UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

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 LA UPV MEDIANTE USO DE HERRAMIENTAS ÚTILES

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: *Marcos Perelló Soto*

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 vendrán a estudiar a la Universitat Politècnica de València (UPV).

Para conseguir desarrollar esta aplicación, primero se estudió el alcance que podríamos tener en función del 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 está disponible en el Play Store de Google.

Palabras clave: Android, aplicación, ERASMUS, UPV, Valencia, foro, GCM, 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 university, 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.

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

Tabla de contenidos

Lista de Figuras

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. Este año el 87% de todos los dispositivos vendidos serán teléfonos y tabletas. Además, se prevé que la venta de tabletas superará por primera vez a la de los PC en 2015[1]. 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 o editoriales de libros permiten a los usuarios tener el contenido que siempre se ha vendido en papel en sus dispositivos móviles.

De la misma manera, la Universitat Politècnica de València es consciente de esta situación y por ello ha desarrollado aplicaciones móviles para el acceso a la información.

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 de carácter más general.

En este contexto, se veía necesario desarrollar una aplicación que permitiese localizar, integrar y hacer más cómoda la estancia en la universidad de aquellos 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 obtiene una serie de 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 entrega 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 Objetivos

El objetivo del proyecto es el desarrollo de una aplicación móvil para la plataforma Android que se distribuya mediante la tienda de aplicaciones Play Store, 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:

- Permitir al usuario identificarse de forma segura con sus credenciales de la UPV.
- Proporcionar al usuario la posibilidad de traducir idiomas de forma eficaz.
- Facilitar la integración del estudiante en el ámbito social y las relaciones con otros estudiantes.
- Permitir al usuario hacer uso de un foro con las funcionalidades básicas que este requiere.
- Notificar mediante el sistema de notificaciones de Android (GCM) cuando otros usuarios han realizado acciones en el foro que puedan resultar de interés.
- Mantener un rendimiento y un tiempo de respuesta de la aplicación decente.
- 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.

1.3 Estructura de la memoria

A continuación se describe brevemente cada uno de los capítulos que componen esta memoria:

- Introducción.

  En este capítulo se realiza la presentación del proyecto, se explica cual es la motivación por la cual se lleva a cabo, así como los principales objetivos que se desea lograr a través de él.

- Android:

  En este apartado se realiza una explicación sobre el sistema operativo Android, historia y estructura interna.

- Estado del arte:

  Este capítulo realiza una revisión sobre el estado en el que se encuentra Android en el mercado de los dispositivos móviles y compara nuestra aplicación con otras del mismo sector.

- Análisis de requisitos:

  Se exponen los requisitos necesarios que deben desarrollarse a lo largo del proyecto.

- Herramientas y tecnologías:

  En este apartado se exponen todas la herramientas, lenguajes de programación y técnicas que se han utilizado en el desarrollo de cada una de las capas del proyecto.

- Desarrollo:

  Se describe como se ha llevado a cabo el proceso de implementación de cada una de las partes del proyecto.

- Testeo y pruebas:

  Para asegurarnos que en el desarrollo se cumple con lo exigido en los requisitos, se realizaran ciertas pruebas y mostrarán sus resultados..

- Conclusiones:

  En este capítulo se realiza una valoración del proyecto y de los objetivos conseguidos.

- Trabajo Futuro:

  Se detallan las posibles evoluciones y mejoras que pueden llevarse a cabo en un futuro.

2. Android

En este capítulo se presenta Android, explicando que es y cómo se originó.
También se analizará el interior de la plataforma detallando su arquitectura.

2.1 ¿Qué es Android?

Android es un sistema operativo para dispositivos móviles desarrollado por Google. Está basado en GNU/Linux.

Esta plataforma es de código abierto y permite el desarrollo de aplicaciones por terceros (personas ajenas a Google). Para ello se proporciona un conjunto completo de APIs y herramientas de desarrollo, compilación, depuración y emulación.

La mayoría del código fuente de Android ha sido publicado bajo la licencia de software Apache, una licencia de software libre y código fuente abierto.

2.2 Breve historia

Android era un sistema operativo para móviles prácticamente desconocido hasta que en 2005 Google lo compró. Hasta noviembre de 2007 sólo hubo rumores, pero en esa fecha se lanzó la **Open Handset Alliance**, que agrupaba a muchos fabricantes de teléfonos móviles, chipsets y Google. Se proporcionó la primera versión de Android, junto con el SDK para que los programadores empezaran a crear sus aplicaciones para este sistema.

Aunque los inicios fueran un poco lentos, debido a que se lanzó antes el sistema operativo que el primer móvil, rápidamente se ha colocado como el sistema operativo de móviles más vendido del mundo, situación que se alcanzó en el último trimestre de 2010.

En febrero de 2011 se anunció la versión 3.0 de Android optimizada para tabletas en lugar de teléfonos móviles. Por tanto, Android ha transcendido los teléfonos móviles para trascender a dispositivos más grandes[2].

2.3 Arquitectura

Android es una plataforma software con el objetivo de abstraer el hardware y facilitar el desarrollo de apps para dispositivos. A continuación, observamos en la Figura 3, como de forma esquematizada aparecen los componentes principales del sistema[3].

Figura 1. Arquitectura Android

2.3.1 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 drivers para dispositivos[3].

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

2.3.2 Runtime

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 Kernel.
El runtime también contiene la mayoría de las librerías disponibles en el lenguaje Java[3].

Librerías nativas

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

2.3.3 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[3].
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).

2.3.4 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[3].

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

En este apartado, expondremos una visión general del estado en el que se encuentra la plataforma Android en el mercado de dispositivos móviles actual. También se hablará de otras aplicaciones similares y su posición frente a estas.

3.1 Tendencias en dispositivos móviles

Para el desarrollo de la aplicación se ha elegido la plataforma Android. Esta elección se fundamenta en el hecho de que el sistema operativo Android ha ido creciendo de forma vertiginosa sobre los demás desde el año 2010 como se puede observar en la Figura 2.

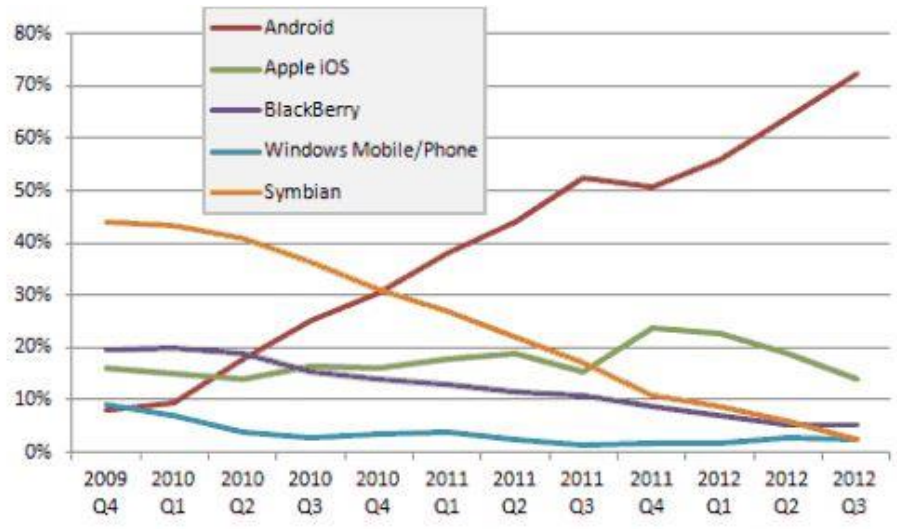

Figura 2. Crecimiento del S.O Android en los últimos años

En la actualidad, mas del 80% de los smartphones del mercado usan el sistema operativo de Google tal y como vemos en la Figura 3:

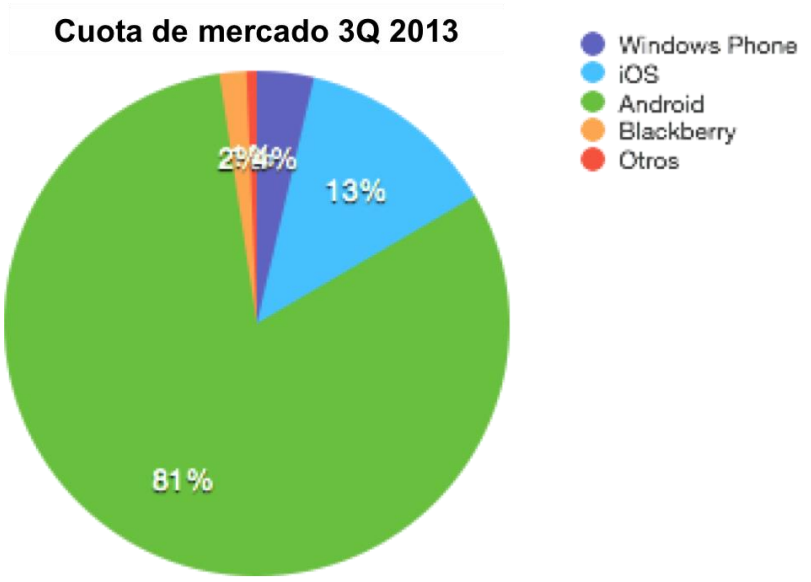

Figura 3. Cuota de mercado de los Smartphone en el 3 trimestre de 2013

Debido a que es el más utilizado, y la previsión es que seguirá siéndolo durante los próximos años, una aplicación desarrollada para este sistema operativo podrá ser instalada en más dispositivos y así permitir su uso a un mayor número de alumnos.

3.2 Otras aplicaciones

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 Play Store, se puede observar que hay muchas aplicaciones que podrían considerarse del mismo tipo que la que desarrollamos en este proyecto.

Para comprobar similitudes en Play Store, instalamos diversas aplicaciones del mismo ámbito que esta aplicación. Son las siguientes:

UPV – Universitat Politècnica de València.

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

Si bien tiene cosas en común como los horarios, la aplicación desarrollada en este proyecto tiene como usuarios a los estudiantes de intercambio, pese a que también puede utilizarse por alumnos de la UPV.

Figura 4. UPV. Universitat Politècnica de València

Figura 5. Estudia UJA. Universidad de Jaén

Estudia UJA– Universidad de Jaén.

Esta aplicación tiene cosas en común con UPV Welcome Incoming. Dispone de mapas de la universidad e información estática.

Pese a ello, es bastante pobre en cuanto a funcionalidad puesto que solo dispone de las dos citadas anteriormente.

Otro inconveniente es su fiabilidad, falla con frecuencia y se reinicia la aplicación como se puede observar en la figura 6.

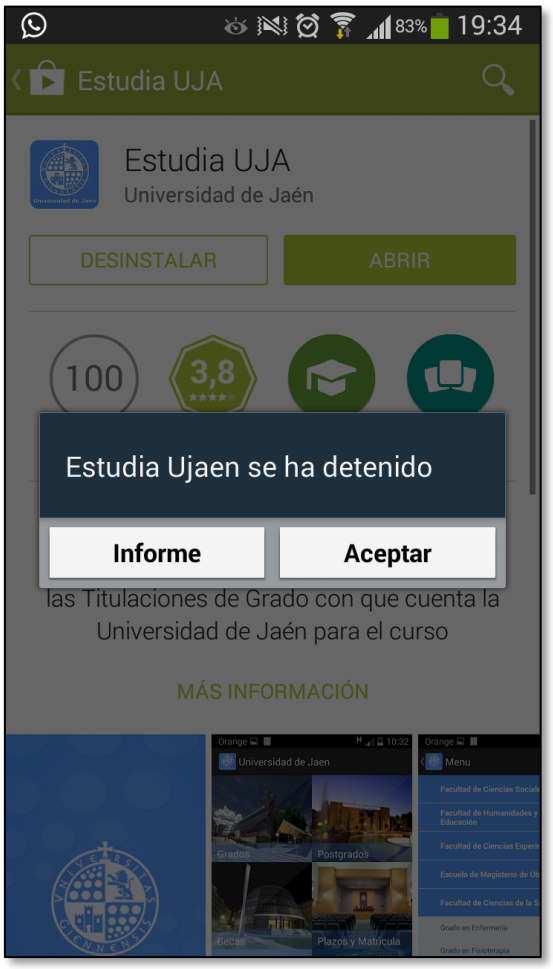

Figura 6. Fallo en la aplicación Estudia UJA.

Figura 7. Universidad de Valencia. Universidad de Valencia

Universidad de Valencia – Universidad de Valencia.

Aplicación bastante completa que contiene muchas de las funcionalidades que se desarrollan en este proyecto y además está aplicada a la misma localidad. No obstante, carece de otros elementos relacionados con la ciudad de Valencia como el uso de mapas o información estática de la ciudad.

La principal diferencia con esta aplicación es el público al cual va dirigida la aplicación, ya que en este caso se dirige a estudiantes de la UV en general.

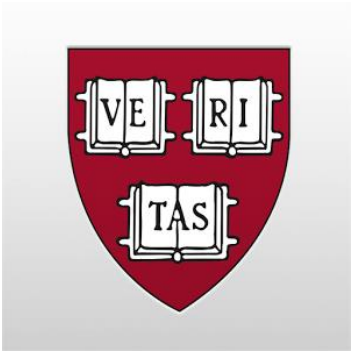

Figura 8. Hardvard mobile. Hardvard University

Hardvard mobile–Hardvard University.

Otra de las aplicaciones más parecidas a Welcome Incoming. Tiene información estática y localización.

El diseño no parece el más adecuado buscando la simplicidad y la comodidad. No se dispone de acceso a la Intranet.

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

Figura 9. University of Phoenix Mobile. University of Phoenix

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.

No cumple los patrones de diseño que propone Google para Android como se observa en la Figura 10.

Figura 10. Interfaz University of Phoenix Mobile

En la Tabla 1 podemos observar una comparativa entre las distintas características que UPV Welcome Incoming posee y las mismas, en las aplicaciones anteriormente expuestas.

	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, UPV Welcome Incoming dispone de unas funcionalidades que no se han visto en ninguna aplicación de este tipo como son el foro, la traducción o el servicio de notificaciones push. El foro 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 están disponibles al ser desarrollada de forma externa.

Aun contando con estas limitaciones, se trata de desarrollar una aplicación que cumpla los objetivos, manteniendo un rendimiento alto, estabilidad y contenido de fácil acceso y cómoda utilización.

3.3 Métodos de traducción

Para la realización de las funciones de traducción se ha realizado una investigación previa de las tecnologías, librerías y distintos servicios que podían usarse para cumplir esta función.

La primera opción era usar la API de Google Translate, pero se descartó esta opción por ser de pago (20 dólares por cada millón de palabras).

Posteriormente se encontró una solución gratuita para la API de Google que consistía en hacer peticiones GET a la dirección de Google tal y como lo hace la web. El resultado fue bastante satisfactorio pero se descartó debido a la posibilidad de que Google cambiase la dirección o los parámetros de la petición y a que no era una solución confiable.

Por último y definitivo, se ha escogido la API de traducción para java de Yandex[4]. Esta API ofrece un servicio de traducción gratuito a 40 idiomas y la función de detectar el idioma de origen.

4. Análisis de requisitos

Antes de pasar a describir el desarrollo de la aplicación, tenemos que numerar los requisitos necesarios que se han tenido en cuenta para desarrollarla. Para esto vamos a explicar en profundidad cada objetivo nombrado en la sección 1.2.

4.1 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.
- Calendario:
    - Mostrar al usuario calendarios de las asignaturas y eventos que le corresponden.
- 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 lograr dicho fin, debe mostrarse diferentes mapas:
        - **Mapa de Valenbisi**: En este mapa se muestra las estaciones de Valenbisi e información relacionada con esta, tal como las plazas totales y las bicis disponibles.
        - **Mapa de EMT**: Se muestra 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 mostrará información sobre tren, metro, bus, avión y taxi.

- Traducción:
    - Traducir con rapidez **frases comúnmente utilizadas** que ayuden a desenvolverse al estudiante en las situaciones más frecuentes.
    - **Traducir cualquier texto** que el estudiante requiera.

- Foro:
    - Permitir la posibilidad del estudiante de **iniciar, leer y participar en discusiones** que puedan resultar de interés.

- Notificaciones:
    - Notificar al usuario distintos eventos de interés como comentarios en el foro, comienzo de las clases, inicios de sesión, etc.

- Contacto:
    - Mostrar noticias de la OPI:
    - 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 11. Imagen de carga de "Welcome Incoming".

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

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

4.2 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 en la tienda Play Store. Debe funcionar sin conexión a Internet, ya que los alumnos que vienen a estudiar no tienen conexión 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 naturalidad 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.

4.3 Especificación de los requisitos

Debido al gran volumen de contenido que se requería para llevar a cabo todas las funcionalidades de una aplicación como UPV Welcome Incoming, se decidió dividir el proyecto en dos partes y así poder implementarlo en su totalidad.

Por un lado se agruparon las tareas que pueden usarse sin la necesidad de conexión a Internet como son la localización, calendarios e información estática de la aplicación.

Por otro lado, se agruparon **las tareas que si requieren de conexión a Internet.** Comprenden las relaciones sociales entre alumnos, traducción y notificaciones push. Estas tareas serán el foco del presente proyecto.

4.3 Casos de uso

Una vez se han establecido los requisitos específicos del proyecto, el primer paso es definir con claridad los casos de uso.

Los casos de uso se utilizan para especificar el comportamiento deseado de un sistema o subsistema. Su especificación describe el conjunto de **secuencias de acciones** que se lleva a cabo en el sistema para producir un resultado para un actor. También, capturan el comportamiento deseado del sistema, sin especificar como se lleva a cabo dicho comportamiento[5].

Un **actor** representa un conjunto coherente de roles que los usuarios de los casos de uso representan al interactuar con el sistema (tipos de usuarios). Normalmente representan a una persona, un dispositivo hardware u otro sistema al interactuar con el nuestro[5].

De acuerdo al presente proyecto encontramos un único actor, el usuario potencial de la aplicación. Se planteó la inclusión de otro actor distinto, un administrador de foro, pero finalmente se descartaron dichas funcionalidades para la versión actual.

En vistas a los requisitos anteriormente expuestos, se han representado gráficamente, mediante la herramienta Bouml Viewer[6], los siguientes casos de uso en notación UML exponiendo los nombres de los mismos con sus actores y relaciones.

Diagrama de caso de uso del subsistema de gestión del foro

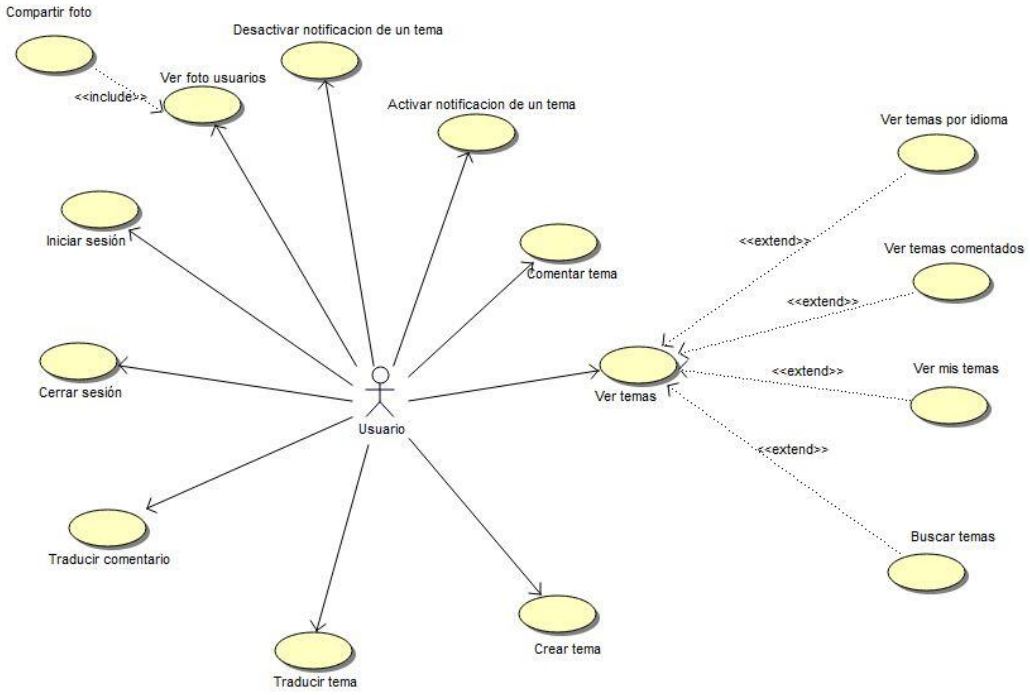

Figura 12. Diagrama de caso de uso del subsistema de gestión del foro.

En la Figura 12, se detalla cada una de las funcionalidades de las que dispone el usuario en el ámbito del foro.

Observamos, que el caso de uso “Compartir foto”, incluye a “Ver foto usuarios”, ya que el primero no sería posible sin la existencia del segundo.

Por otro lado, debemos destacar los 4 casos de uso que extienden la funcionalidad “Ver Temas” aportando nuevas características a dicho caso de uso.

