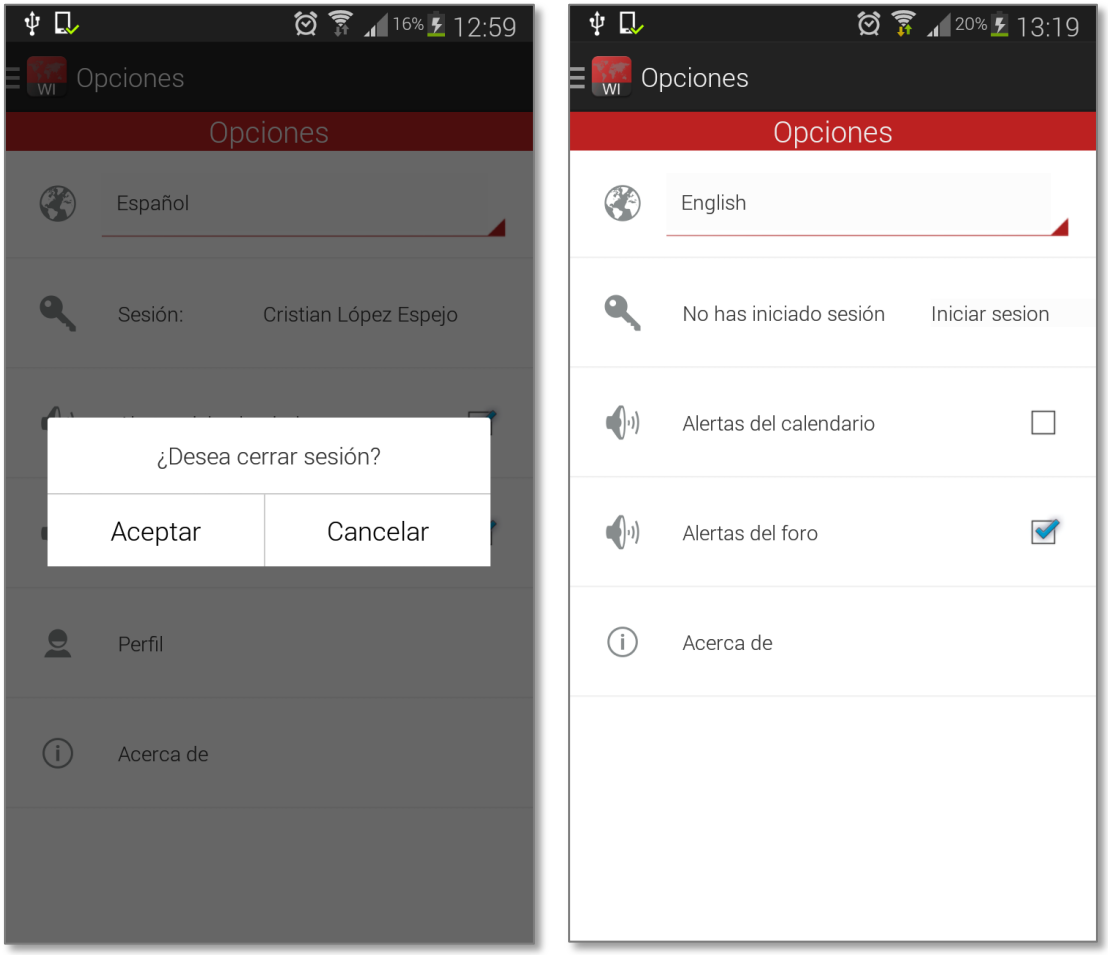

Figura 74 Pantalla de cerrar sesion (IZQ) y pantalla de opciones con sesión cerrada (DER)