Diagrama de caso de uso del subsistema traducción

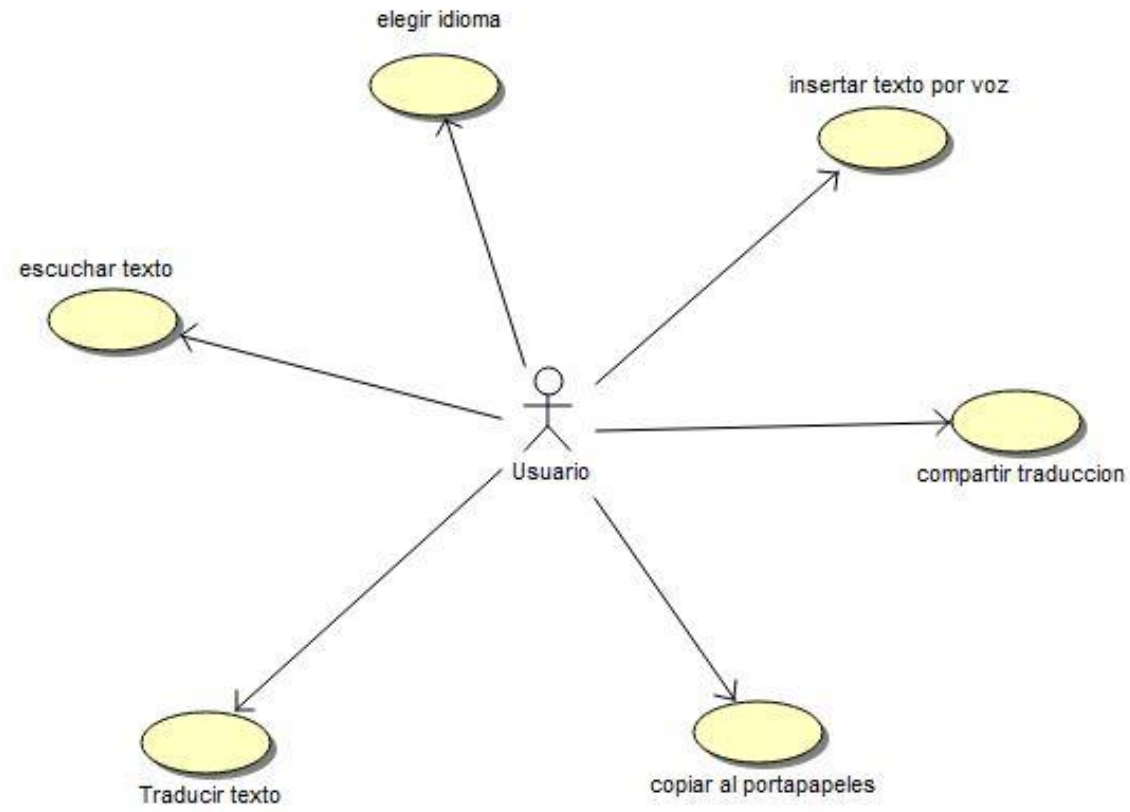

Figura 13. Diagrama de caso de uso del subsistema traducción.

Diagrama de caso de uso del subsistema opciones

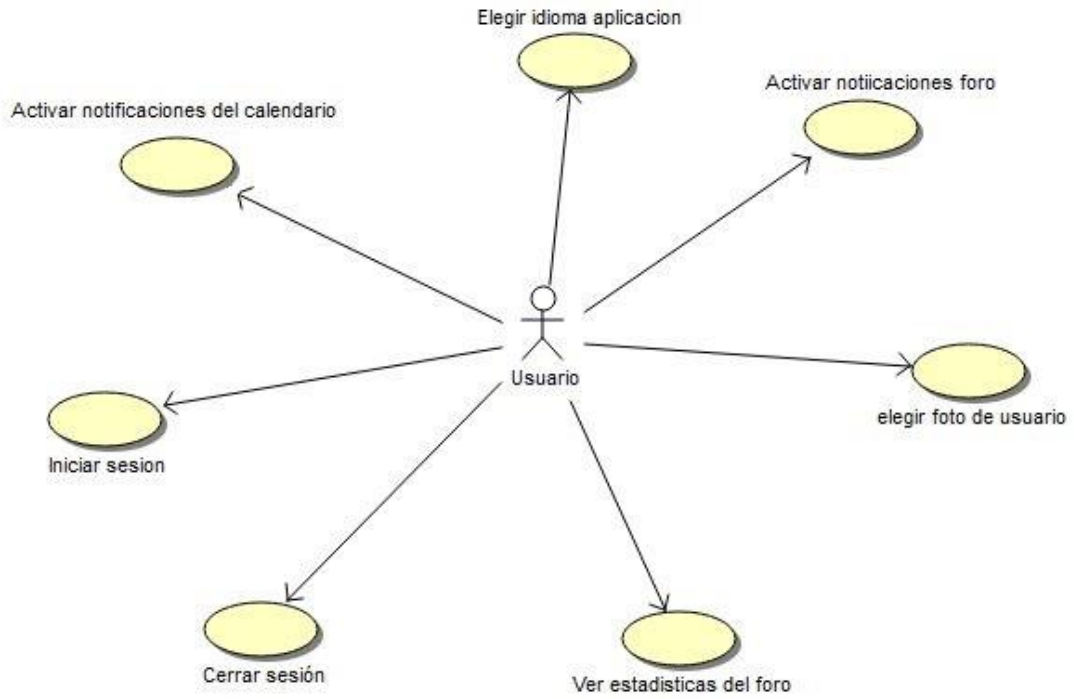

Figura 14. Diagrama de caso de uso del subsistema gestión de opciones.

4.4 Mockups

A continuación se muestran los bocetos de la interfaz que se diseñaron inicialmente para algunas pantallas significativas:

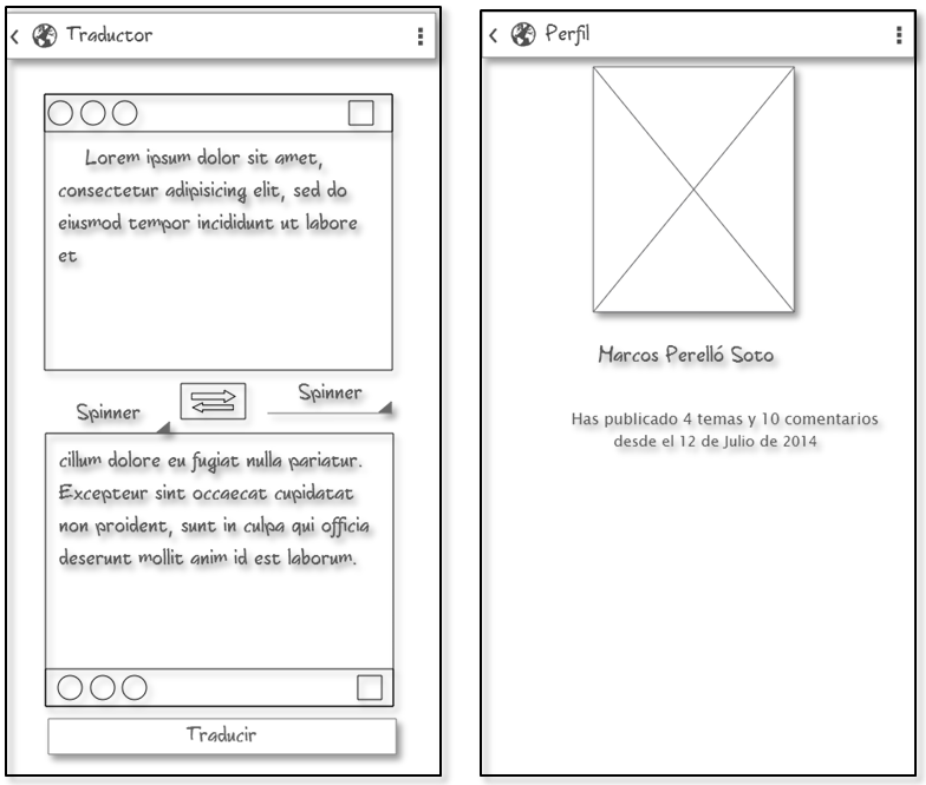

Figura 15. Mockups: Traductor y Perfil

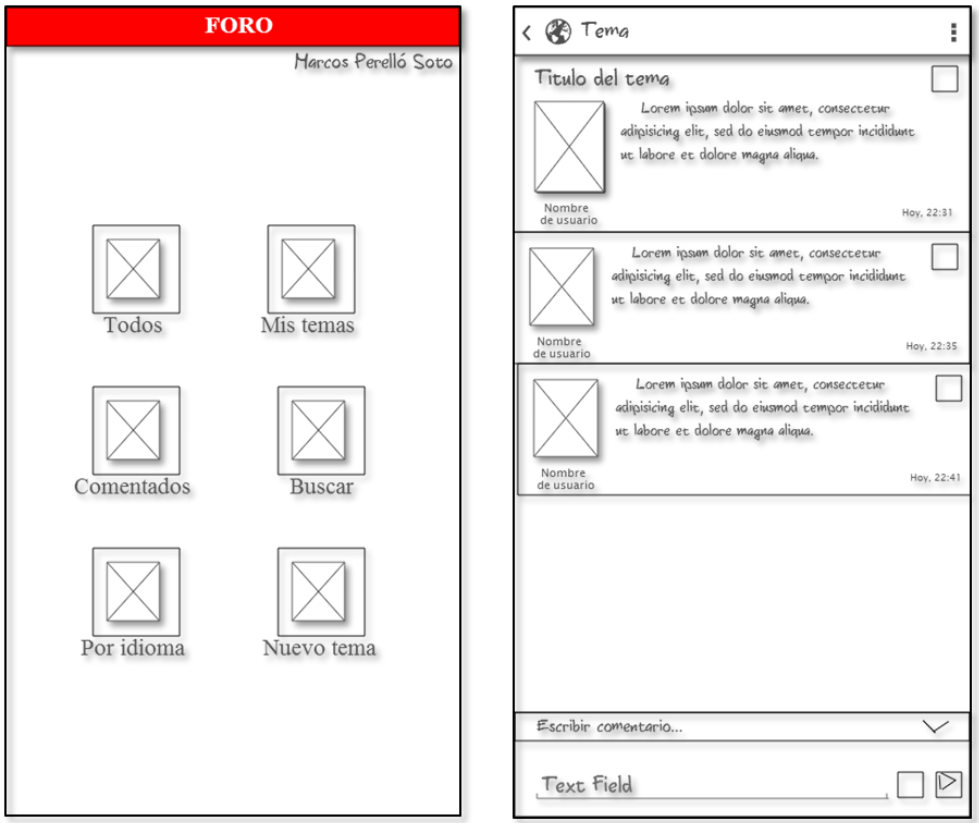

Figura 16. Mockups: Foro y Tema

5. Herramientas y Tecnologías

Con el objetivo de desarrollar e implementar las funcionalidades anteriormente descritas, es necesario el uso de diversas herramientas y programas que nos faciliten, en la medida de lo posible, la tarea a lo largo del desarrollo del proyecto.

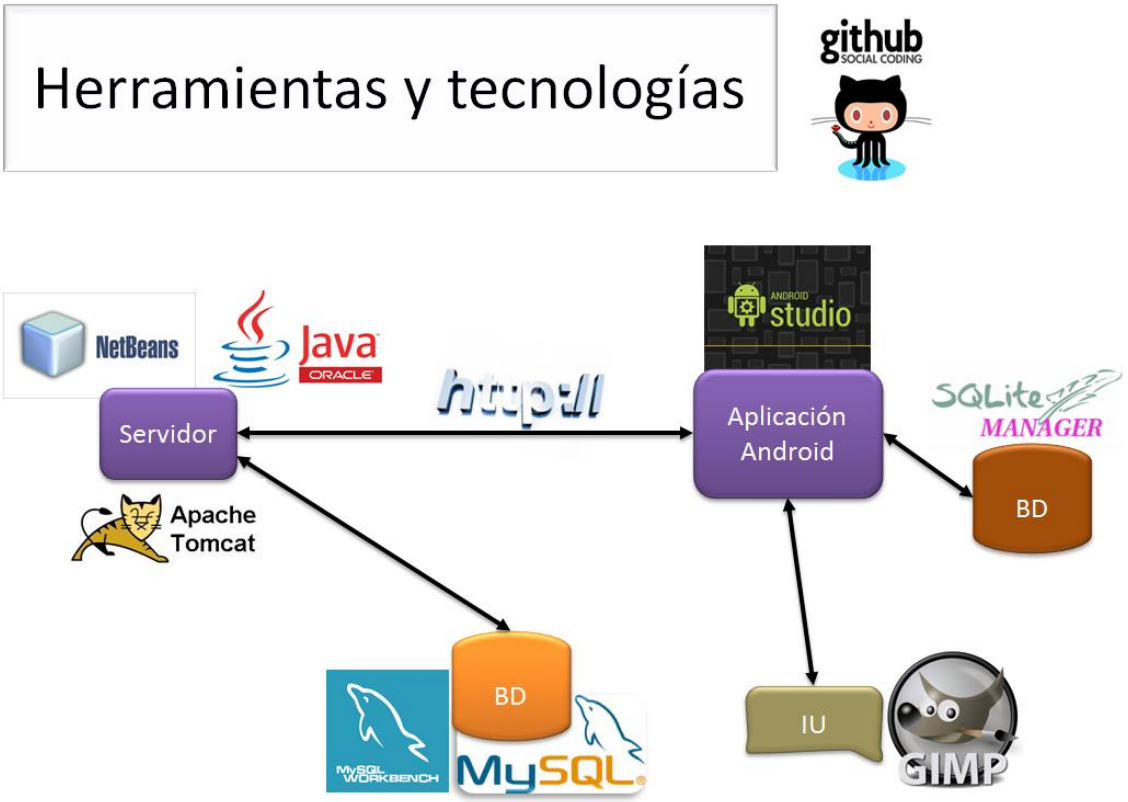

Figura 17. Herramientas y tecnologías utilizadas en el proyecto.

Como puede observarse en la figura 13, se han usado diversas herramientas y tecnologías para el desarrollo de las distintas capas de las que se compone el proyecto.

5.1 Servidor

- **Apache Tomcat 7**: Funciona como servidor web de servlets. Es de código abierto.

- **Netbeans 7. 4**: Es un IDE (Entorno de Desarrollo Integrado) de código abierto Java que se ha empleado para desarrollar la funcionalidad de la capa servidor.

- **Java 7**: Es un lenguaje de programación orientado a objetos.

- **MySQL 5.5.24**: Es un sistema de gestión de bases de datos relacional.

- **MySQL Workbench 6.1 CE**: Es una herramienta visual de diseño de bases de datos que integra desarrollo de Software, administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL[7].

5.2 Aplicación Android

- Android Studio 0.4.2: Es un nuevo entorno de desarrollo de Android basado en IntelliJ IDEA. Se trata del IDE oficial para desarrollo Android pese a que todavía se encuentra en fase beta[8].

- Gimp 2: Es un programa de edición de imágenes digitales de licencia pública. Se ha empleado en el desarrollo de iconos e imágenes de la interfaz de usuario de la aplicación.

- SQLite Manager 0.8.1: Esta herramienta se encarga de gestionar las bases de datos SQLite. Realmente se trata de una extensión del navegador Mozilla Firefox. Se ha empleado en el proyecto para diseñar y gestionar operaciones sobre la base de datos local de la aplicación.

5.3 Otros

- **Protocolo HTTP**: Hypertext Transfer Protocol, (en español, protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web. Define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web para comunicarse[9]. Es el empleado en el proyecto para establecer la conexión entre la aplicación Android y el servidor.

- **Github**: Es el gestor de versiones elegido para alojar el código de la aplicación Android en su versión gratuita. Hay que destacar que, debido al gran volumen de ficheros de la aplicación y a algunas incompatibilidades que surgieron a lo largo del desarrollo de la misma, se decidió abandonarlo e integrar de forma manual.

6 Desarrollo

Para el desarrollo de la aplicación se ha tenido en cuenta que debe ser una aplicación intuitiva y fácil de usar. Se ha divido la aplicación en diferentes bloques, los correspondientes a esta parte del proyecto son los siguientes:

- Inicio
- Traducción
- Foro
- Opciones
- Notificaciones Push

6.1 Inicio

La primera pantalla que observamos después de la ventana de carga de la aplicación es 'Inicio'.

Si es la primera vez que lanzamos la aplicación, se muestra una pantalla informativa que contiene instrucciones del uso de la aplicación como podemos ver en la figura 18.

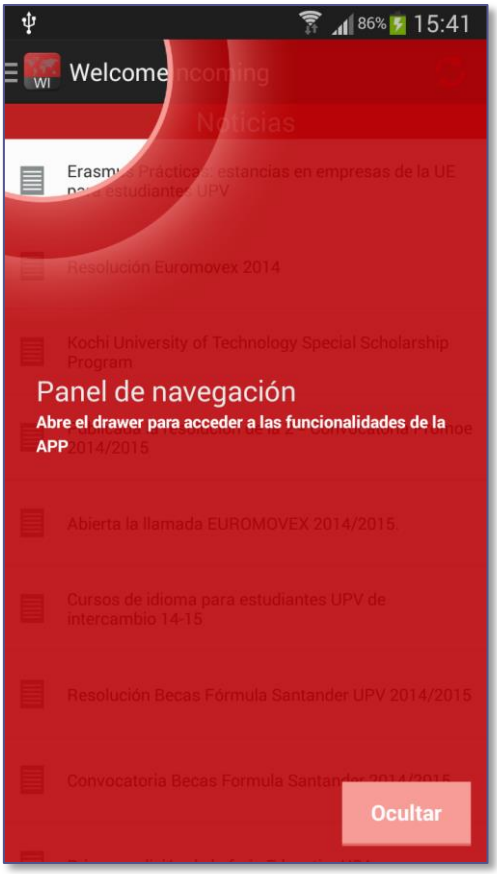

Figura 18. Pantalla tutorial

El fragment inicio contiene las últimas noticias de la OPI (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 últimas noticias.

6.1.1 Navigation Drawer

Con el fin de conseguir una aplicación intuitiva se ha implementado el patrón de diseño Navigation Drawer (Figura 19), Es un panel lateral que se desliza desde el borde izquierdo de la pantalla y muestra las principales opciones de navegación de la aplicación.

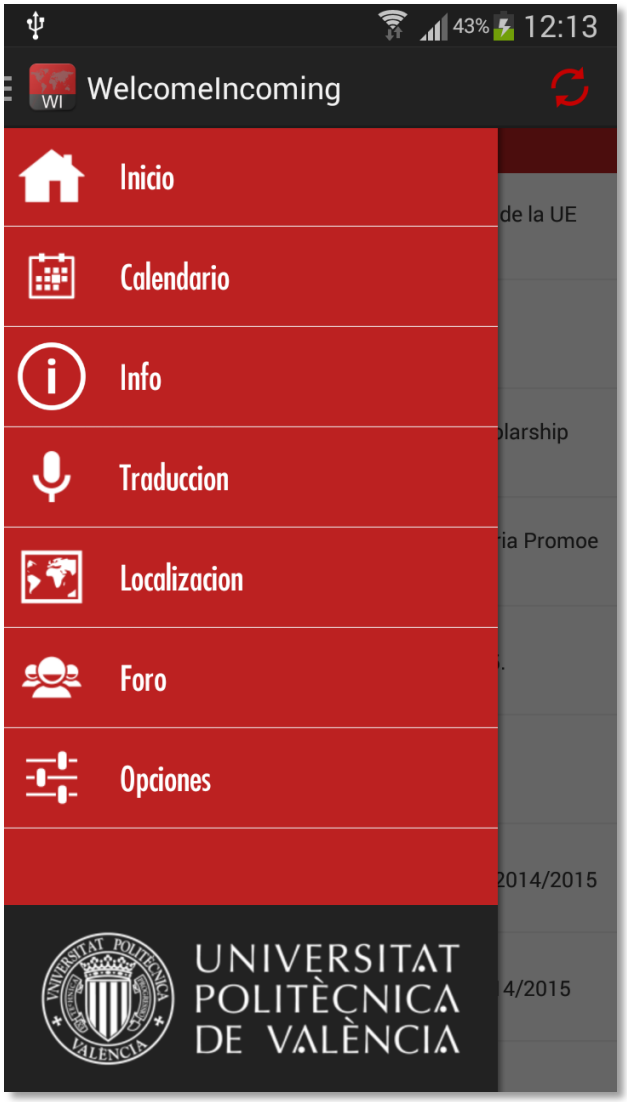

Figura 19. Navigation Drawer de la aplicación

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

El drawer por defecto, dispone de un adaptador que te permite mostrar los elementos pero solo con texto, por lo tanto hubo que crear nuevas vistas personalizadas como se muestra en la Figura 15.

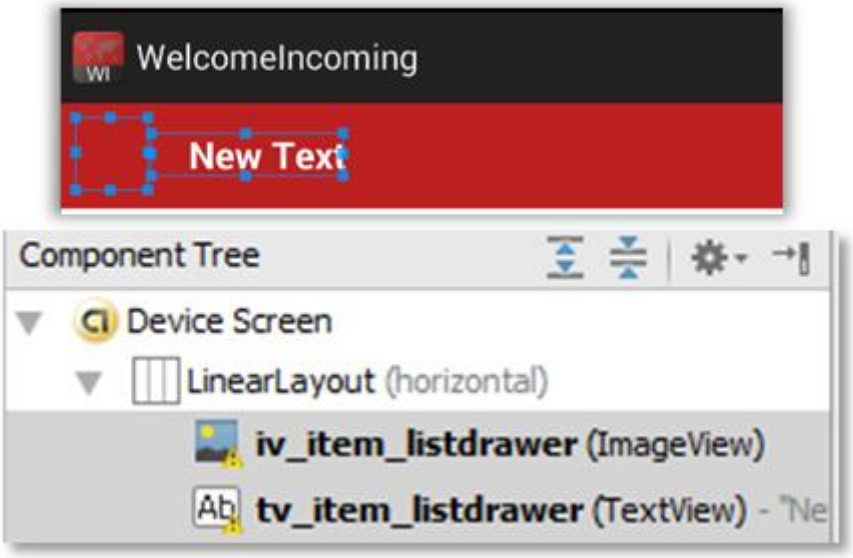

Figura 20. Elemento del drawer. Estructura.

En términos más específicos, cada ítem del drawer, está formado por un layout Horizontal que contiene un ImageView y un TextView respectivamente. Se observa claramente en la Figura 20.

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 conocer que es un Fragment:

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[10].

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 esté en marcha (algo así como una "sub actividad" que pueda reutilizar en diferentes actividades)[10].

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

Al pulsar sobre un ítem del drawer, el fragment principal de la aplicación se cambiará por el correspondiente fragment creado para cada uno de los ítems del drawer.

6.1.1.1 Diseño

En la figura 21 mostramos como seria la estructura del drawer. En la parte izquierda en azul, se observa el panel lateral que está oculto. En la parte derecha, en color morado vemos el contenedor de fragments donde se va a colocar el fragment de cada elemento del drawer.

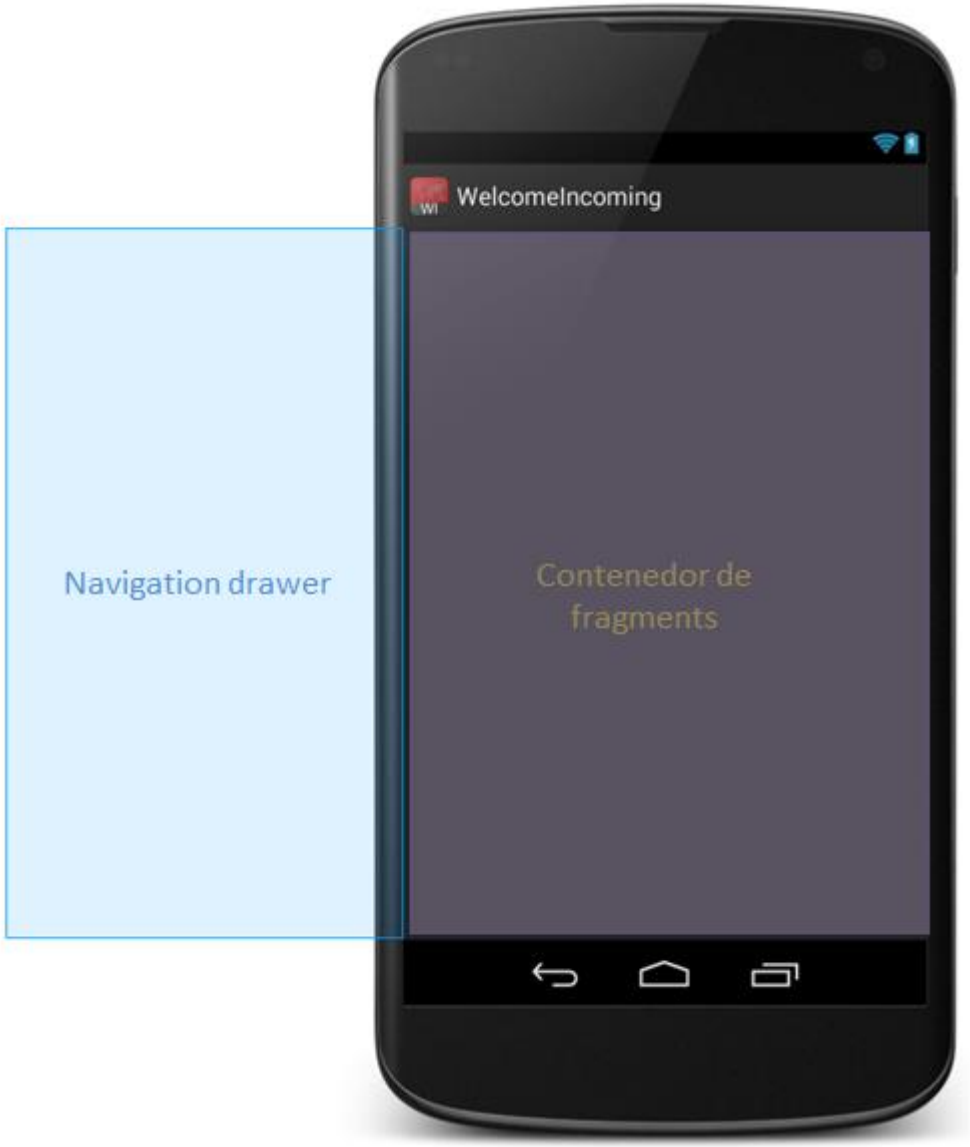

Figura 21. Diseño de la pantalla principal.

6.1.1.2 Carga de fragments

Al deslizar de izquierda a derecha de la pantalla se abre el drawer (También podemos abrirlo pulsando sobre el icono en la parte superior izquierda de la activity). Cada elemento del drawer que vemos en la figura 19 tiene un fragment asociado y ya creado, que ha sido diseñado de forma diferente a cada uno de los demás bloques de informacion.

Cuando pulsamos sobre un ítem provocaremos que el fragment asociado se cargue en el contenedor de fragments ocupando la totalidad de la pantalla.

En la figura 22 podemos ver que se muestra el 'Fragment_Inicio", el cual contiene una lista (ListView) que muestra noticias de la Oficina internacional de la OPI. Este fragment se ha cargado en contenerdor de fragments, como podemos ver en la figura 18, debido a que hemos pulsado sobre el ítem 'Inicio' del drawer. Si pulsásemos sobre cualquier otro ítem, sustituirá su fragment asociado al fragment cargado en este momento. El menú lateral queda por tanto oculto siempre que se carga un fragment debido a que se llama al método closeDrawer del drawer.

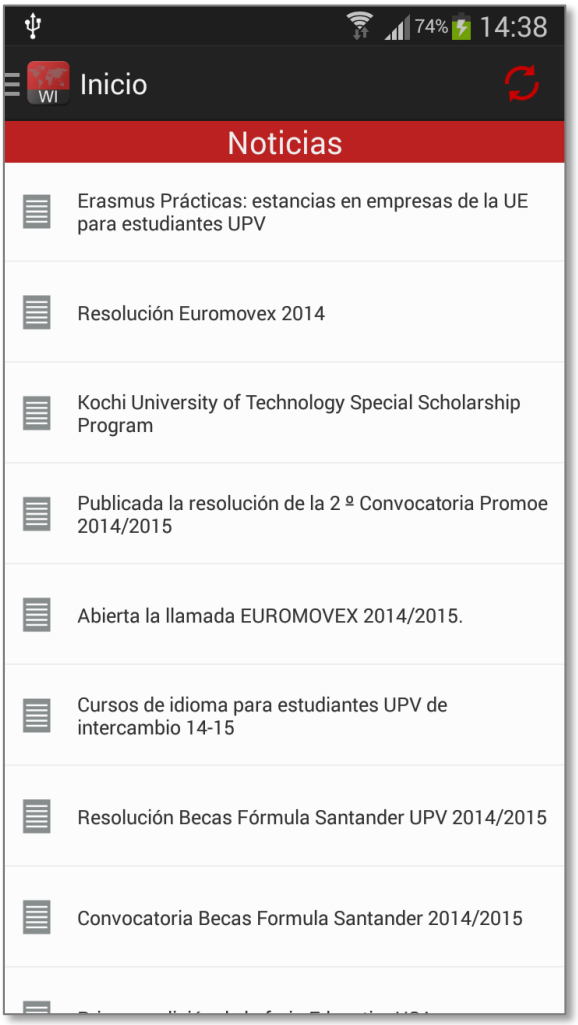

Figura 22. Ejemplo de fragment cargado en el contenedor.

6.2 Traductor

6.2.1 Interfaz de usuario

Para el desarrollo del traductor, teniendo en cuenta la especificación de requisitos realizada anteriormente, se ha decidido dividir las funcionalidades relacionadas con la traducción en dos ventanas distintas.

Por un lado, tenemos la funcionalidad de "**common phrases**" que son aquellas frases comunes y frecuentes que pueden ayudar al alumno de intercambio a comunicarse y a orientarse en español. Se entiende que, en caso de necesidad, se debe acceder de forma rápida a esta funcionalidad. Esto es motivo suficiente por el que esta será la primera ventana a la que accederemos al pulsar el icono de traductor en el drawer.

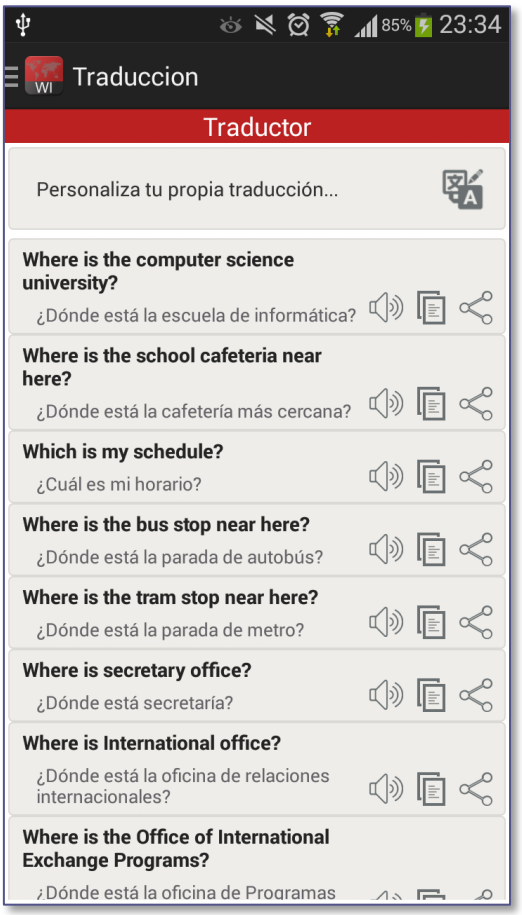

Figura 23. Fragment de traducción.

Esta ventana, es en realidad un fragment que dispone de una vista de scroll que contiene cada una de las frases a traducir. Estas frases están predefinidas en nuestro fichero strings.xml y se traducirán en el idioma seleccionado en el dispositivo. En la parte derecha de cada uno de los ítems se observan los iconos de escuchar, copiar al portapapeles y compartir la traducción de la frase.

Por otro lado, el usuario debe poder traducir frases personalizadas entre distintos idiomas. Para ello se ha realizado un traductor (Figura 24).

Figura 24. Activity Traductor.

El traductor es capaz de traducir texto entre 7 idiomas distintos (español, catalán, portugués, inglés, francés, alemán e italiano). En la estructura de la interfaz distinguimos tres componentes importantes:

- El texto a traducir
- El texto traducido
- El botón de traducir.

En la parte superior colocamos el botón que realizará la función de traducción para que sea fácilmente visible y accesible por el usuario. Este botón se habilita cuando introducimos texto a traducir y se deshabilita en caso contrario.

Debajo de este se encuentra la caja de texto en la que introduciremos el mensaje que queremos traducir. Esta caja contiene, en la parte superior derecha, una bandera que nos indicará el idioma del cual se va a traducir y un Spinner en la parte inferior derecha mediante el cual seleccionaremos el idioma. También disponemos de tres iconos situados en la parte superior izquierda y que nos permiten realizar las funcionalidades de escuchar el texto en el idioma seleccionado, introducir texto por voz y vaciar o eliminar el texto introducido.

Debajo de esta caja de texto encontramos el texto traducido distribuido de forma similar al anterior con la distinción de los iconos de función. Puesto que no tendría sentido poner los mismos que para el texto a traducir. Aquí nos encontramos con las funcionalidades de escuchar voz, copiar al portapapeles y compartir la traducción.

También hay que destacar el botón situado entre ambos Spinners de selección de idioma y que nos permite intercambiar los dos idiomas seleccionados cuando es pulsado.

6.2.2 Aspectos técnicos

6.2.2.1 Traducción

Como hemos comentado en el apartado 3.3, para realizar las funciones de traducción se ha escogido la API de traducción para java de Yandex[4]. Su uso es bastante simple, tan solo ha de importarse la librería (añadiendo la carpeta con las clases Java al proyecto), añadir la API KEY que se obtiene registrándose en la web de forma gratuita y llamando a los métodos deseados: traducir y detectar idioma como se muestra en la figura 25.

```
Translate.setKey(ApiKeys.YANDEX_API_KEY);

idiomaDetectado = Detect.execute(textoAtraducir);
textoTraducido = Translate.execute(textoAtraducir, idiomaDetectado, idiomaDestino);
```

Figura 25. Ejemplo de uso de la API de Yandex

Esta API, como el resto de las opciones que se plantearon, hace uso de internet. Puesto que Android no nos permite usar servicios de internet en el hilo principal se ha definido una clase AsyncTask a la que se le pasan los parámetros necesarios junto con un TextView en el cual se mostrará el texto traducido. Mediante el método *doInBackground* de la clase se llama a los métodos mencionados y se publica el resultado en el Textview en su método *onPostExecute*.

6.2.2.2 Entrada y salida de voz

Para introducir texto mediante voz se ha usado la API de Google SpeechToText y que requiere que el dispositivo móvil en el que se ejecute tenga instalada la aplicación de síntesis de voz de Google. Se mostrará una Activity de entrada de voz que lanzaremos mediante un Intent. Debemos de indicar el idioma de entrada mediante un objeto de la clase Locale en forma de String como extra del Intent y llamar a la actividad para esperar una respuesta (*startActivityForResult*) como vemos en la Figura 26.

```
Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, Language.getLocaleByName(spinnerFrom_selection).toString());
try {
    startActivityForResult(intent, RESULT_SPEECH);
    etTrad.setText("");
} catch (ActivityNotFoundException a) {
    Toast.makeText(getApplicationContext(),getString(R.string.noSoportaSpeechToText),Toast.LENGTH_SHORT).show();
}
```

Figura 26. Ejemplo de uso de la API SpeechToText (llamada)

La respuesta a la actividad la recibimos en forma de String en el método *onActivityResult* de la Actividad desde la que la hemos llamado. Y con el identificador que pasamos como parámetro de esa actividad al llamarla, en este caso: RESULT_SPEECH (variable de tipo entero de la clase).

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    switch (requestCode) {
        case RESULT_SPEECH: {
            if (resultCode == RESULT_OK && null != data) {

                ArrayList<String> text = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                editTextTraduccion.setText(text.get(0));
                editTextTraduccion.selectAll();
            }
            break;
        }

    }
}
```

Figura 27. Ejemplo de uso de la API SpeechToText (resultado)

Para la salida de voz se ha usado otra API de Google: TextToSpeech.

Su uso es más sencillo, tan solo debemos crear un objeto de la clase TextToSpeech, inicializarlo con el lenguaje que queremos escuchar y lanzar la llamada pasando en un String el texto a escuchar como parámetro.

```
private TextToSpeech ttobjTo;

ttobjTo = new TextToSpeech(getApplicationContext(),
        new TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int status) {
                if (status != TextToSpeech.ERROR) {
                    ttobjTo.setLanguage(new Locale("es", "ES"));//español de españa
                }
            }
        }
);

ttobjTo.setLanguage(Language.getLocaleByName(spinnerTo_selection));
```

Figura 28. Ejemplo de uso de la API TextToSpeech (inicialización)

Para hacer más agradable la satisfacción del usuario se cambia la imagen del botón que realiza esta acción (imagen de un altavoz transparente) por otra (altavoz sólido) mientras se está reproduciendo el sonido. Para ello se hace uso de un Listener de la clase que nos permite gestionar el inicio y fin de la reproducción de sonido.

Debemos hacer uso de un objeto Handler y su método post para poder así cambiar la interfaz de usuario y elegir la imagen que corresponda a inicio o fin.

```
Handler mHandlerStart = new Handler();
Handler mHandlerDone = new Handler();

ttobjTo.setOnUtteranceProgressListener(new UtteranceProgressListener() {
    @Override
    public void onStart(String s) {
        Runnable run = new Runnable() {
            public void run() {
                iv_bottom_altavoz.setImageDrawable((getResources().getDrawable(R.drawable.ic_action_altavoz_solido)));
            }
        };
        mHandlerStart.post(run);
    }
    @Override
    public void onDone(String s) {
        Runnable run = new Runnable() {
            public void run() {
                iv_bottom_altavoz.setImageDrawable((getResources().getDrawable(R.drawable.ic_action_altavoz_hueco)));
            }
        };
        mHandlerDone.post(run);
    }

    @Override
    public void onError(String s) { }
});
```

Figura 29, Ejemplo de uso de la API TextToSpeech (Listener)

Otros aspectos a destacar referentes a la traducción es la implementación de las funciones de compartir la traducción y copiar al portapapeles. Estas funciones se realizan de forma sistemática mediante el siguiente código (Figura 30 y Figura 31) donde “texto” es la variable tipo String que contiene la traducción:

```
android.content.ClipboardManager clipboard = (android.content.ClipboardManager)
        getApplicationContext().getSystemService(getApplicationContext().CLIPBOARD_SERVICE);

android.content.ClipData clip = android.content.ClipData.newPlainText("label", texto);
clipboard.setPrimaryClip(clip);

Toast.makeText(getApplicationContext(),
        getString(R.string.textCopyToClipboard), Toast.LENGTH_SHORT).show();
```

Figura 30. Ejemplo de implementación de la acción de copiar traducción al portapapeles

```
texto += "\n(Translated by upv welcomeincoming)";
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, texto);
sendIntent.setType("text/plain");