Para cambiar el elemento según se haya iniciado sesión o no, se hace uso de una preferencia que permite saber si hay una sesión activa:

```
itemSesion.setId(R.id.sesion_opciones);
itemSesion.setOnClickListener((view) -> {
        if (!Preferencias.logeado(getActivity())) {
            if (icc.checkInternetConnection(getActivity())) {
                Intent i = new Intent(getActivity(), Activity_login.class);
                startActivityForResult(i,1);
            } else {
                Intent intent = new Intent(getActivity(), Activity_no_connection.class);
                intent.putExtra("case",
                        "To access to this feature, you need internet connection...");
                startActivity(intent);
            }
        } else {
            final ConfirmationDialog_Custom dialog = new ConfirmationDialog_Custom(getActivity(),
                    "¿Desea cerrar sesión?");
            dialog.setOkListener((view) -> {
                    new DesloguearTask(itemSesion).execute();
                    dialog.dismiss();
            });
            dialog.setCancelListener((view) -> {
                    dialog.dismiss();
            });
            dialog.show();
        }
});
```

Figura 75 Generar vista de sesión

En el primer 'If' se consulta si se ha iniciado sesión. Si no se ha iniciado sesión, aparecerá la pantalla de inicio de sesión (Activity_login).

Si se ha iniciado sesión, aparecerá un dialogo confirmando el cierre de la sesión.

Alarmas

Al desactivar el Checkbox de cualquiera de las dos alarmas ya sea del foro o del calendario, se cancelan las notificaciones que llegan a la aplicación.

```
if(Preferencias.getCalendarAlerts(getApplicationContext()))
mManager.notify(id, mBuilder.build());
```

Figura 76 Comprobación de alerta

Para las alarmas también se ha creado una preferencia que permite establecer las alarmas esté activado el Checkbox o no.

```
check.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
        if (b) {
            if (tipo.equals("Calendar")) {
                Preferencias.setCalendarAlerts(getActivity(), true);
            } else if (tipo.equals("Forum")) {
                Preferencias.setForumAlerts(getActivity(), true);
            }
        } else if (!b) {
            if (tipo.equals("Calendar")) {
                Preferencias.setCalendarAlerts(getActivity(), false);
            } else if (tipo.equals("Forum")) {
                Preferencias.setForumAlerts(getActivity(), false);
            }
        }
    }
});
```

Figura 77 Proceso de activación de alarmas

Si un usuario decide cancelar la alarma, una vez desactivada la aplicación cancela el PendingIntent con cierto ID que se ha creado anteriormente:

```
int id = getUniqueID(actual);
PendingIntent pendingIntent = PendingIntent.getBroadcast(getActivity(), id, intent, 0);
pendingIntent.cancel();
actual.setAlertado(0);
eventos.set(position, actual);
mCheckedState[position] = false;
eventosConAlerta.remove(actual);
borrarUniqueID(actual);
```

Figura 78 Cancelación de la alarma del calendario

7. Pruebas

Para comprobar que la aplicación cumple los objetivos expuestos en la planificación, se desarrollaron varias pruebas para evaluar rendimiento en distintos dispositivos móviles y pruebas de aceptación para saber la opinión de los usuarios.

7.1 Pruebas de rendimiento

Las primeras pruebas ejecutadas sobre la aplicación corresponden a las de rendimiento. Para que estas pruebas sean más objetivas, se han ejecutado sobre diferentes dispositivos Android. Como se puede comprobar en la

Tabla 2 Dispositivos utilizados en las pruebasTabla 2, se han utilizado dispositivos de distintas gamas. El HTC Desire corresponde a una gama alta de móviles Android, el Samsung Galaxy S3 a una gama media-alta y el Orange Luno a la gama más discreta de los dispositivos Android.

	HTC Desire 816	Samsung Galaxy S3	Orange Luno
Procesador	Quad-core 1.6GHz	Quad-core 1.4 GHz	Dual Core 1.3 GHz
Memoria RAM	1.5GB RAM	1GB RAM	512MB RAM
Sistema Operativo	Android 4.4 KitKat	Android 4.3 JellyBean	Android 4.3 JellyBean
Pantalla	5.5 pulgadas	4.8 pulgadas	4 pulgadas

Tabla 2 Dispositivos utilizados en las pruebas

Se han realizado diversas pruebas de tiempo de respuesta en los diferentes móviles.