startActivity(Intent.createChooser(sendIntent,
        getResources().getText(R.string.compartirTraduccion)));
```

Figura 31. Ejemplo de implementación de la acción de compartir traducción

Cabe destacar, que toda la funcionalidad de la traducción en lo referente a los lenguajes, banderas de los lenguajes, objetos tipo Locale, etc se ha realizado apoyándose en una clase estática creada con ese propósito (Language) y que contiene arrays con los elementos mencionados y métodos que los gestionan como hemos podido observar en algunos ejemplos de código.

6.3 Foro

Es el sexto ítem del drawer y constituye la mayor parte del proyecto.

6.3.1 Modelo de dominio

Puesto que se ha considerado para esta primera versión de la aplicación la realización de un foro sencillo con las funcionalidades básicas se ha diseñado un modelo de dominio sencillo (Figura 32).

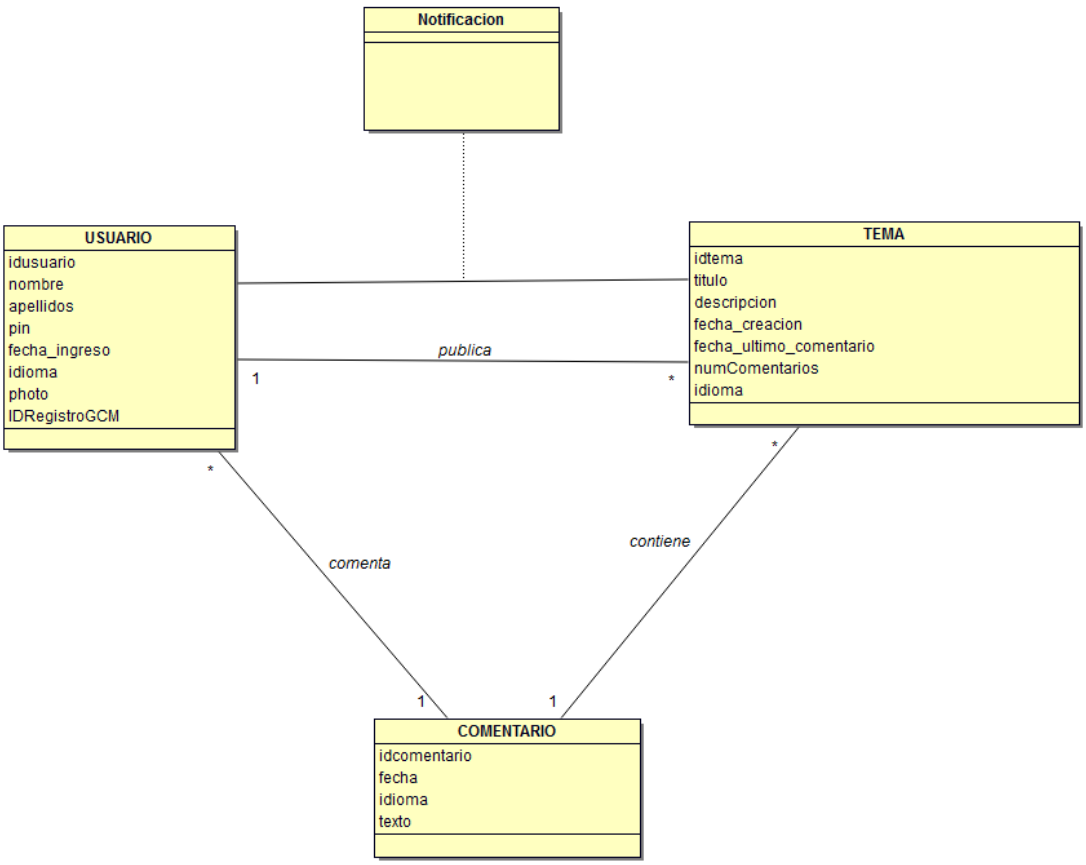

Figura 32. Modelo de dominio del foro

Como se puede observar, consta de tres tablas fundamentales: usuario, tema y comentario, relacionadas entre sí de la siguiente forma:

- Usuario-Tema: Un usuario puede, o no, publicar muchos temas mientras que un tema corresponde a un único usuario.

- Tema-Comentario: Un tema puede, o no, contener múltiples comentarios, mientras que un comentario debe pertenecer a un único tema.

- Comentario-Usuario: Un usuario puede, o no, publicar diversos comentarios en temas, mientras que un comentario pertenece a un único usuario.

- Notificación: es una tabla relación que hace referencia a la posibilidad que tiene un usuario de suscribirse a un tema para recibir notificaciones.

6.3.2 Arquitectura Cliente-Servidor

Para que sea posible el acceso de múltiples usuarios a una información común (temas, comentarios, etc.) es necesario usar una arquitectura cliente/servidor.

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (servidor) que le da respuesta.

Aplicado al presente caso de estudio, nuestra aplicación Android haría la función de cliente haciendo peticiones a nuestro servidor para obtener o modificar datos comunes, que posteriormente pueden ser requeridos por otras aplicaciones clientes.

Cabe destacar también que es una **arquitectura de 3 capas**, quiere decir que es un conjunto de subsistemas cada uno de los cuales depende del que se encuentra en la capa inferior y proporciona los cimientos del que se encuentra inmediatamente por encima de él.

La arquitectura de tres capas es de las más habituales en sistemas informáticos y podemos distinguir los siguientes niveles:

- Presentación: Hace referencia a la parte visual que se utilizará para mostrar la información y los datos. Normalmente, es la parte con la que el usuario final interaccionará.

- Negocio o Lógica: Es la capa intermedia encargada de interactuar con la interfaz y los datos. Su función es recopilar los datos y procesarlos para que se muestren o almacenen según se desee.

- Persistencia: Es la capa de datos. Se encarga de las tareas que realizamos habitualmente con los datos: insertar, modificar, consultar o borrar. Por norma general, la información persiste en esta capa.

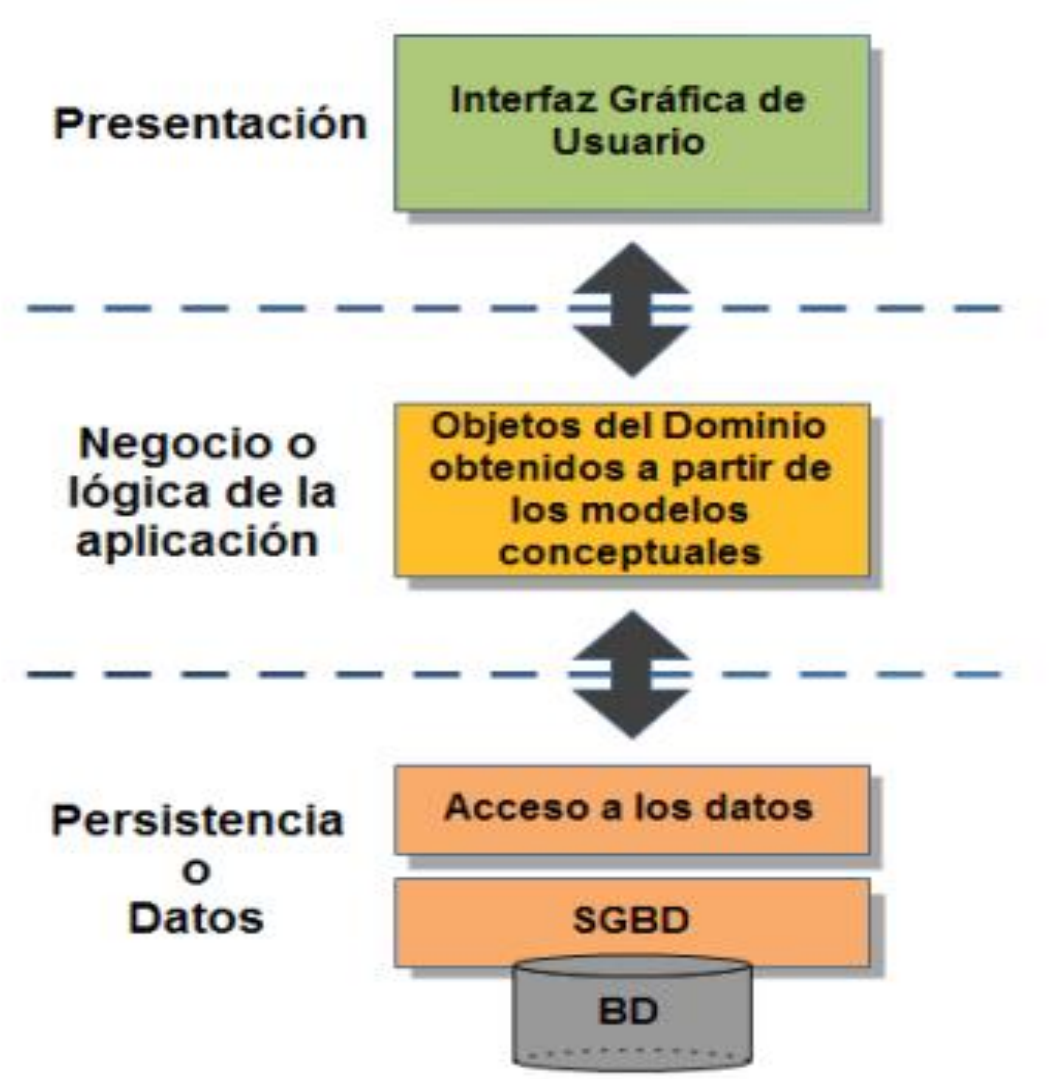

Figura 33. Arquitectura de 3 capas

La adaptación de esta arquitectura a nuestro caso de estudio tendrá una estructura similar a la Figura 34.

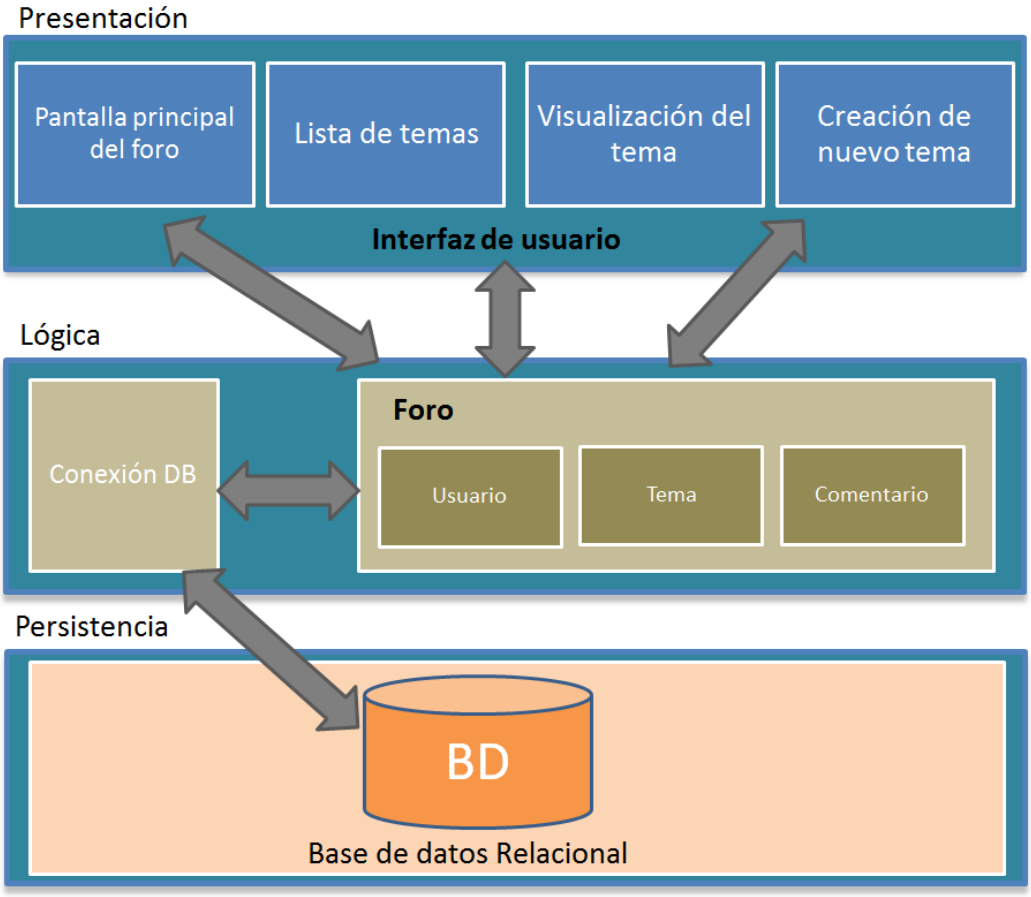

Figura 34. Arquitectura de la aplicación UPV Welcome Incoming

- En la capa de **presentación,** encontramos entre las más destacadas, las pantallas de lista de lista de temas o visualización del tema que requieren de datos para poder mostrarlos. Por otro lado, tenemos el formulario de creación de un nuevo tema que conduce el flujo de información en sentido inverso, es decir, agregando datos a la fuente de datos.

- En la capa de **Lógica** destacamos las clases de Usuario, Tema y Comentario, que hacen referencia a los objetos del modelo de dominio. Y la clase Conexión DB que es la encargada de conectar con la capa de datos.

- En el nivel más bajo encontramos la capa de **Persistencia,** en esta capa encontramos el conector a la base de datos mediante el cual ejecutamos las sentencias SQL sobre la base de datos.

6.3.2.1Elección del servidor

Se valoraron distintas alternativas y tecnologías para implementar la capa servidor del sistema. Finalmente se optó por un servidor de aplicaciones web java (Tomcat 7.0) por las siguientes razones:

- **Compatibilidad:** Es compatible con la mayoría de sistemas operativos actuales. Lo que facilitaría una posible migración del servidor a otra máquina.

- **Lenguaje Java:** Que ejecute código Java es una clara ventaja frente a otras elecciones ya que la aplicación Android también se desarrolla en Java. Por este motivo podemos reutilizar clases comunes (Tema, comentario…) que también son usadas en la aplicación cliente. De este modo, podemos enviar datos de forma rápida en formato JSON sin tener que preocuparnos en exceso por la conversión de datos.

- **GCM:** El servicio de notificaciones push de Google es otro aliciente más, ya que nos facilita el uso acoplando la demo de servidor Java que Google nos proporciona. (Este concepto se explica con más detalle en el apartado 5.5).

6.3.3 Base de datos

6.3.3.1 Tecnología

Para la implementación de la Base de datos se ha optado por una Base de datos relacional MySQL 5.5.24.

Se valoraron otros motores de bases de datos relacionales como SQL Server 2014 y SQLite 3. Finalmente, se optó por MySQL debido a diversas razones:

- Compatibilidad: Es una ventaja clara sobre SQL Server, y es que MySQL está disponible en la mayoría de sistemas operativos (Windows, Linux, Mac OS…).

- Migración y configuración: Realizar la migración de una base de datos MySQL a otra máquina resulta bastante sencillo. Además, se dispone de diversas herramientas de apoyo y configuración (MySQL Workbench, phpMyAdmin…) que nos permiten administrarla de forma rápida y cómoda tanto en local como de forma remota.

- Rendimiento: Alta velocidad de acceso y conexión (mayor que sqlite3).

- Seguridad.
- Open Source.

6.3.3.2 Esquema relacional

Siguiendo el modelo de dominio definido en el apartado 5.3.1 se ha definido el esquema de la base de datos de la siguiente forma:

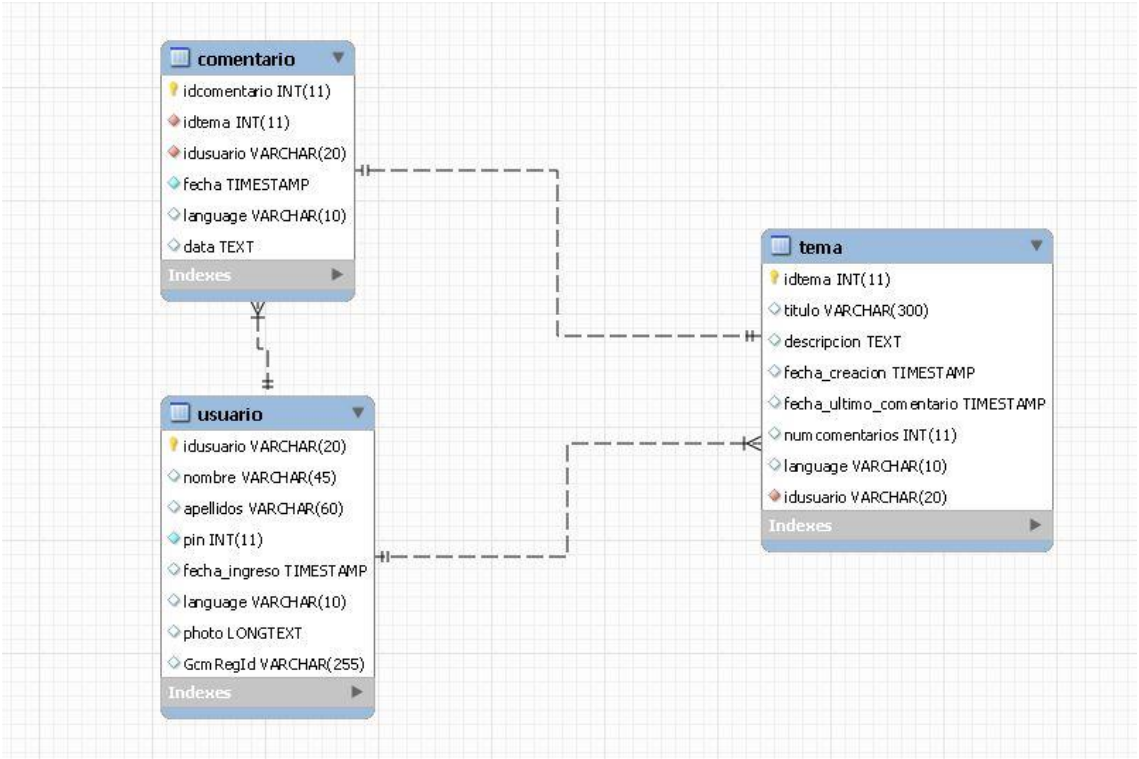

Figura 35. Esquema relacional de base de datos del servidor

Con tres tablas relacionadas entre sí y definidas de la siguiente forma:

- Usuario: Almacena los usuarios.
  - Clave Primaria: *idusuario*, es de tipo varchar y representa el dni del usuario. Es única y no permite valores null.
  - Clave ajena: No contiene ninguna clave ajena.
  - Otros campos: Podemos destacar el campo *photo* de tipo LONGTEXT y que almacena las fotos de los usuarios codificadas en base64.

- Tema: Almacena todos los temas.
  - Clave Primaria: *idtema*, de tipo entero e incremental.
  - Clave ajena: *idusuario*, referencia a la tabla usuario y nos indica que usuario ha creado ese tema.
  - Otro campos: Destacamos el campo *fecha_último_comentario*, este campo se actualiza cada vez que se publica un comentario en ese tema. Esto se hace para ordenar los temas por dicho campo y que no sea necesario calcularlo en cada consulta ganando así en velocidad. Por

motivos similares se ha creado el campo *numcomentarios* (número de comentarios del tema), que se incrementa cada vez que introducimos un comentario.

- Comentario: Almacena todos los comentarios de todos los temas.
  - Clave primaria: *idcomentario*, de tipo entero e incremental.
  - Clave ajena: *idtema*, referencia la tabla tema y nos indica a que tema pertenece ese comentario. *idusuario*, referencia a la tabla usuario y nos indica que usuario ha publicado el comentario.

Como puede observarse, en comparación con el modelo de dominio definido en el apartado 6.3.1, puede notarse la ausencia de una tabla notificaciones que relacione a los usuarios con los temas de los cuales quiere que se le notifique. Esto es debido a que finalmente se optó por implementar esta tabla localmente para cada usuario y de esta forma agilizar la comunicación en la consulta de temas al servidor. El almacenamiento de notificaciones en el servidor es un tema que se tratará en el apartado 5.3.6.

6.3.4 Interfaz de usuario

6.3.4.1 Pantalla principal

Pulsando en la opción Foro del Navigation Drawer accedemos a la pantalla principal del foro en el caso que estemos logueados, si no lo estamos visualizaremos la pantalla de inicio de sesión y, si introducimos los datos correctos, nos redirigirá a la pantalla deseada.

Figura 36. Pantalla principal del foro

Esta pantalla, que es un fragment, contiene las funcionalidades principales sobre los temas y que son las siguientes:

- **Ver todos los temas**: Esta funcionalidad nos lleva a una nueva pantalla en la cual visualizaremos una lista con todos los temas que existen ordenados por fecha de último comentario, de más reciente a más antiguo.

- **Mis Temas**: Al igual que en la funcionalidad anterior, esta opción nos abre una nueva ventana con todos los temas que ha creado el usuario.

- **Mis temas comentados:** Al igual que las dos opciones anteriores, aquí mostramos todos los temas en los que ha comentado el usuario.

- **Buscar:** Esta opción es algo diferente a las anteriores y nos permite buscar una cadena de texto en el titulo de los temas, devolviendo una lista similar a las anteriores con los temas que cumplen la condición.

- **Idioma:** Vuelve a mostrarnos una nueva pantalla, pero en este caso nos aparece un Spinner en la parte superior que nos permite seleccionar un idioma por el cual vamos a filtrar los temas. Por lo tanto, veremos una lista con todos los temas del idioma seleccionado ordenados por fecha del último comentario.

- **Nuevo tema:** Esta funcionalidad nos permite crear nuevos temas.

6.3.4.2 Pantallas con listas de temas

Se incluye en este apartado a todas las pantallas (ActionBarActivity) que muestran una lista de temas y corresponden a las 5 primeras funcionalidades explicadas en el punto anterior. Su interfaz es muy similar en todas a excepción de Buscar (tiene una barra de búsqueda en la parte superior) e Idioma (que posee un Spinner de selección de idioma también en la parte superior).

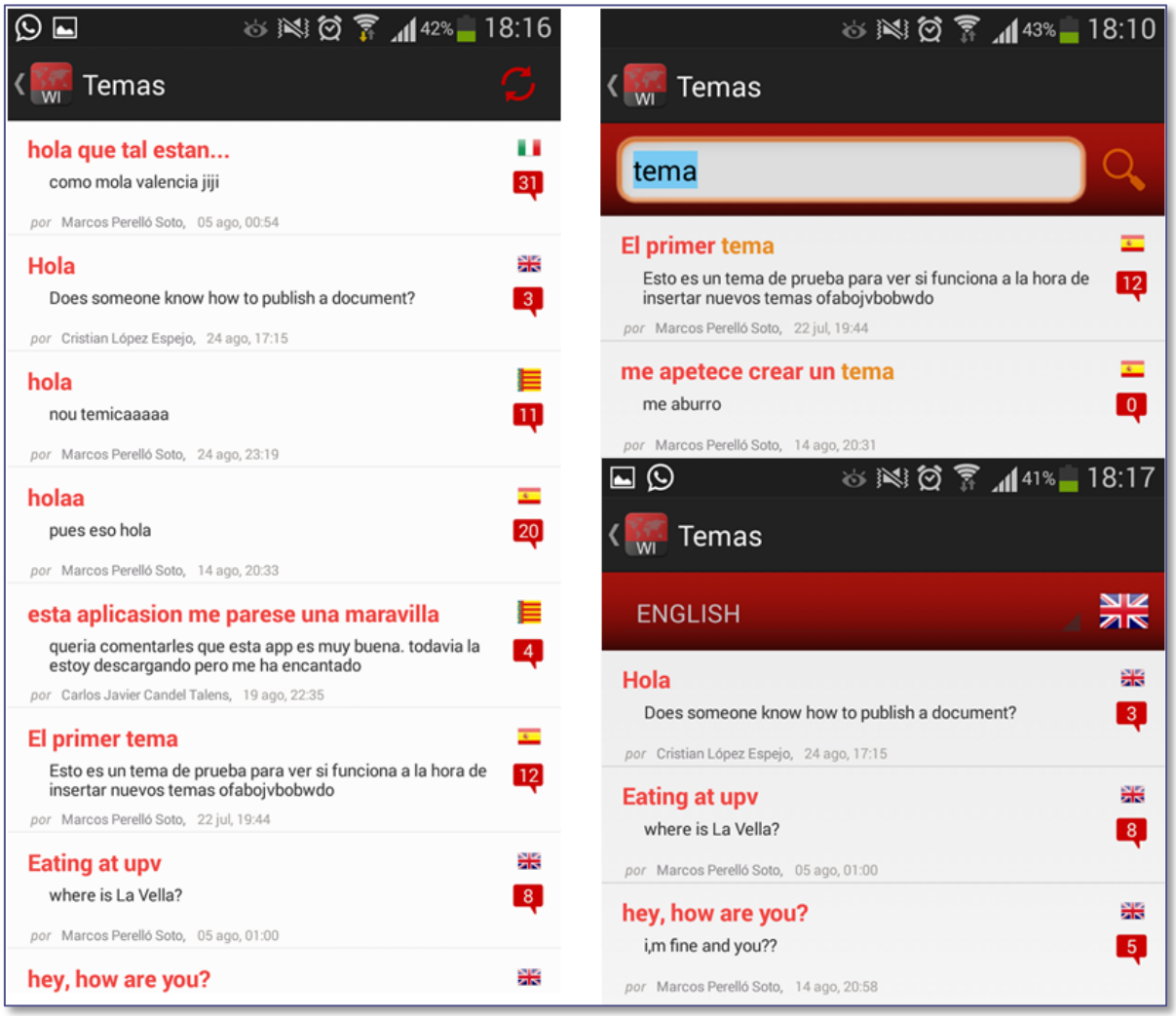

Figura 37. Pantallas con listas de temas de la aplicación.

6.3.4.3 Pantalla nuevo tema

Desde esta pantalla es donde introduciremos los datos del tema**:** título, descripción e idioma. Y una vez introducidos, y si son correctos, podremos pulsar el botón de publicar el tema. Si todo ha ido correctamente veremos un mensaje indicándolo o, en caso contrario, indicando que no se ha podido publicar el tema (fallo en la conexión, parámetros incorrectos, mensaje vacío, etc.) y que lo intentemos de nuevo más tarde. En el caso de que se haya publicado con éxito visualizaremos un nuevo botón “Ver tema” que nos llevara a la visualización del tema.

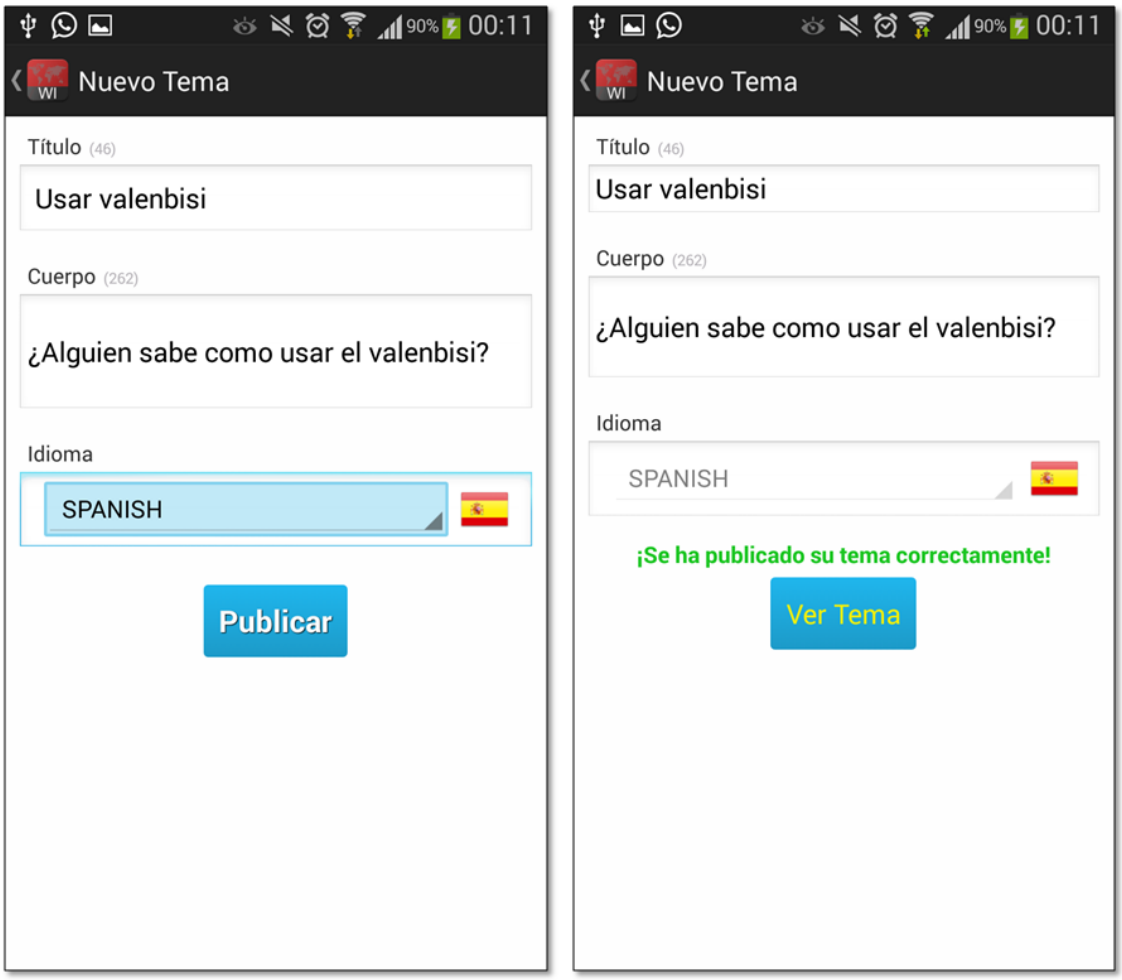

Figura 38. Pantalla de nuevo tema.

6.3.4.4 Visualización del tema

Podríamos decir que esta pantalla es la más relevante del foro. En ella es donde visualizaremos toda la información acerca de un tema: descripción, comentarios, fotos de los usuarios, etc. Esta pantalla contiene diversas funcionalidades:

- **Cargar comentarios**: al abrir la pantalla se descargarán y se mostrarán automáticamente todos los comentarios del tema en cuestión.

- **Cargar fotos**: cada foto de usuario que podemos ver en los comentarios se descargará en cuanto se procese el comentario correspondiente.

- **Activar/desactivar notificaciones**: en la barra superior de la pantalla encontramos un icono de una campana, pulsando en él podremos activar o desactivar las notificaciones para ese tema. Este apartado se explica con detalle en el punto 5.3.6.

- **Traducir**: también es posible traducir los comentarios y la descripción del tema en el idioma deseado. Deberemos pulsar sobre la banderita de idioma y elegir el idioma en el diálogo emergente. Visualizaremos un mensaje en la esquina inferior izquierda indicando que el comentario ha sido traducido, pulsando sobre este mensaje volveremos a ver el comentario original.

- **Escribir comentario**: si bajamos el scrollView hasta bajo del todo (podemos hacerlo automáticamente pulsando en la imagen de la flecha blanca, Figura 39) veremos la caja de escribir comentario. Pulsando sobre ella se desplegará y nos mostrará una caja de texto donde introduciremos el mensaje, seleccionaremos el idioma en el que lo queremos publicar y pulsaremos a enviar. Recibiremos un mensaje de respuesta (icono verde o rojo) indicándonos si se ha publicado correctamente el comentario.