Para esta prueba se ha medido el tiempo de respuesta de distintas funcionalidades de la aplicación. Los resultados se pueden observar en la

- Por una parte se ha medido el tiempo de cargar de la aplicación, tanto el primer arranque como los demás. Para el primer arranque tenemos que tener en cuenta que la aplicación obtiene los recursos e introduce la información en la base de datos. Esto aumenta el tiempo de espera de arranque, pero una vez completado el primer arranque se disminuyen los tiempos de arranque. Si se mira desde esta perspectiva, aumentando el tiempo necesario para el primer arranque, se disminuye los tiempos posteriores de respuesta.
- La segunda prueba medida es la descarga de todos los eventos del Calendario. Realmente esta prueba depende de la velocidad de internet que se disponga, ya que solo se requiere descargar los eventos la primera vez que accedes al Calendario. Una vez completado el primer acceso, los eventos son obtenidos de la base de datos, donde se almacenaron en el primer arranque del Calendario.

- La tercera prueba corresponde al tiempo de acceso al foro de la aplicación. Esta funcionalidad, al requerir conexión a internet, también depende de la velocidad de la conexión. Como se puede aprecia, los tiempos de espera son mínimos y muy rápidos en móviles de gama baja.
- Por último, se ha medido el tiempo en cargar completamente los mapas. Debido a que en el primer arranque se procesan todos los marcadores de los mapas, en los siguientes accesos no se necesita procesamiento y el tiempo de respuesta depende de la calidad de procesamiento del dispositivo.

	HTC Desire 816	Samsung Galaxy S3	Orange Luno
Inicio y carga (primera vez)	6.2 segundos	6.8 segundos	8.1 segundos
Inicio y carga	1.1 segundos	1.3 segundos	1.5 segundos
Descarga de eventos de calendario	14 segundos	14.2 segundos	16 segundos
Listar temas del foro	0.8 segundos	0.9 segundos	1 segundo
Mostrar mapas	1.2 segundos	1.1 segundos	1.4 segundos

Tabla 3 Resultados pruebas de respuesta

7.2 Pruebas de satisfacción del usuario

Otro factor a analizar para saber si la aplicación cumple con los requisitos necesarios de cara al cliente es realizar unas pruebas de satisfacción.

La población utilizada para estas pruebas constaba de un equipo de 6 personas que enviaban comentarios y sugerencias para la aplicación. Para este fin se creó un grupo de Google+ cuya intención era poder probar la beta de la aplicación.

Para este cometido se ha realizado una encuesta para los usuarios de la beta de la aplicación, donde se preguntaban sobre las siguientes cuestiones:

- **La aplicación es fácil de entender para el usuario**

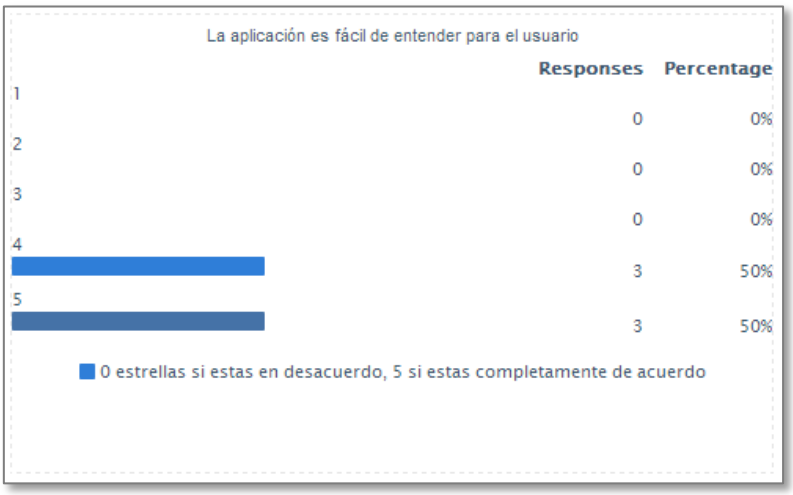

Figura 79 Pregunta 1 de pruebas de satisfacción

Como se puede observar la mayoría de los usuarios que probaron la aplicación quedaron satisfechos con la disposición de sus funcionalidades y su facilidad de uso.