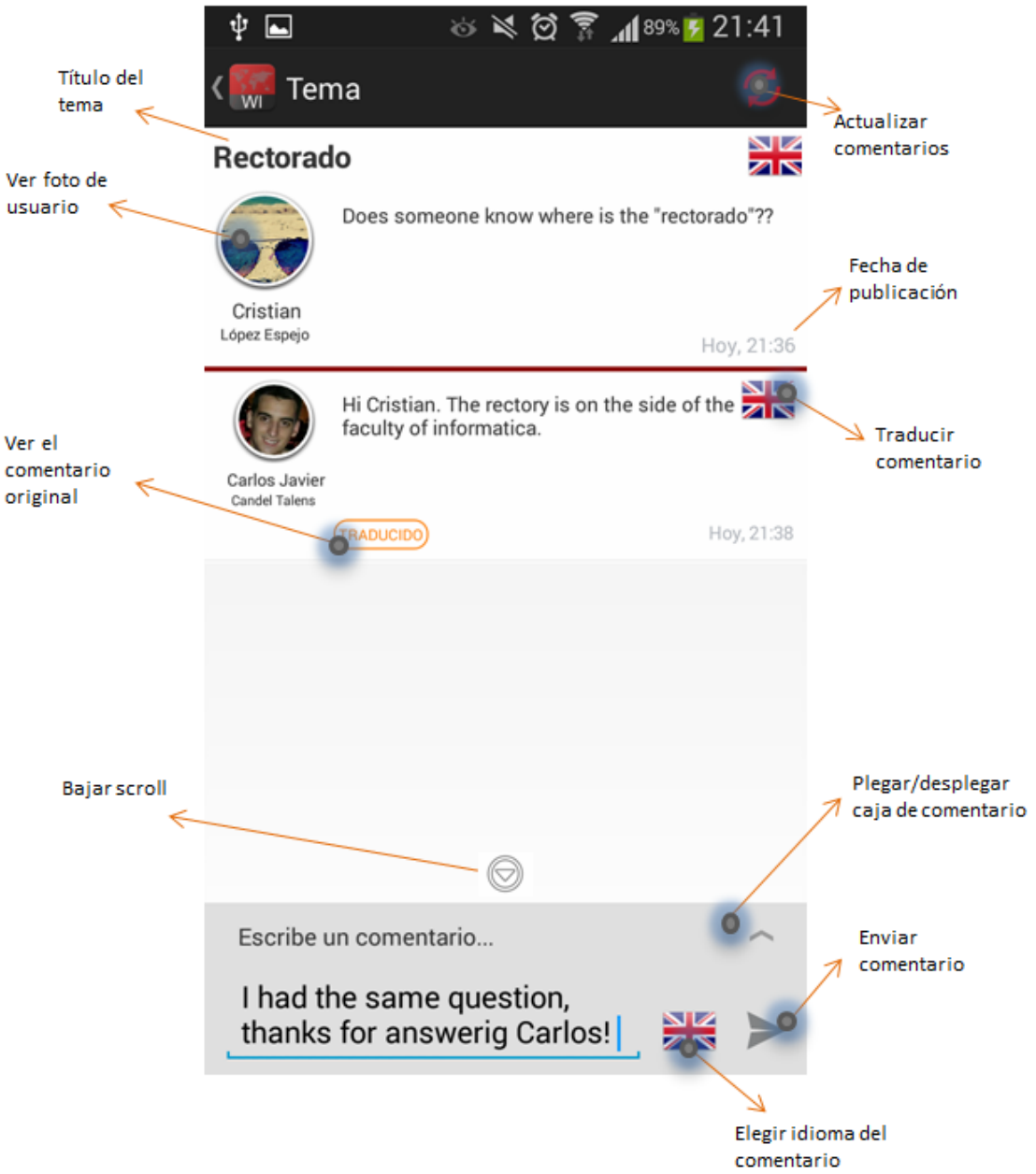

Figura 39. Pantalla visualización del tema. Acciones y elementos

6.3.4.5 Visualización de la foto de usuario

Es una pantalla simple a la que se accede pulsando sobre la foto de un usuario y en la que podremos visualizar en mayor tamaño la foto en cuestión (Figura 29). Podremos compartir esta imagen pulsando sobre el icono de compartir situado en la parte derecha de la barra de acción.

Figura 40. Pantalla de visualización de la foto de usuario

6.3.5 Implementación del servidor

Como hemos comentado en el apartado 4, para el desarrollo de la lógica de negocio del servidor se ha utilizado el IDE Netbeans y lenguaje java.

La interacción con el servidor se realiza mediante **servlets**. Un servlet es un pequeño programa Java que se ejecuta dentro de un servidor Web. Los servlets se encargan de recibir y atender las peticiones de los clientes Web, por lo general a través de HTTP[11].

Se han implementado múltiples servlets para satisfacer las distintas peticiones y consultas que requiere la aplicación Android. Son los siguientes:

- **getComentarios**: Dado un identificador de tema, devuelve todos los comentarios de dicho tema en formato JSON.

- **getEstadisticas**: Devuelve las estadísticas del foro del usuario en cuestión, numero de comentarios totales y numero de temas creados, en formato JSON.

- **getTemas**: Devuelve los temas según el parámetro "tipo" (todos los temas, temas creados por el usuario, temas en un idioma determinado, temas que contienen una cadena de caracteres concreta, etc.)

- **getUserPhoto**: Devuelve la foto de un usuario en base 64.

- **insertComentario**: Inserta en la base de datos un comentario con los parámetros necesarios pasados en la petición, devuelve "-1" si no se ha podido realizar la operación. Una vez se ha devuelto la respuesta, en el caso de que haya sido satisfactoria, se notifica de ello mediante GCM a los usuarios que corresponde.

- **insertTema**: Inserta un tema en base de datos. Devuelve "-1" en caso de que no sea posible la operación.

- **insertarUsuario**: Inserta un usuario en la base de datos. Si ya existe actualiza sus campos.

- **setPhotoUser**: Actualiza la foto del usuario.

- **unRegisterUsuario**: Este método elimina el ID de registro de GCM de un usuario de la base de datos. En ningún caso elimina al usuario de la base de datos.

El funcionamiento básico de la mayoría de servlets consta de los siguientes pasos:

- Obtención de los parámetros de la petición.
- Comprobación de credenciales del usuario, DNI y PIN.
- Interacción con la base de datos. Ya sea consulta, actualización o eliminación de alguno o varios campos.
- En caso de ser una consulta, procesamiento de los datos recogidos y creación de objetos de lógica.
- Respuesta. Se devuelve en formato JSON los datos de la consulta o un código de respuesta si se trata de una transacción (actualización, eliminación, etc.) que suele ser el número de elementos afectados (filas de la tabla en cuestión) o "-1" en caso de error.

6.3.6 Notificaciones

Debido a la necesidad de alertar a los usuarios sobre comentarios en el foro, se ha integrado el servicio de notificaciones de GCM. Ante dicha integración, se veía necesario gestionar estas alertas de forma que fuesen editables por el usuario, ya que en algunos casos podían llegar a ser molestas.

Pese a que existe la opción de activar o desactivar todas las alertas del foro, se consideró necesario un sistema de activación y desactivación de alertas selectivo. Es decir, que nos permita desactivar notificaciones para los temas indicados.

En un principio se optó por realizar esta agrupación (usuario-tema) en la base de datos del servidor, pero se desestimó debido al gran volumen de filas que podía contener la tabla y a que disminuiría en gran medida la velocidad de consulta de los temas. Ya que tenía que relacionar los temas con las notificaciones. Por ello, y puesto que es uno de los objetivos del proyecto, se prefirió la velocidad de la aplicación.

La alternativa que se planteó fue almacenar estas relaciones de forma local en el dispositivo. Para ello se hizo uso de ficheros de base de datos SQLite3. Si se almacena una notificación en la base de datos se entiende que esta se encuentra desactivada.

Por lo tanto, y puesto que una notificación solo llegara al usuario si es un tema creado por el o si ha comentado en el tema en cuestión, se puede gestionar de la siguiente forma.

En primer lugar, el servidor manda la notificación push a todos los dispositivos que deben recibirla por los motivos anteriormente expuestos. Una vez recibida el sistema consultara en la base de datos y si la notificación no se encuentra, y todas las notificaciones de la aplicación están activas (valor guardado en Preferencias de la aplicación), lo notificará al usuario.

Podemos activar o desactivar las notificaciones del tema desde la pantalla de visualización del tema, en la barra de acciones superior encontraremos un icono de una campana y pulsando sobre la notificación quedará activada o desactivada. Por defecto las notificaciones estarán activas siempre y deberemos desactivarlas si lo deseamos. Si no podemos recibir notificaciones de un tema (no hemos comentado en el) no se mostrara el icono correspondiente.

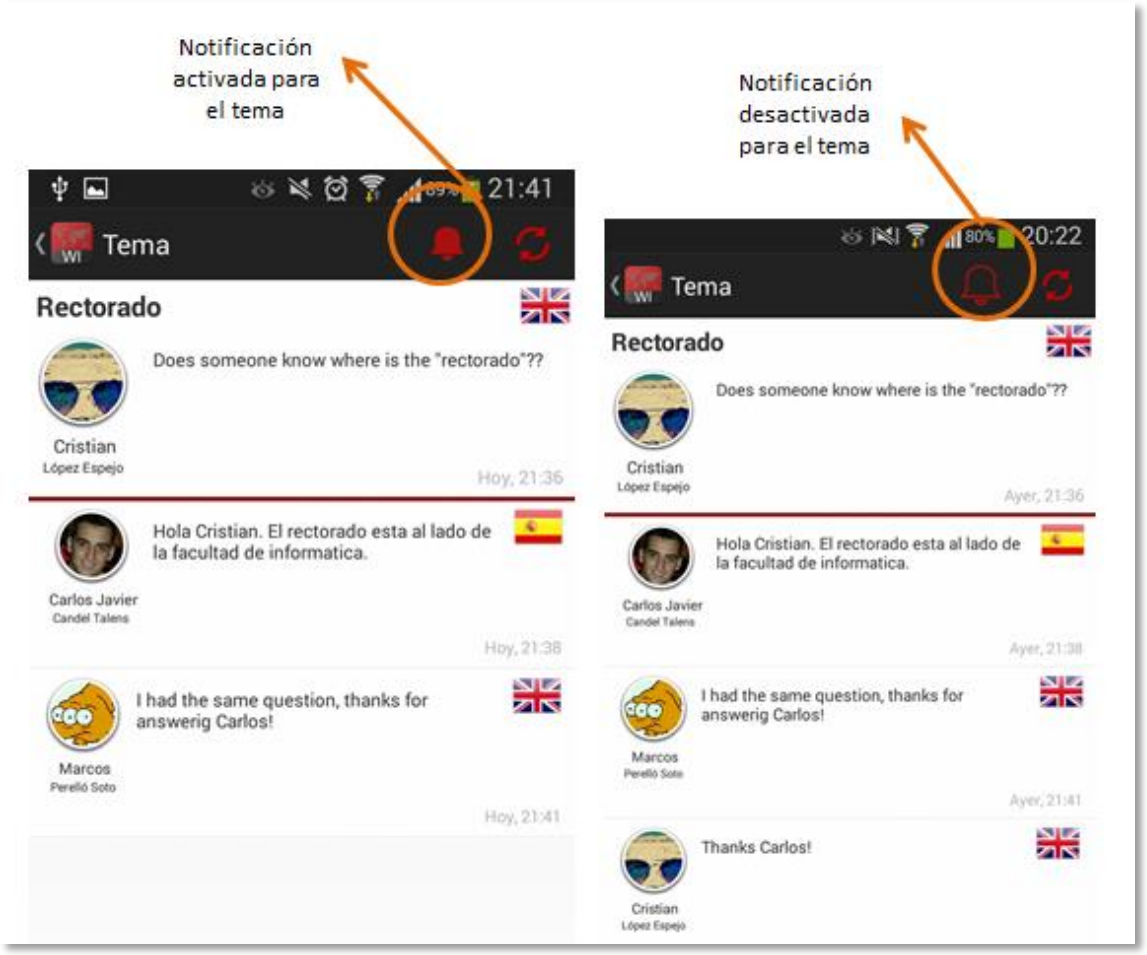

Figura 41. Ejemplo de notificación activada/desactivada

6.4 Opciones

El último elemento del menú lateral de la aplicación corresponde al apartado opciones.

Desde este panel, el usuario podrá configurar a su elección diversos aspectos de la aplicación.

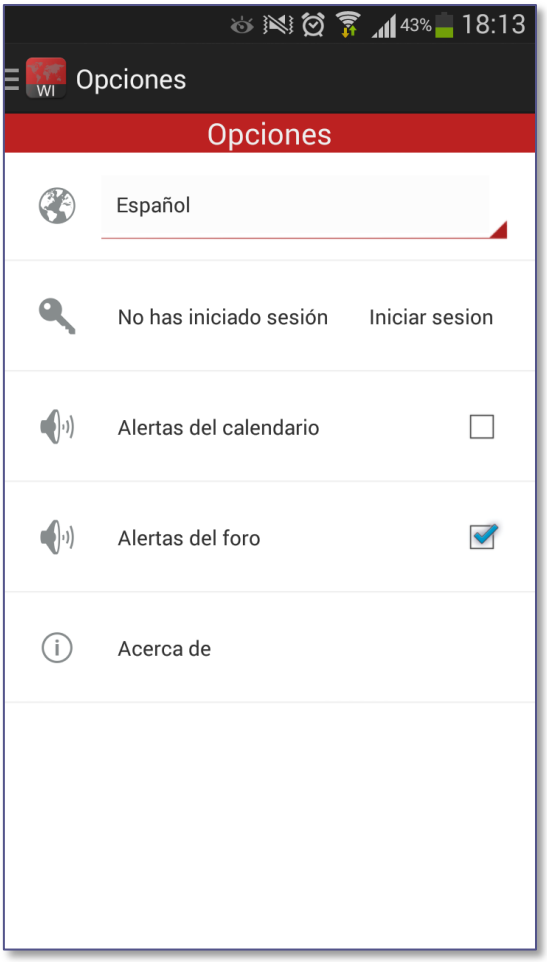

Figura 42. Pantalla de opciones

6.4.1 Preferencias

Para gestionar mejor las preferencias comunes se han dividido en dos archivos distintos:

- **Preferencias del usuario**: almacenan los datos referentes al usuario como puede ser el DNI, PIN, nombre apellidos, etc.

- **Preferencias de la aplicación**: Almacenan datos que tienen mayor relación con la aplicación en sí, como por ejemplo si están activadas las alertas del foro o calendario.

Esta distinción facilita el inicio y cierre de sesión, ya que solo debemos limpiar las preferencias de usuario cuando cerramos sesión.

Para gestionar datos de configuración, Android usa un mecanismo ligero de almacenamiento de pares clave-valor, los archivos de preferencias compartidas. Fundamentalmente se usan para compartir información entre actividades de una aplicación.

6.4.2 Cambio de idioma

Este elemento, el primero del menú de opciones, consta fundamentalmente de un Spinner que nos permite seleccionar el idioma completo de la aplicación. Al seleccionar un idioma nos aparecerá un diálogo de confirmación. En el caso de que pulsemos la opción de aceptar el idioma de la aplicación cambiará instantáneamente y nos redirigirá a la pantalla inicial de la aplicación (apartado de noticias).

Para hacer posible el multi-idioma en la aplicación se han ido añadiendo todas las cadenas de texto visibles por el usuario al fichero strings.xml de la aplicación ubicado en el directorio de recursos de nuestro proyecto Android (*/res/values/strings.xml)*. De tal forma que, para añadir nuevos idiomas a la aplicación solo es necesario traducir las cadenas de texto de este fichero al idioma deseado y almacenarlo en una carpeta de recursos siguiendo el estándar impuesto por Android. Por ejemplo, si se quiere traducir al inglés debería almacenarse el fichero strings.xml correspondiente en la carpeta (*/res/values-en/strings.xml*) .

Por el momento la aplicación sólo soporta inglés y español pero espera poder traducirse a distintos idiomas en el futuro.

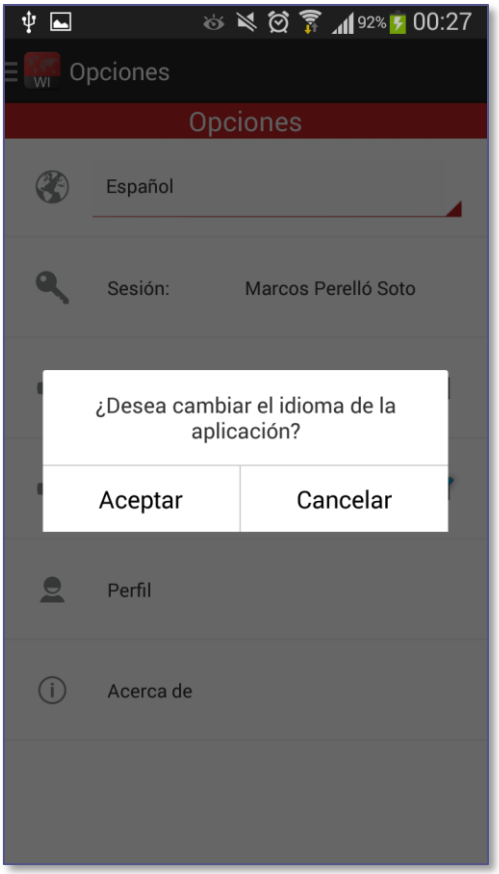

Figura 43. Pantalla de cambio de idioma de la aplicación

6.4.3 Inicio y cierre de sesión

Este ítem nos permite iniciar o cerrar nuestra sesión de usuario.

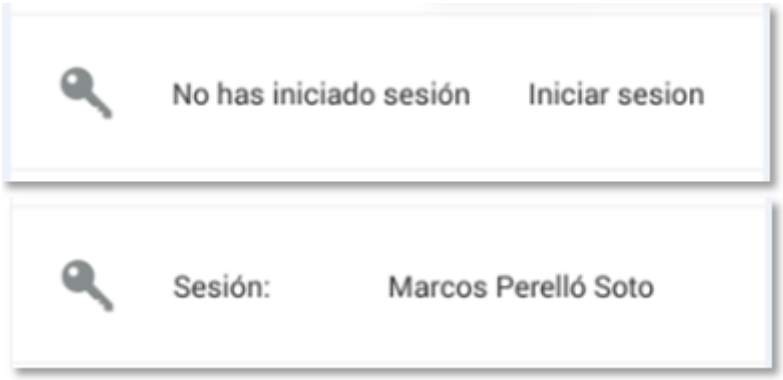

Figura 44. Ítem de inicio de sesión en la pantalla opciones

6.4.3.1 Autenticación

Puesto que la aplicación va dirigida a alumnos de intercambio de la Universidad Politécnica de Valencia, sólo se podrá acceder a determinados servicios como Foro o Calendario si se ha registrado previamente como tal introduciendo sus credenciales de usuario.

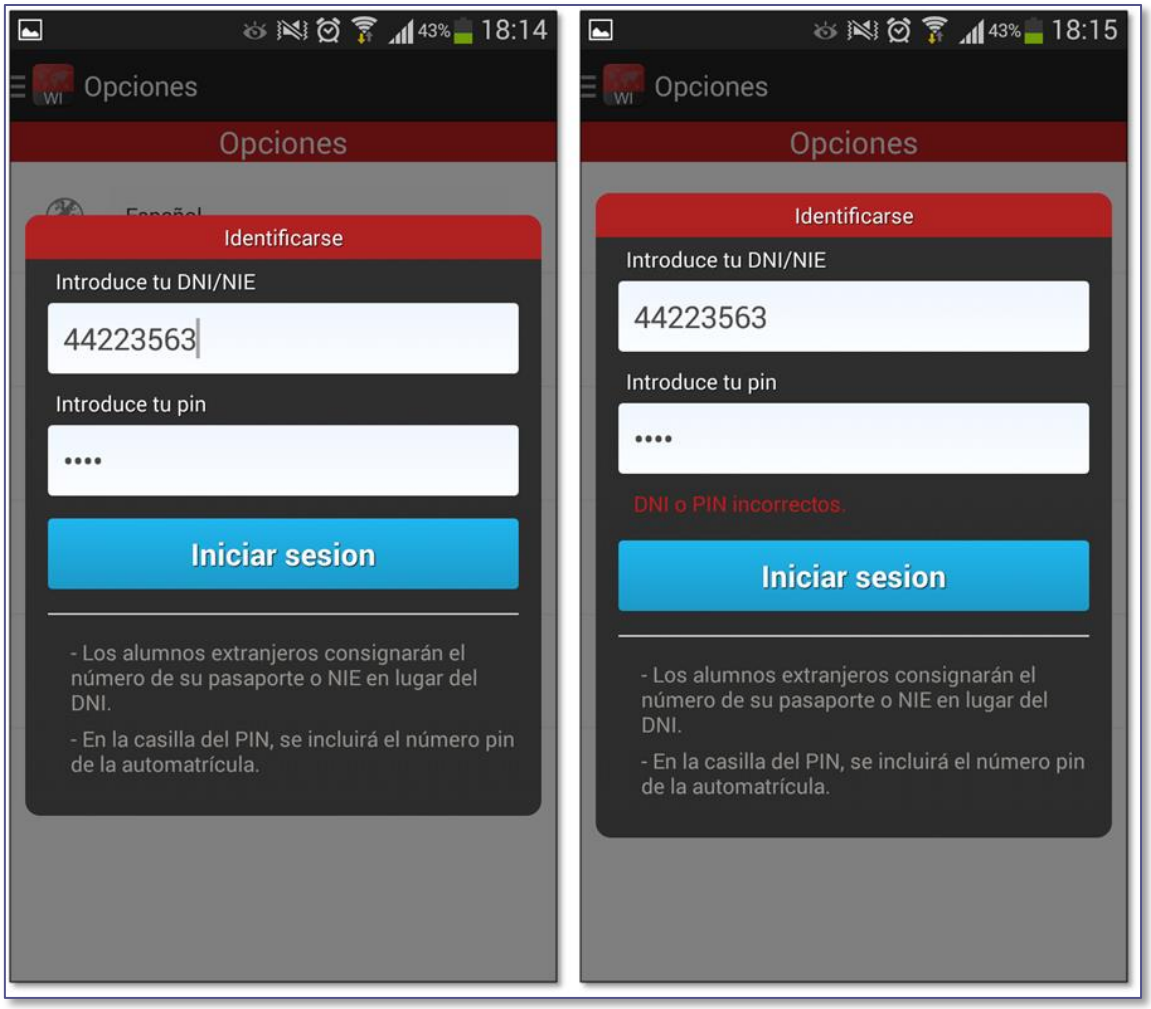

Figura 45. Pantalla de inicio de sesión

Si introducimos nuestro usuario y pin y pulsamos el botón de iniciar sesión, el sistema comprobará que los datos son correctos, en ese caso enviará una petición al servidor de la aplicación para que nos registre como nuevos usuarios si es la primera vez que nos registramos, o si ya nos hemos registrado en la aplicación con anterioridad, nos actualizará en la base de datos del sistema.

6.4.3.2 Seguridad

Seguridad es uno de los objetivos que pretende el proyecto. Puesto que trata con datos personales para el usuario como son el DNI y PIN de acceso a la intranet de la UPV, se ha tratado de poner en riesgo lo mínimo posible.

Cada vez que se hace una petición al servidor de la aplicación se pasa como parámetros el DNI y PIN del usuario para prevenir que cualquier persona que conozca los parámetros de la petición pueda suplantar la identidad de otro usuario. Esta idea nos parecía segura en un principio, pero generaba ciertas dudas el hecho de tener pasar el DNI y PIN del usuario en todo momento ya que la conexión usada para comunicar al servidor de la aplicación era una conexión http no segura.

Por todo esto se han tomado las siguientes medidas de seguridad:

- La comprobación inicial de credenciales del usuario: DNI y PIN a la intranet se realiza mediante el protocolo de transferencia segura (HTTPS).

- Una vez se ha obtenido respuesta del servidor de la intranet y sabemos que los datos son correctos se genera una cadena de texto aleatoria en sustitución al PIN real. De esta forma el PIN no tiene que viajar por la red en todo momento comprometiendo así la seguridad del usuario. Por lo tanto, usaremos esta cadena aleatoria que se ha generado a modo de identificador de sesión, generando otra completamente distinta en el caso de que cerremos sesión y volvamos a registrarnos.

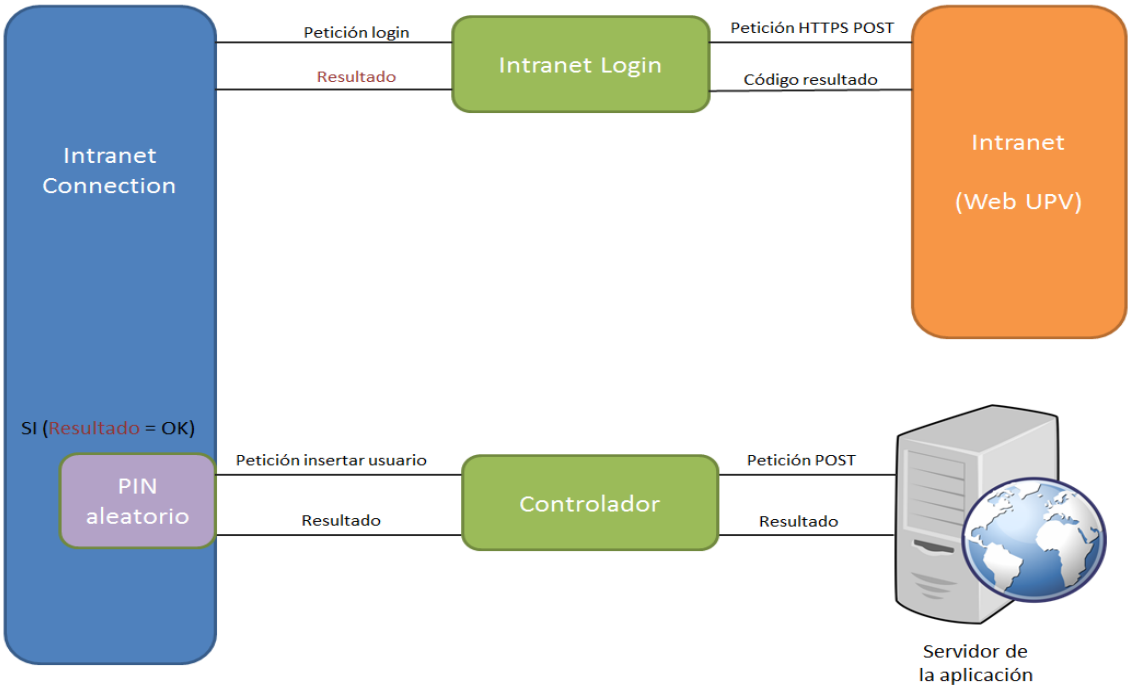

Figura 46. Diagrama de inicio de sesión

6.4.3.2 Sesiones no concurrentes

Puesto que la aplicación solo está desarrollada en la plataforma Android y se entiende que un usuario no suele disponer de dos dispositivos móviles Android en uso a la vez, se ha diseñado la aplicación para que no existan dos sesiones simultáneas de un mismo usuario.

Desarrollar la aplicación de esta forma facilita algunos aspectos del diseño:

- El uso de un PIN aleatorio, ya que un usuario genera un pin aleatorio cada vez que inicia sesión, podría darse el caso que un usuario identificado correctamente en la intranet de la UPV, no pudiese identificarse como tal en el servidor de la aplicación debido a que el PIN generado aleatoriamente por otra sesión posterior sería el vigente, de acuerdo al diseño de la base de datos.

- El ID de registro de GCM identifica al dispositivo en el que se ha iniciado la sesión. A la hora de enviar notificaciones a un usuario se consulta este campo. El hecho de que solo exista una sesión hace posible que no se entre en conflicto y la notificación llegue al usuario correctamente.

En caso de querer implantar sesiones concurrentes se debería modificar la base de datos añadiendo una tabla de sesiones que contemplase las distintas sesiones que pueda tener un usuario y asociadas a distintos ID de registro de GCM.

Se ha conseguido tener una única sesión por usuario de la siguiente forma:

Cuando nos logueamos, y una vez hemos comprobado que las credenciales son correctas en la intranet, si es la primera vez nos registramos en la base de datos del servidor sin ningún problema. Sin embargo, si ya existía una sesión previa activa en otro dispositivo, enviará una notificación de cierre de sesión a dicho dispositivo, el cual realizará automáticamente la acción de cerrar sesión y lo notificará al usuario (Figura 35).

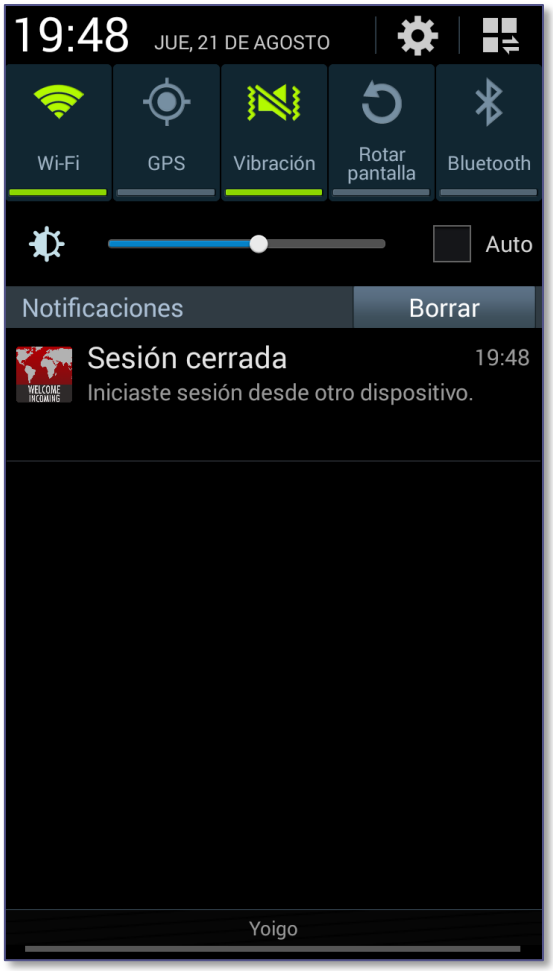

Figura 47. Notificación de cierre de sesión en el dispositivo

6.4.4 Activar notificaciones

Existe la posibilidad de que el usuario prefiera no recibir notificaciones del foro o calendario. Por ello, se da la opción de activar o desactivar las notificaciones tanto del foro como del calendario, pulsando sobre la casilla correspondiente y guardando la opción elegida en las preferencias de la aplicación.