- **La interfaz de la aplicación es atractiva y amigable**

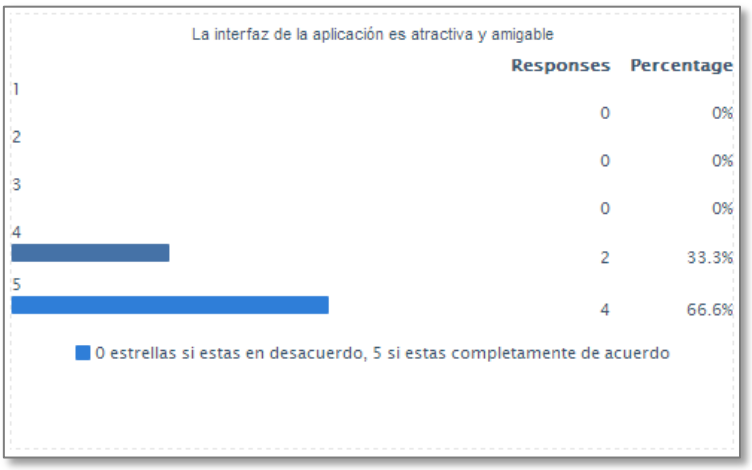

Figura 80 Pregunta 2 de pruebas de satisfacción

Como se puede observar en la Figura 80, los usuarios consideran el diseño de la interfaz de la aplicación atractiva y amigable. Se puede considerar que el patrón de diseño ofrecido en la aplicación ha gustado entre los usuarios ya que la aplicación parece atractiva y fácil de usar para los usuarios.

- **El tiempo de respuesta es correcto bajo ciertas condiciones**

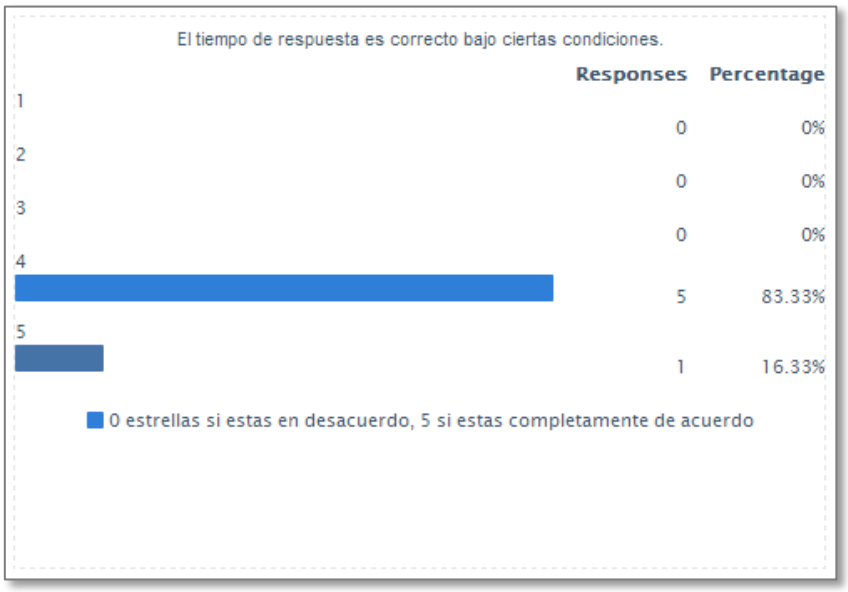

Figura 81 Pregunta 3 de pruebas de satisfacción

En esta pregunta pudimos comprobar como los usuarios consideraban notable el tiempo de respuesta. Gracias al *feedback* obtenido mediante la consola de Google Play, pudimos observar que una de las cosas que más preocupaban a los usuarios era el tiempo que tardaba la aplicación en cargar el mapa cada vez que se requería.

El tiempo requerido para abrir un mapa aumentaba hasta los 3 segundos, un tiempo de respuesta decente pero no muy bueno. Gracias a estos avisos decidimos cargar los mapas la primera vez que se ejecutaba la aplicación y se consiguió reducir este tiempo alrededor de 2 segundos como se muestra en la Tabla 3.

- **En términos generales, la aplicación cumple su cometido y con buen rendimiento**

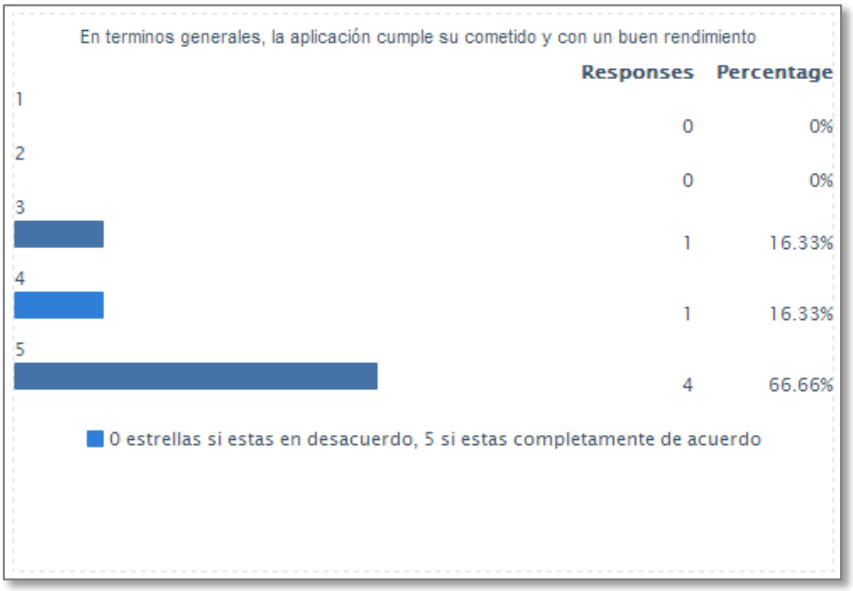

Figura 82 Pregunta 4 de pruebas de satisfacción

Observando esta pregunta podemos ver que los usuarios están contentos con la funcionalidad obtenida en la aplicación. Podemos ver que algún usuario ha valorado con un 3 el rendimiento de la aplicación. Esto es debido a lo comentado anteriormente sobre los tiempos de cargas de mapas ya resuelto.

Se podría decir que se ha obtenido un rendimiento y una cantidad de funcionalidades que todos los usuarios esperaban. Han valorado positivamente el tiempo de respuesta y el rendimiento por lo tanto podemos concluir que la aplicación cumple con los requisitos no funcionales expuestos en la fase de especificación.

- **¿Qué es lo que más le ha gustado de la aplicación?**

A modo de buscar un punto fuerte de la aplicación, se preguntó por lo que los usuarios creían más atractivo de la aplicación. Esto sirve al desarrollador que funcionalidad o que punto de la aplicación causa más impacto en los usuarios y mejorarlo.

Figura 83 Pregunta 5 de pruebas de satisfacción

Como podemos ver, las opciones más marcadas en la encuesta fueron la interfaz y las funcionalidades. Mediante el *feedback* se pudo observar que la funcionalidad más votada de la aplicación fue el foro seguido por la localización y la realidad aumentada.

La interfaz también ha causa muy buena impresión con sus colores y su patrón de diseño tan llamativo con animaciones y efectos de transición.

A modo de resumen, se puede decir con total libertad que se han cumplido todos los objetivos puestos para la aplicación, y que los usuarios que prueban la aplicación están satisfechos con las funcionalidades de la aplicación.

Se debe de tener en cuenta que durante la fase alpha de la aplicación, se puso a disposición de los usuarios de prueba de la alpha y la beta una versión de la aplicación. Esto ha permitido tener un seguimiento de las sugerencias de los usuarios y de los problemas que ocasionaba la aplicación.

8 Conclusiones

En este trabajo de final de grado se ha desarrollado una aplicación que permita la integración de los estudiantes de intercambio en la Universitat Politècnica de Valencia. Esta aplicación tenía la simple idea de ser desarrollada por y para dichos estudiantes que pueden tener mayor problema al llegar a la UPV. La idea del cliente era sencilla: disponer de una aplicación que permita acceder al calendario del alumno, localizar al alumno dentro de la UPV y Valencia, disponer de una herramienta de traducción para facilitar a los alumnos expresarse y comunicarse y la posibilidad del alumno de acceder a información de la Guía de Intercambio fácilmente.