```java
public static boolean getForumAlerts(Context context) {
    try {
        return context.getSharedPreferences(APP_FILE, Activity.MODE_PRIVATE).getBoolean(APP_FORUM_ALERTS, true);
    } catch (Exception e) {
        Log.d(Preferencias.class.getSimpleName(), "Exception", e);
    }
    return false;
}

public static void setForumAlerts(Context context, boolean id) {
    try {
        SharedPreferences.Editor editor = context.getSharedPreferences(APP_FILE, Activity.MODE_PRIVATE).edit();
        editor.putBoolean(APP_FORUM_ALERTS, id);
        editor.commit();
    } catch (Exception e) {
        Log.d(Preferencias.class.getSimpleName(), "Exception", e);
    }
}
```

Figura 48. Ejemplo de uso de Preferencias

6.4.5 Perfil

Al pulsar sobre esta opción se nos abrirá una nueva pantalla que muestra algunas de las preferencias del usuario: foto, nombre, apellidos y estadísticas.

Figura 49, Pantalla de Perfil sin foto de usuario.

Esta opción solo será visible en el caso que nos encontremos logueados, ya que en caso contrario no tiene sentido mostrarla.

Desde la pantalla del perfil podremos visualizar, como hemos dicho anteriormente, además del nombre y apellidos del usuario, estadísticas del foro (número de temas abiertos y número de comentarios totales).

También podremos ver y editar nuestra foto desde esta pantalla.

6.4.5.1 Foto de usuario

El usuario puede elegir tener una foto de perfil que será visible al visualizar sus comentarios o temas en el foro. Desde esta pantalla se podrá elegir la foto que se desee o eliminar la existente.

Pulsando sobre la foto nos aparecerá un diálogo con tres opciones para la elección de la foto:

- Cámara: mediante un *Intent* lanzamos la actividad de la cámara del teléfono y podremos tomar una foto en el momento.
- Galería: de forma similar a la opción anterior, nos aparecerá un diálogo con las aplicaciones instaladas en el teléfono capaces de proporcionarnos una foto ya existente.
- Eliminar foto: elimina la foto existente, en este caso aparecerá la foto de usuario por defecto.

Figura 50. Diálogo de elección de la foto de usuario

Una vez seleccionada la foto debemos recortarla para que todas las fotos tengan las mismas dimensiones y se vean de forma clara y sin deformarse. Para ello en un principio se comenzó a diseñar una actividad que nos permitiese recortar la imagen de acuerdo a unas dimensiones específicas, pero no se obtuvo el resultado esperado.

Por ello, se buscó una solución alternativa y finalmente se encontró una librería gratuita para Android que nos permitía recortar la imagen a nuestro antojo, cropimage[12].

Para importarla únicamente había que incluirla como un nuevo paquete del proyecto y combinar y añadir algunos recursos a nuestro directorio.

Por lo tanto, una vez se ha elegido y recortado la imagen, esta se escala a 154 x 154 pixeles para que tenga un tamaño cuadrado y ocupe poco espacio a la hora de recogerlas del servidor.

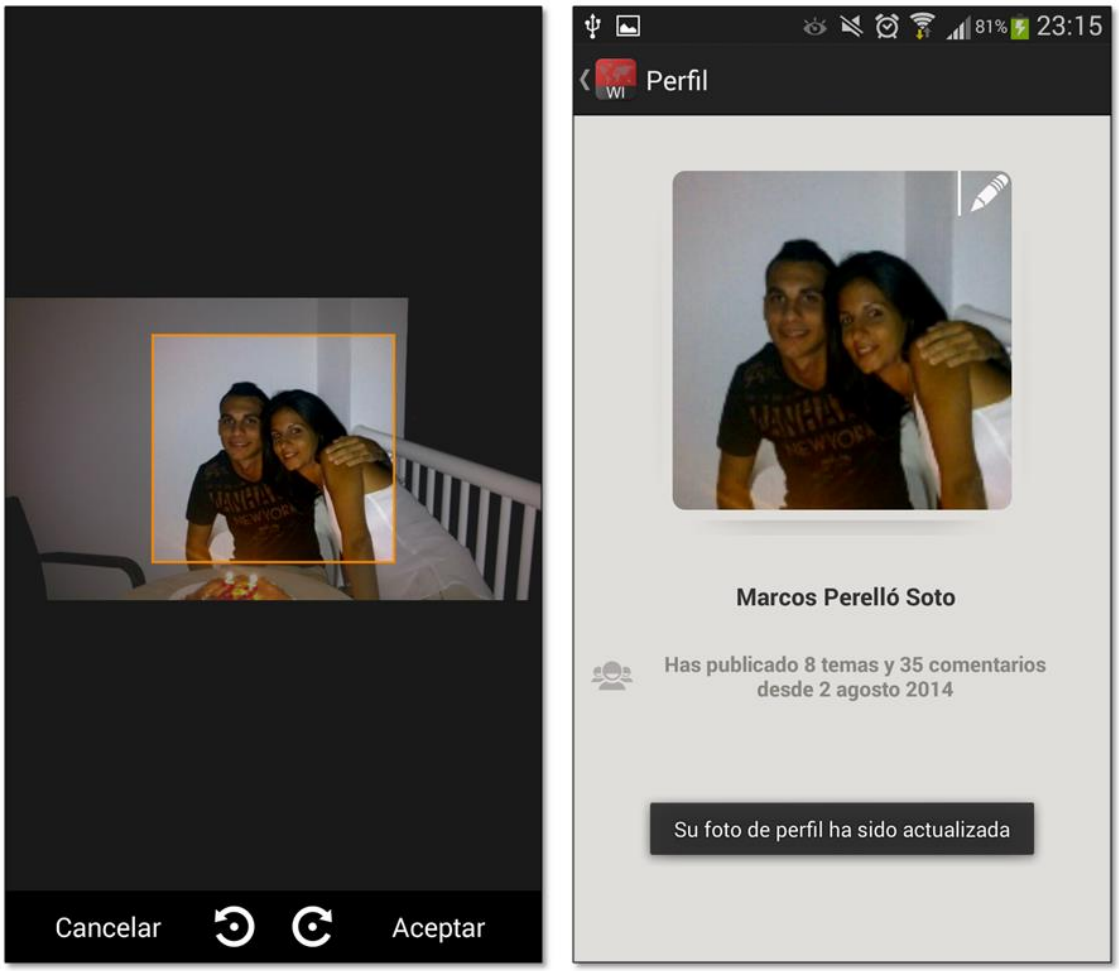

Figura 51. Cambio de la foto de perfil

Como mencionamos también en otros apartados, las fotos se codifican en base64 para su transferencia y posterior almacenamiento. Al pasar la foto a texto, nos permite almacenarla fácilmente en las preferencias del usuario y cargarlas con rapidez sin tener que pedirla al servidor.

6.4.6 Acerca de

Esta opción es puramente informativa y muestra los nombres de los desarrolladores de la aplicación y la versión de la misma.

6.5 Notifications Push y Google Cloud Messaging (GCM)

Con el fin de alertar a los dispositivos de distintos eventos de interés (comentarios en el foro, registros desde otros dispositivos, etc.) se ha decidido hacer uso del servicio de notificaciones push que ofrece Google.

Google Cloud Messaging (GCM) es un servicio gratuito que ayuda a los desarrolladores de aplicaciones Android a enviar datos desde los servidores de sus aplicaciones. El servicio gestiona las colas de mensajes en todos los aspectos y nos permite enviar mensajes de hasta 4 KB a los dispositivos móviles[13].

La aplicación que se ejecuta en el dispositivo móvil no necesita estar ejecutándose cuando recibe la notificación, sino que hará uso de la clase *BroadcastReceiver* para despertarla.

Nos hemos decidido a usar este servicio ya que, como hemos comentado anteriormente, es completamente gratuito, sin cuotas y sin límite de mensajes.

Para hacer uso del servicio, el primer paso fue crear un proyecto en la consola de desarrolladores de Google y habilitar el servicio de GCM (*Ver Anexo A)*

Una vez habilitado podemos obtener una Server key para nuestro servidor desde el apartado de credenciales. Esta Server key la usaremos posteriormente al enviar mensajes desde el servidor. Una vez realizados estos pasos, ya tenemos todo configurado para empezar a usar el servicio.

6.5.1 Componentes

Son las entidades que desempeñan un papel primordial en GCM. Aplicación cliente, servidor de terceros y servidor de GCM.

- **Aplicación cliente**: Es la aplicación que se ha habilitado en la consola de desarrolladores de Google como proyecto (UPV Welcome Incoming). El dispositivo en el que se ejecuta la aplicación debe tener una cuenta de google habilitada y disponer de una versión Android superior a la 2.2.
- **Servidor de terceros**: Es el servidor de la aplicación. Es el que registramos previamente en la consola de desarrolladores y quien envía datos a las aplicaciones mediante el servidor de GCM.
- **Servidor de conexión GCM**: Es el servidor de Google encargado de recibir el mensaje de parte del servidor de terceros, gestionarlo y enviarlo cuando sea posible al dispositivo que contiene la aplicación.

En la figura 52 podemos ver la forma en la que los componentes interactúan.

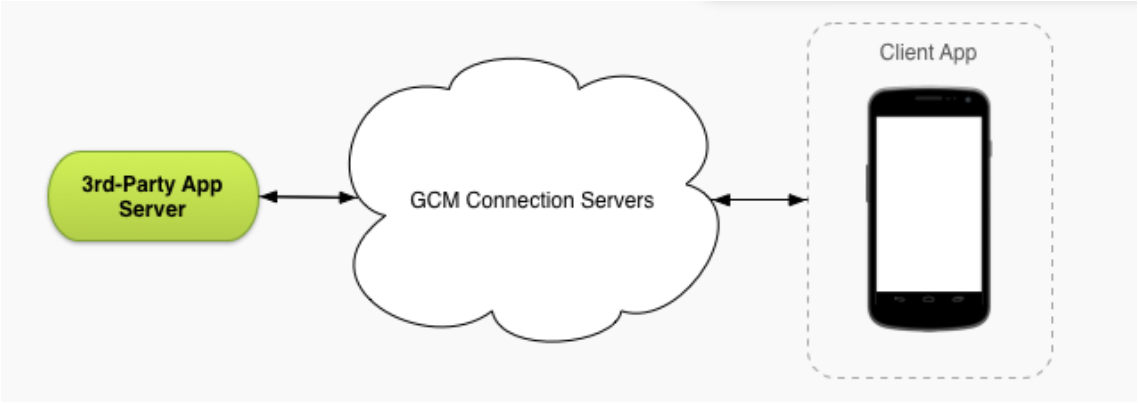

Figura 52. Componentes de GCM

6.5.2 Flujo del ciclo de vida

El flujo de vida de un mensaje requiere de los siguientes pasos:

1. Habilitar GCM: La aplicación Android que se ejecuta en el dispositivo debe registrarse en GCM para recibir mensajes. Para ello, la primera vez que la aplicación necesita usar el servicio de mensajería conecta con el servidor de GCM mediante el método *register* que devuelve un ID de registro que debe almacenar para su uso posterior.

2. Enviar un mensaje:
    1. El servidor de aplicaciones de terceros envía un mensaje al servidor de GCM.
    2. El servidor de GCM lo encola y lo almacena a la espera de que el dispositivo móvil que debe recibirlo entre en conexión, y una vez que se encuentra en línea envía el mensaje.
    3. Recibir un mensaje: el dispositivo móvil que está en conexión recibe el mensaje entrante, identifica la aplicación a la que va dirigida y la despierta para que procese los datos.

La aplicación Android puede anular el registro de GCM si ya no quiere recibir mensajes.

A continuación, se describe el flujo necesario para que el mensaje de bienvenida llegue a nuestra aplicación:

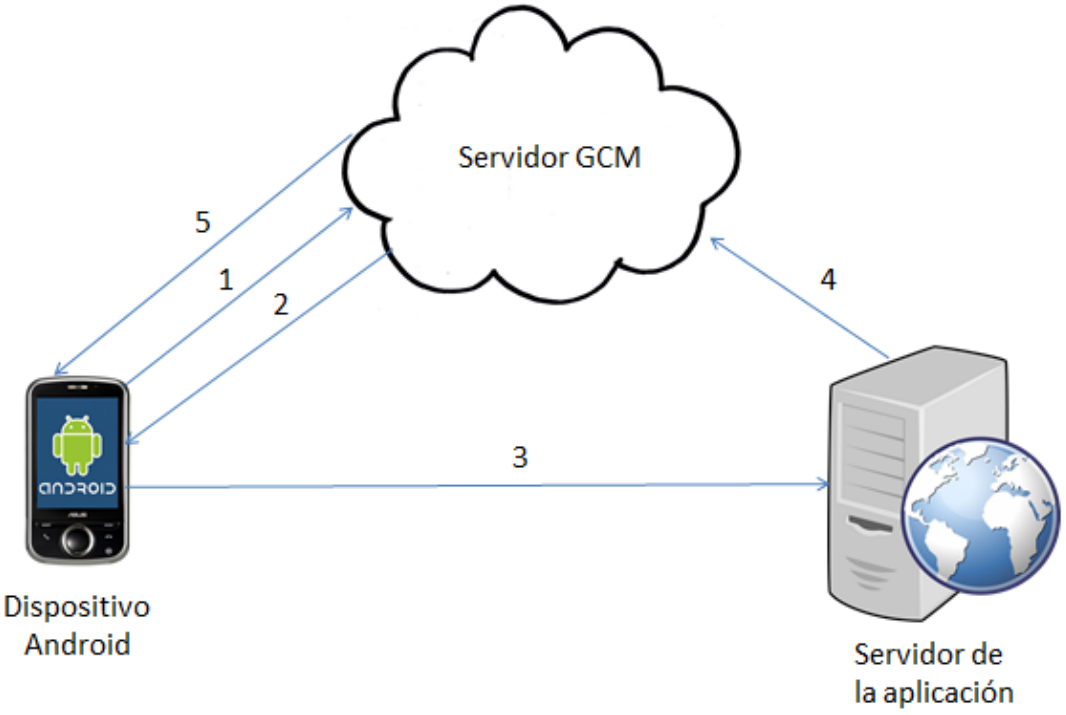

Figura 53. Flujo de la comunicación del mensaje de bienvenida de la aplicación

1. El usuario se loguea con sus credenciales de la UPV y la aplicación solicita el ID de registro al servidor GCM.

2. El servidor de GCM genera un ID de registro y lo envía la aplicación.

3. La aplicación envía los datos del usuario (DNI, pin, nombre, apellidos…) junto con el ID de registro.

4. El servidor almacena los datos en la base de datos y envía al servidor de GCM el mensaje de bienvenida asociado al ID de registro.

5. El servidor de GCM comprueba que el dispositivo se encuentra conectado a Internet y envía el mensaje para que sea procesado por el dispositivo.

6.5.3 Configuración de la aplicación cliente

Para configurar la aplicación se ha seguido la guía que proporciona Google al respecto[14], realizando modificaciones cuando ha sido necesario. Google nos

proporciona unas librerías (o paquetes de código que debemos añadir a nuestro proyecto) para facilitarnos la tarea y un proyecto de ejemplo (demo[15]) en el que nos hemos basado para configurar el servicio y que cuenta con tres clases básicas:

- GcmBroadcastReceiver: Se activa cuando llega el mensaje y lo recibe.
- GcmIntentService: Esta clase es un servicio (extiende de IntentService) y es invocada desde la clase anterior para que procese el mensaje y, mediante la clase NotificationManager, nos envíe una notificación indicándonos que se ha publicado un comentario en el foro, que se inició sesión en otro dispositivo, etc.
- GcmConfig: Es una clase estática que nos proporciona métodos para almacenar el ID de registro, solicitarlo al servidor GCM, comprobar que le dispositivo sea compatible, etc. En definitiva, nos proporciona los métodos básicos que debemos implementar para configurar la aplicación cliente.

Siguiendo las indicaciones de Google, nos recomiendan que al iniciar la aplicación comprobemos que la aplicación tiene un ID de registro almacenado y si no es así lo solicite y lo almacene en las preferencias.

En cambio, solo nos interesa registrar el dispositivo si el usuario se ha logueado previamente en la aplicación. Por lo tanto, inmediatamente después de que el proceso de identificación como alumno de la UPV haya sido satisfactorio, solicitamos el ID de registro, lo almacenamos y lo enviamos al servidor de la aplicación para que lo almacene.