De cara al desarrollo de la aplicación ha sido necesario aprender a desarrollar en esta plataforma móvil como lo es Android. Se contaba con una base sólida debido a que con anterioridad se han realizado cursos de este sistema operativo, pero se han tenido que investigar muchos apartados que eran desconocidos.

Se ha tenido que aprender conceptos sobre métodos de obtención de datos estáticos. Muchos de estos datos obtenidos se han tenido que extraer mediante acceso a la web desde el dispositivo móvil ya que no se ha dispuesto de un método facilitado por la UPV para acceder a los datos de la universidad.

Toda la aplicación está desarrollada partiendo de la base de que los usuarios de esta, en este caso los estudiantes de intercambio (sobre todo los extranjeros), no disponen de conexión a internet mediante la cual obtener todos los datos, por eso se ha optado por guardar los datos de forma local e ir actualizándolos manualmente.

Se han programado varias reuniones con el cliente a lo largo del desarrollo del proyecto, demostrando el cliente su satisfacción por cómo ha ido quedando la aplicación. Podríamos decir que se han cumplido con un gran porcentaje las expectativas nombradas al principio de la fase de análisis.

A lo largo del desarrollo del proyecto se han encontrado muchos problemas y muchos cambios en las implementaciones. Como puede ser el intento de uso de mapas internos que al final no se llevó a cabo debido a que no obtuvimos una respuesta por parte de Google. También hubo problemas con los calendarios de la UPV ya que se tenían que obtener manualmente y no se precisaba experiencia necesaria para poder realizar un parseador de un archivo HTML ni tampoco conocimiento sobre JSON e ICAL.

Hay que comentar que la planificación del proyecto se ha cumplido completamente salvo con algunas variaciones casi inapreciables. Debido a que se ha cumplido con las exigencias del cliente y con la planificación se puede decir que el proyecto ha sido un éxito y estoy muy satisfecho con el resultado obtenido. Personalmente este proyecto me ha ayudado mucho a aprender sobre Android y sobre el parseo de HTML que anteriormente desconocía. Ha sido una experiencia gratificante que me ha enseñado a trabajar como grupo de desarrollo software.

9 Mejoras futuras

Para organizar las ideas que pueden servir para una extensión del proyecto o simplemente una mejora de la aplicación, vamos a dividir las mejoras en las partes que se divide la aplicación.

9.1 Inicio

Para la pantalla de inicio desde el principio se quería añadir información relativa a la OPII y crear un enlace con ayuda e información disponible en la web. Esto sería interesante para ofrecer ayuda fácilmente y cómodamente a los estudiantes de intercambio.

Una opción también interesante sería que la pantalla de inicio contemplara la última actividad reciente en el foro, y los temas más interesantes y útiles votados por la comunidad.

9.2 Calendario

Durante el desarrollo de la aplicación se han contemplado muchas posibilidades de implementar el calendario. Como mejora principal del calendario, se debería poder acceder al calendario de exámenes del alumno, no solo al horario de clases. Esta funcionalidad estaba pensada hacerla, pero por falta de tiempo se tuvo que dedicar tiempo a otras funcionalidades que se consideraban más necesarias. También sería una mejora importante cambiar la vista del calendario por algo más intuitivo que una lista de eventos.

9.3 Información

La información es el apartado más mejorable de la aplicación. Aunque se haya introducido los apartados que se han creído más importantes de la Guía de Intercambio, se queda un poco vacío en cuanto a información de Valencia y fiestas de esta. En las próximas versiones se debería integrar esta información.