```
public String getRegID(Activity _context) {
    String regid = "";
    regid = Preferencias.getGCMID(_context);
    if (!regid.isEmpty()) return regid;
    if (GCMConfig.checkPlayServices(_context)) {
        try {
            GoogleCloudMessaging gcm = GoogleCloudMessaging.getInstance(_context);
            regid = gcm.register(GCMConfig.SENDER_ID);
        } catch (IOException e) {
            e.printStackTrace();
        }
    } else {
        Log.i("", "No valid Google Play Services APK found.");
    }
    return regid;
}
```

Figura 54. Método de obtención del ID de registro

Como podemos observar en la figura 54, debemos comprobar previamente que el dispositivo soporta el servicio de mensajería mediante el método *checkPlayServices* de la clase GCMConfig. A continuación, solicitar el ID de registro pasando como parámetro el SENDER_ID que corresponde al identificador de proyecto que obtuvimos en la consola de desarrolladores tal y como mencionamos en el Anexo 1.

6.5.4 Configuración del servidor de la aplicación

Como hemos comentado anteriormente, el servidor de la aplicación es un contenedor de servlets Java (Tomcat 7.0) que se conecta a una base de datos MySQL.

Para hacer uso de GCM, se puede usar dos tipos de servidores: un servidor de conexión XMPP o un **servidor de conexión HTTP**. Este último es el que se ha seleccionado para este proyecto.

En primer lugar, el servidor de la aplicación debe ser capaz de:

- Comunicarse con la aplicación cliente.
- Enviar solicitudes en el formato correcto al servidor de GCM.
- Debe ser capaz de manejar las solicitudes de respuesta de GCM y enviar el mensaje de nuevo si es necesario.
- También, debe almacenar los identificadores de registro de las aplicaciones cliente, así como de su API KEY obtenida en la consola de desarrolladores de Google.

Para enviar un mensaje al servidor de GCM, se debe enviar una solicitud POST a la dirección `https://android.googleapis.com/gcm/send` especificando en la cabecera del mensaje la Server key y el tipo de contenido que tendrá nuestro mensaje, en nuestro caso: *application/json* para enviar el mensaje en formato JSON.

Dentro del contenido del mensaje deberemos incluir, entre otros, dos campos fundamentales: ID de registro de los dispositivos a los que va dirigido el mensaje y el contenido del mensaje en sí en el formato que hayamos especificado en la cabecera (JSON en nuestro caso).

Una vez enviado el mensaje existen dos tipos de resultados posibles por parte del servidor: el mensaje ha sido procesado con éxito o el mensaje se ha rechazado.

En el segundo caso el servidor GCM nos devuelve información del motivo del rechazo para que podamos tratarla. En la práctica, no se ha tratado esta excepción

y si un mensaje se rechaza en el servidor por algún motivo nuestro servidor no intentará corregir el error y volver a enviarlo. Este es un tema a corregir en una futura versión de la aplicación.

Hay que destacar que en primera instancia se intentó comunicar con el servidor de GCM de esta forma (peticiones post con los parámetros especificados anteriormente) pero los resultados no fueron satisfactorios, por lo que finalmente se optó por basarse en la demo que Google ofrecía para la configuración del servidor[15] y que a continuación explicaremos en detalle.

El paquete (demo) consta de 5 clases java fundamentales que debemos añadir a nuestro proyecto y son las siguientes:

- **InvalidRequestException**: Se usa para gestionar las posibles excepciones.
- **Constants**: clase estática empleada para guardar variables de carácter final que nos servirán en el resto de clases como el ID de nuestro proyecto o la dirección URL a la que se debe hacer las peticiones POST.
- **Message**: Esta clase nos permite generar y gestionar el mensaje que vamos a enviar y los parámetros que le pasaremos.
- **Result**: la clase obtenida una vez enviado el mensaje que nos facilita la gestión del resultado de la petición.
- **Sender**: Es el objeto encargado de enviar el mensaje al servidor de GCM, para inicializarlo debemos pasarle como parámetro el ID de nuestro proyecto.

Por último, haciendo uso de las clases mencionadas podemos enviar un mensaje tal y como se observa en la Figura 55.