9.4 Localización

En este apartado hay tantas posibilidades de implementación que existen muchas mejoras. Se ha obtenido unos mapas sólidos y repletos de información, por tanto no se necesita cambiar estos mapas. Aunque no se tengan que cambiar los mapas existentes, se ha planteado la necesidad de añadir nuevos mapas con información totalmente nueva a la existente. Mapas con bares, cafeterías, integración con Foursquare… Existen muchas posibilidades debido a la variedad de posibilidades que se obtiene con los mapas de Google. La integración con Foursquare y otras aplicaciones de este tipo se ha creído lo más interesante.

Glosario

SDK - Software Development Kit (Kit de desarrollo de Software) es un conjunto de herramientas que permiten crear aplicaciones para cierto sistema.

Máquina virtual Dalvik - Dalvik es la máquina virtual que utiliza la plataforma para dispositivos móviles Android. Dalvik ha sido diseñada por Dan Bornstein con contribuciones de otros ingenieros de Google.

IDE - Un entorno de desarrollo integrado, llamado también IDE (integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación.

UML - Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad

TFG - *Trabajo final de grado*

iCAL – iCalendar (conocido como iCal por la aplicación de Apple) es un estándar para el intercambio de información de calendarios.

JSON - JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos.

POI - Un punto de interés o "PDI" (en inglés POI), es un punto de ubicación específica que alguien puede encontrar útil o interesante.

Bitmap - Una imagen en mapa de bits o Bitmap, es una estructura o fichero de datos que representa una rejilla rectangular de píxeles o puntos de color, denominada matriz, que se puede visualizar en un monitor, papel u otro dispositivo de representación.

UPV – *Universitat Politècnica de València*

SDM - *Soluciones informáticas para dispositivos móviles (Asignatura)*

RSS - RSS son las siglas de Really Simple Syndication, un formato XML para sindicar o compartir contenido en la web.

OPII - *Oficina de Programas Internacionales de Intercambio (UPV)*

HTTP - Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web.

ICS - Extensión de los archivos tipo iCalendar.

URL - Un localizador de recursos uniforme o URL —siglas en inglés de uniform resource locator— es un identificador de recursos uniforme (URI) cuyos recursos

referidos pueden cambiar, esto es, la dirección puede apuntar a recursos variables en el tiempo.

XML - XML, siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

EMT - *Empresa Municipal de Transporte*

UV – *Universidad de Valencia*

Bibliografía

[1]

DICE. (s.f.). ***Google's Android Studio vs. Eclipse: Which Fits Your Needs?***

Obtenido de http://news.dice.com/2014/03/19/googles-android-studio-vs-eclipse-fits-needs/

[2]

Google Android Developers. (s.f.). ***Android Studio*****.**

Obtenido de https://developer.android.com/sdk/installing/studio.html

[3]

Google developers**. (s.f.).**

Obtenido de

https://developer.android.com/about/dashboards/index.html?utm_source=ausdroid.net

[4]

Google Play. (s.f.). ***El Mundo en Google Play*****.**

Obtenido de https://play.google.com/store/apps/details?id=com.gi.elmundo.main

[5]

Google Play. (s.f.). ***Libros en Google Play*****.**

Obtenido de https://play.google.com/store/books?hl=es

[6]

Google Play. (s.f.). ***UPV en Google Play*****.**

Obtenido de https://play.google.com/store/apps/details?id=es.upv&hl=es

[7]

IDC. (s.f.). ***IDC*****.**

Obtenido de http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[8]

Microsiervos. **(s.f.).**

Obtenido de http://www.microsiervos.com/archivo/gadgets/casi-tantos-smartphones-como-ordenadores-en-el-mundo.html

[9]

***Open Handset Alliance*. (s.f.).**

Obtenido de http://www.openhandsetalliance.com/

[10]

***Paid Content*. (s.f.).**

Obtenido de http://paidcontent.org/tech/419-googles-strong-mobile-related-patent-portfolio

[11]

Web BOUML. (s.f.). *Download - BOUML*.

Obtenido de http://www.bouml.fr/download.html

[12]

Web DBVis. (s.f.). *DbVisualizer: Database Management Software Tools*.

Obtenido de http://www.dbvis.com/

[13]

Web GIMP. (s.f.). *The GNU Image Manipulation Program*.

Obtenido de http://www.gimp.org/

[14]

Web NinjaMock. (s.f.). *Ninkamock - free tool for mobile app wireframes and website designing*.

Obtenido de http://ninjamock.com/

Anexos

ANEXO 1: Obtención API de Google Maps

Para implementar los mapas de google en la aplicación se han seguido unos pasos necesarios para poder acceder a la API de Google Maps v2.

Primero se ha tenido que añadir la librería de Google Play Services al SDK de Android para poder acceder a las clases y vistas de las que se componen.

Después de esto, se ha tenido que acceder a la consola de desarrollador de Google y crear el proyecto Welcome Incoming. Una vez creado el proyecto, se debe añadir todas las APIs que se van a utilizar en la aplicación para poder utilizarlas.

Una vez añadida la API de Google Maps para Android v2 al proyecto en la consola, se necesita las credenciales para usarla. Aquí entra en juego el certificado SHA1 de la aplicación que estamos desarrollando, ya que mediante este SHA1 y el nombre del paquete de nuestra aplicación, se genera una API que podremos utilizar.

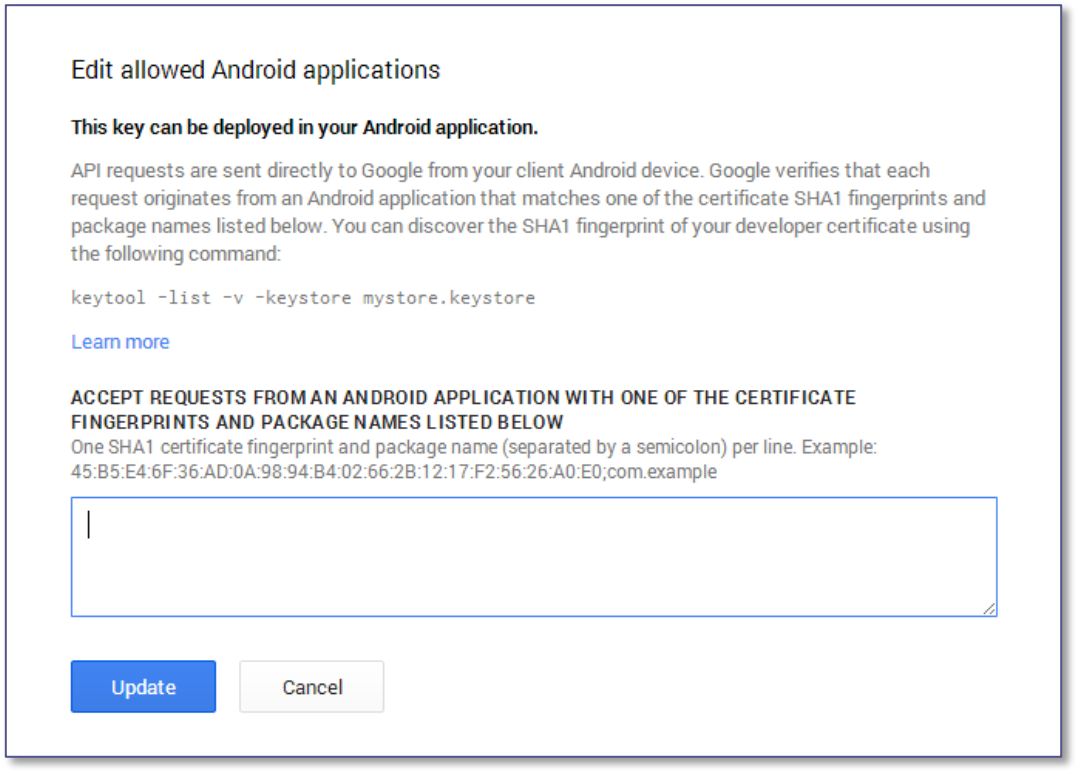

Figura 84 Obtencion de una clave api mediante SHA1

Esta clave API que obtenemos se debe de poner en el AndroidManifest.xml ya que no podríamos visualizar los mapas sin este elemento en el archivo.

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" />
```

Figura 85 Introduccion de apikey en el manifest