```
public NotificationsLogueo(String regid) {
    this.regid = regid;
}

public void send() throws IOException {
    sender = new Sender(SENDER_ID);
    Message message = new Message.Builder().addData("desregistro", "OK").build();
    Result result = sender.send(message, regid, 5);
    String status = "Sent message to one device: " + result;
}
```

Figura 55. Ejemplo envío de mensaje a GCM desde el servidor

En el caso que queramos hacer múltiples envíos, por ejemplo al enviar notificaciones a los usuarios alertando de que un tema ha sido comentado, se ha usado hilos para agilizar el proceso y enviar los mensajes de forma concurrente.

En la aplicación se hace uso del servicio de notificaciones Google Cloud Messaging en los siguientes casos:

- Informar a los usuarios de que **otro usuario ha comentado en un tema** creado por él o en el que haya comentado al menos una vez.
- Se usa como **mensaje de bienvenida** cuando un usuario se registra en la aplicación por primera vez.
- Por último, y con fines menos informativos que en los casos anteriores, se hace uso del servicio notificando a la aplicación cliente de que **un usuario se registró** en otro dispositivo usando las mismas credenciales.

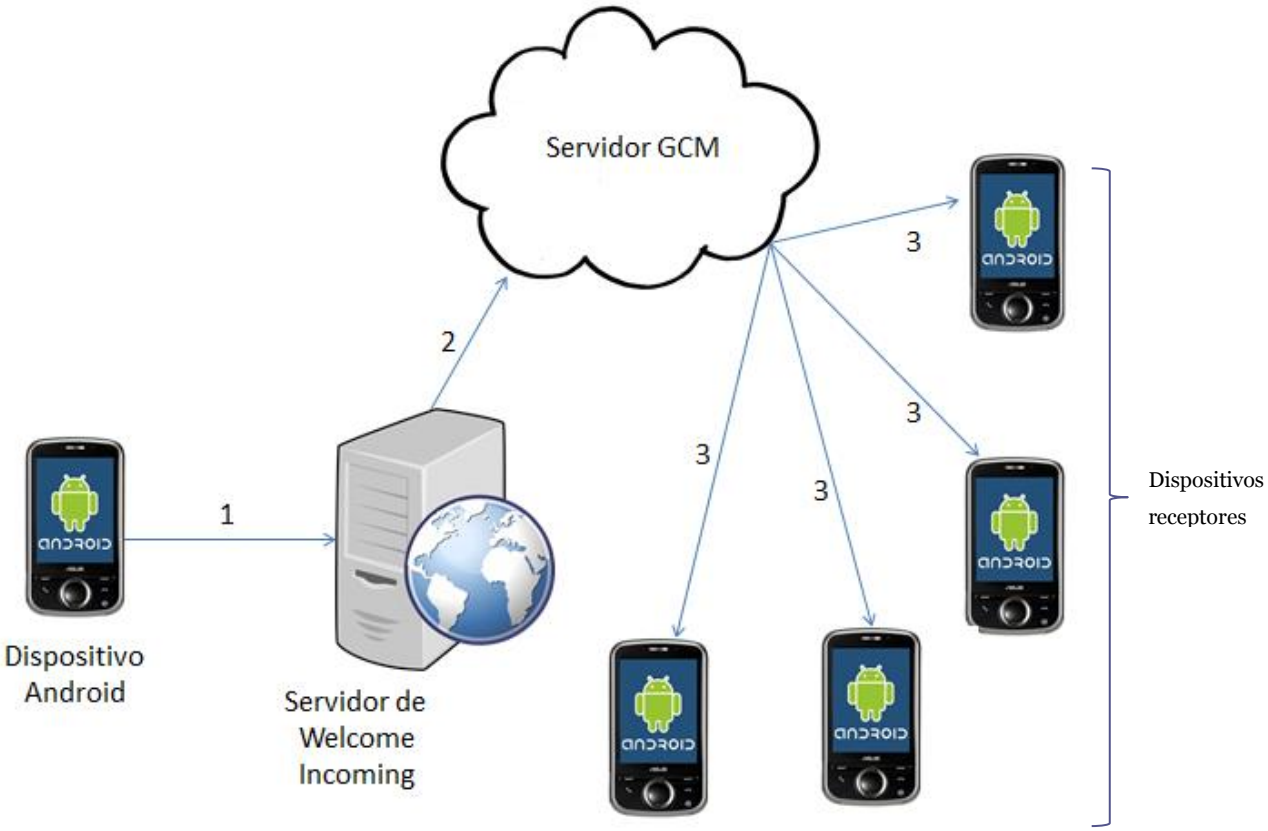

Figura 56. Flujo de la comunicación al realizar un comentario en el foro

En la figura 56 se puede observar cual sería el flujo de la comunicación en cualquiera de los tres casos expuestos anteriormente.

1. En primer lugar, un dispositivo cliente realiza una acción que requiere conexión con el servidor (por ejemplo, comentar un tema).
2. El servidor procesa la petición y envía los mensajes requeridos al servidor de GCM.

3. El servidor de GCM procesa los mensajes y los envía al dispositivo o dispositivos requeridos cuando están en conexión (por ejemplo, a los usuarios que también han comentado un tema concreto).

7 Testeo y pruebas

Para comprobar que la aplicación cumple los objetivos expuestos en la planificación, se desarrollaron varias pruebas para evaluar el rendimiento en distintos dispositivos móviles. También se realizaron pruebas de satisfacción 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 **¡Error! No se ncuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.**, 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 Tabla 2.

- 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 de las pruebas de rendimiento

7.2 Pruebas de satisfacción de usuarios

Otro factor a analizar para saber si la aplicación cumple con los requisitos necesarios de cara al cliente es realizar pruebas para determinar el agrado del usuario.

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 a estos usuarios, donde se preguntaban sobre las siguientes cuestiones:

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

Figura 57. Pregunta 1 de las pruebas de aceptación

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***

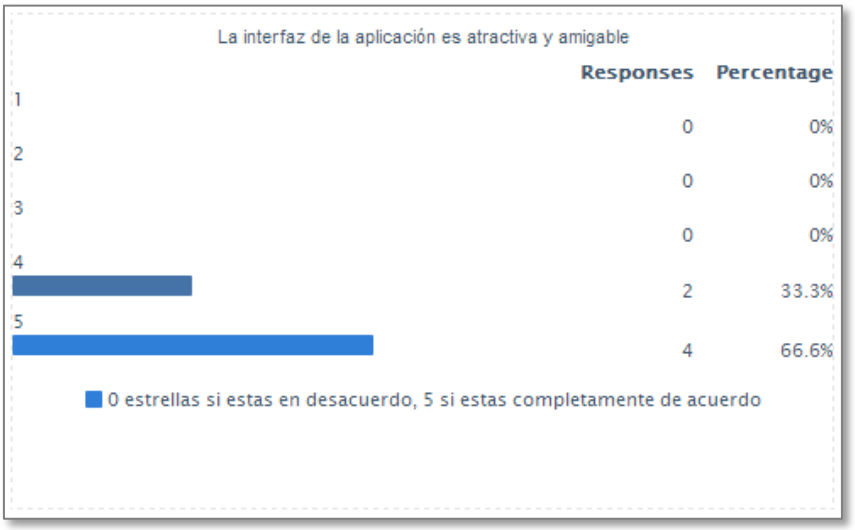

Figura 58. Pregunta 2 de las pruebas de aceptación

Como se puede observar en la Figura 58, 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.

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

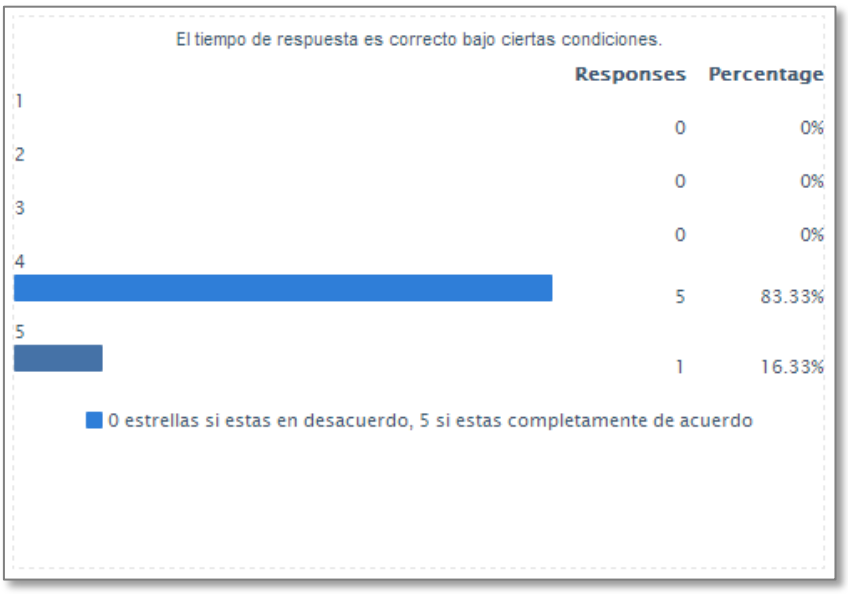

Figura 59. Pregunta 3 de las pruebas de aceptación

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***

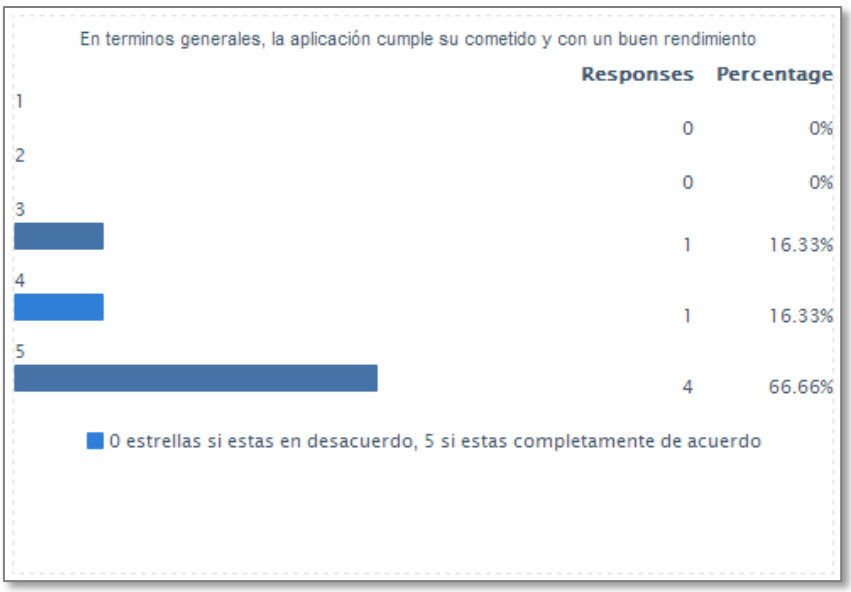

Figura 60. Pregunta 4 de las pruebas de aceptación

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 61. Pregunta 5 de las pruebas de aceptación

Como podemos ver, las opciones más marcadas en la encuesta fueron la interfaz y las funcionalidades. Una vez más, gracias al *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 causado muy buena impresión gracias 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 impuestos para la aplicación, y que los usuarios que la prueban están satisfechos con las funcionalidades que ofrece.

Se debe de tener en cuenta que durante la fase alpha de la aplicación, se puso a disposición de los usuarios de prueba 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

El proyecto realizado responde a las expectativas y requerimientos que se pusieron sobre la mesa en el punto de partida del mismo. Desarrollar una aplicación sobre la plataforma Android con el fin de facilitar la integración de los estudiantes de intercambio en la UPV. Pese a que, a causa de la falta de tiempo, no se han podido implementar todas las funcionalidades que se hubiese deseado, lo cierto es, que se han cumplido los objetivos iniciales del proyecto.

En lo referente al desarrollo de la aplicación, hay que decir que no se disponía de demasiada experiencia en el desarrollo de aplicaciones Android. Únicamente se conocían los aspectos más básicos. A causa de ello, se ha tenido que buscar mucha información y ejemplos de cómo realizar diferentes funciones que requería la aplicación. También hay que decir que, pese a disponer de pequeños conocimientos, era la primera vez que se desarrollaba una capa servidor sobre una aplicación real, con lo cual también se han tenido que investigar diversos aspectos relacionados.

Durante todo el proyecto, se han ido cambiando y retocando partes que ya estaban implementadas debido a diversos factores. En algunos casos, a causa de que se iban aprendiendo nuevas técnicas capaces de mejorar lo que ya había. En otros, por motivos de incompatibilidades entre distintos dispositivos. También, debido a la necesidad de integración con la otra parte de este proyecto. Por ello, el re-trabajo ha sido un aspecto muy a tener en cuenta en el desarrollo.

Otro aspecto determinante en el proyecto, ha sido el escaso tiempo del que se disponía, ya que se designó como tal a pocos meses de la entrega. Hubo que definir diferentes plazos, en los que debían acabarse distintas tareas. Mirando el lado positivo, este contratiempo nos ha enseñado a planificar y trabajar como grupo de desarrollo software.

En conclusión, creo que se ha estado completamente involucrado en el proyecto invirtiendo mucho tiempo y sacrificio con el fin de dar a luz esta primera versión de la aplicación, que ya se encuentra disponible en el Play Store de Google.

9 Trabajo Futuro

Puesto que se trata de una primera versión de la aplicación y dado el amplio abanico de posibilidades que ofrece la plataforma Android para su desarrollo, existen infinitas maneras de mejorar la aplicación UPV Welcome Incoming. A continuación se expondrán las que se consideran más viables y/o necesarias dado el estado actual de la aplicación.

Comenzaremos hablando de la **traducción.** Se han cumplido los objetivos de traducción que se propusieron al inicio del proyecto, no obstante, sería interesante darle un toque distinto a la interfaz de traducción y hacerla más agradable e intuitiva, ya que la actual se queda un poco corta en dispositivos con una densidad de pantalla baja.

Por otro lado, otra mejora imprescindible sería ampliar los **idiomas disponibles** de la aplicación. Actualmente está disponible en español e inglés pero estaría bien adaptarlo, al menos, al resto de idiomas disponibles en el apartado de traductor. Es decir, francés, alemán, italiano, portugués y valenciano.

En lo que al **foro** se refiere, puesto que los objetivos eran la implementación de un foro sencillo, las posibilidades de mejora y ampliación de funcionalidades son infinitas. No obstante algunas mejoras inminentes podrían ser las siguientes:

- Posibilidad de borrar comentarios y temas: Es una propuesta que ya se planteó en la versión actual y al final hubo que desestimarla por falta de tiempo.

- Dividir los comentarios de un tema en páginas: Debido al gran volumen de comentarios que puede llegar a albergar un tema, podría ser que el retardo del servidor en proporcionar tal cantidad de comentarios en una sola petición pudiese resultar frustrante para el usuario. Por ello se pensó cargar un número máximo de comentarios por temas e ir pidiéndolos al servidor cuando sea necesario. Al igual que la opción anterior, esta también se desestimó por falta de tiempo.

- Visualizar más información del tema en las pantallas de lista de temas: Seria interesante visualizar la hora de último comentario y/o el autor de este entre otros datos.

- Calificar comentarios: Habilitar un sistema de puntos similar al de “Me gusta” en Facebook para que los usuarios puedan votar los comentarios que le han resultado de más ayuda, y poder ordenar o filtrar los comentarios en relación a esta escala.

- Administrador: Dar la posibilidad de registrarse como administrador en el foro y de esa forma poder gestionar, borrar y controlar la actividad que se produce por parte de los usuarios.

- Teclado emoji: Permitir a los usuarios introducir emoticonos en los comentarios es una opción que se intentó implementar a lo largo del proyecto y que al final se tuvo que desestimar debido a la aparición de nuevos errores y falta de tiempo. No obstante sería interesante retomarlo en una próxima versión.

- Notificaciones en el servidor: Modificar el diseño de la base de datos, con la finalidad de ofrecer la opción de suscribirse a notificaciones de un tema en el que no hemos comentado. Al almacenar las notificaciones en el servidor, serian de carácter persistente y no se borrarían al desinstalar la aplicación.

Otra mejora de carácter más general, es la **migración del servidor** a una maquina más estable y potente. Que fuese capaz de satisfacer mejor las necesidades de una aplicación en explotación y con la posibilidad de acceso a múltiples usuarios como lo es UPV Welcome Incoming.

Glosario

API: Interfaz de Programación de Aplicaciones, es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

APP: Aplicación móvil.

GCM: Google Cloud Messaging, es el servicio de notificaciones push gratuito que ofrece Google.

ID Registro: Es una cadena de caracteres aleatorios que proporciona el servidor de GCM a una aplicación cuando se registra en el servicio identificando al dispositivo en el que está instalada.

IDE: Entorno de desarrollo integrado, es un programa informático compuesto por un conjunto de herramientas de programación.

JSON: JavaScript Object Notation, es un formato ligero de intercambio de datos.

OPII: Oficina de Programas Internacionales de Intercambio.

SQL: Lenguaje de Consulta Estructurado, es un lenguaje de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

TFG: Trabajo de Fin de Grado.

UML: Lenguaje Unificado de Modelado, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema software.

UPV: Universidad Politécnica de Valencia.

URL: Localizador de Recursos Uniforme, es un identificador que apunta a recursos variables en el tiempo.

Referencias bibliográficas

1-Jimenez M. La venta de tabletas superará por primera vez a la de los PC en 2015. Madrid. *Cinco Días*. Disponible en: http://cincodias.com/cincodias/2014/07/07/tecnologia/1404759006_564333.html

2-Xataka. Historia Android. 2011. Disponible en: http://www.xatakandroid.com/sistema-operativo/que-es-android

3- Arquitectura Android. Agosto 2014. Disponible en: http://www.androidcurso.com/index.php/tutoriales-android-fundamentos/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android

4-Yandex. API traducción java. 2014. Disponible en: http://api.yandex.com/translate/

5-Departemento de sistemas informáticos y computación (DSIC). Análisis y especificación de requisitos. Tema 2b. Modelos de Análisis y Casos de Uso. 17-24. 2013

6-Pages B. Bouml Viewer 6.6.2. 2014. Disponible en http://www.bouml.fr/download.html

7- MySQL. MySQL Workbench.2014. Disponible en: http://www.mysql.com/products/workbench/

8- Developer Android. Android Studio. 2014. Disponible en: https://developer.android.com/sdk/installing/studio.html

9-Wikipedia. Hypertext Transfer Protocol. 2014. Disponible en: http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol

10-Andorid developers. Fragment. 2013. Disponible en: http://developer.android.com/guide/components/fragments.html

11-Tomcat Apache. Servlet. 2013. Disponible en: http://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/Servlet.html

12- Jan Muller. cropimage. 2013. Disponible en: https://github.com/biokys/cropimage

13- Android Developer. Google Cloud Message. 2013. Disponible en: http://developer.android.com/google/gcm/index.html

14-Android Developer. Implementing GCM Client. 2013. Disponible en: http://developer.android.com/google/gcm/client.html

15- Android Developer. GCM demo. 2013. Disponible en: https://code.google.com/p/gcm/

Anexos

Anexo A. *Crear un proyecto en la consola de desarrolladores de Google y habilitar el servicio de GCM*

Para hacer uso del servicio de Google Cloud Messaging en la aplicación es necesario activarlo de la siguiente forma:

En primer lugar, debemos crear un proyecto desde la página principal de la consola de desarrolladores: https://console.developers.google.com/project haciendo clic sobre el botón "Create Project".

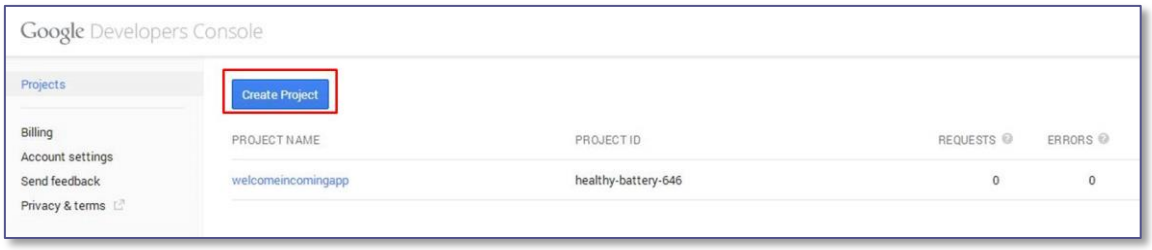

Figura 62. Creando proyecto en la consola de desarrolladores Google (1)

Nos aparecerá la siguiente ventana donde deberemos introducir el nombre del proyecto, en este caso "welcomeincomingapp" y nos generará un ID de proyecto aleatorio que usaremos para registrarnos desde la aplicación cliente.

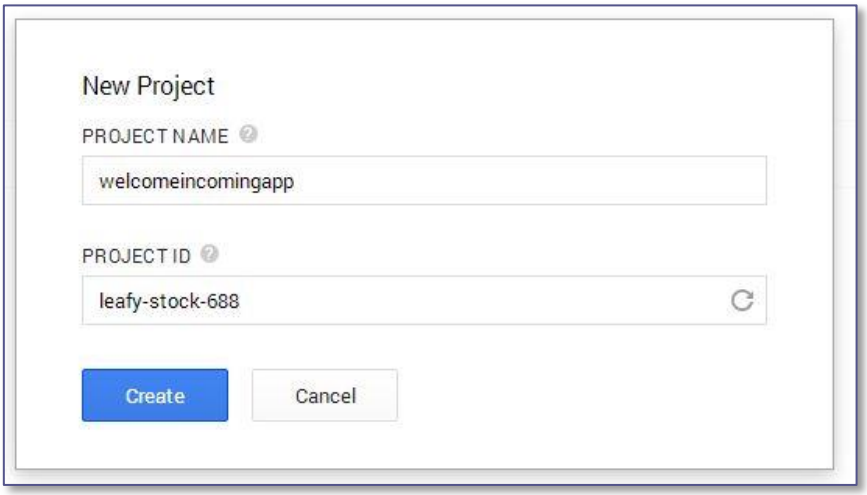

Figura 63. Creando proyecto en la consola de desarrolladores Google (2)

Una vez creado el proyecto accedemos al apartado **APIs & auth** del panel de la izquierda y activamos el servicio **Google Cloud Messaging for Android.**

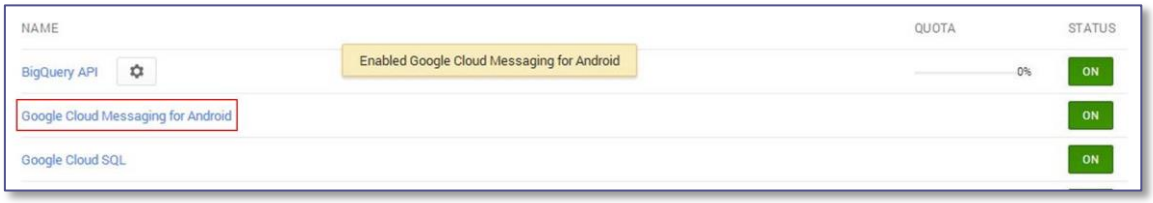

Figura 64. Activación del servicio GCM

Posteriormente, accedemos al apartado **Credentials** y pulsamos sobre "Create New Key" en el aparatado clave de acceso pública.

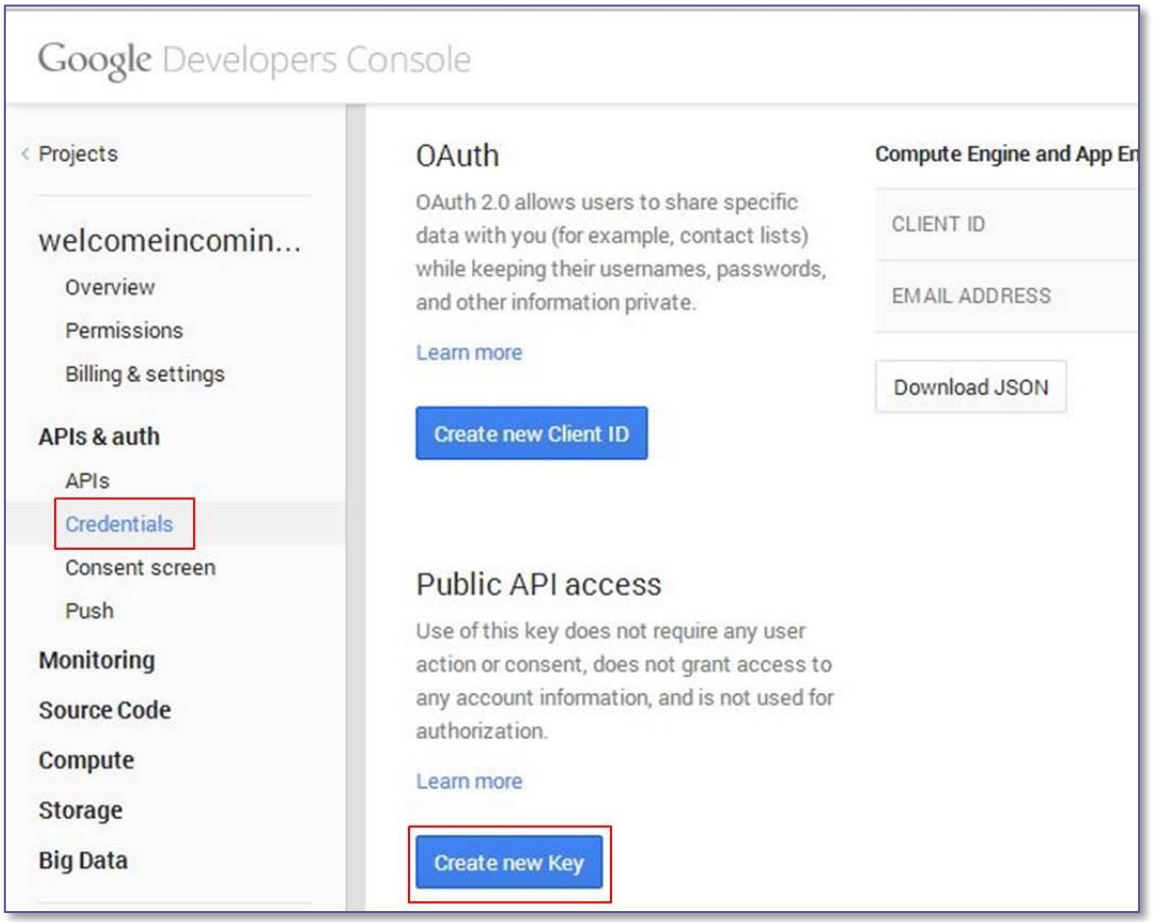

Figura 65. Obtención de una "Server key"

En la siguiente ventana introducimos la dirección IP del servidor de la aplicación desde el que enviaremos las notificaciones. Nos generará una “Server key” que usaremos en dicho servidor para enviar mensajes a GCM.

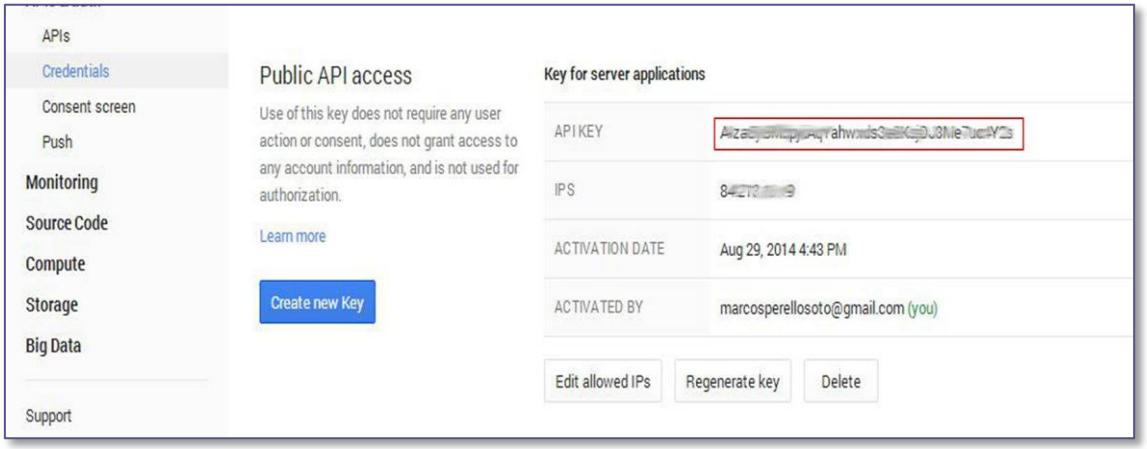

Figura 66. Parámetros de la API de acceso público de Google