Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

APLICACIÓN MÓVIL PARA EL CONGRESO PROCESAMIENTO DEL LENGUAJE NATURAL

Alumno: Juan García Anguita

Tutores: Prof. D. L. Alfonso Ureña López
D. Eugenio Martínez Cámara
Dpto.: Departamento de Informática

Septiembre, 2015

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En este capítulo procederé a la presentación de mi trabajo de fin de grado (TFG). Tras la lectura de esta introducción, el lector conocerá diferentes aspectos sobre mi trabajo, desde el propósito y las motivaciones del mismo, hasta los objetivos y metodología a seguir. El lector descubrirá que no solo se trata de un proyecto software en el que se emplean los conocimientos básicos sobre ingeniería del software y programación obtenidos durante el desarrollo del grado, si no que se amplían dichos conocimientos con la programación para móviles, la cual no se ha tratado durante el grado.

1.1. Propósito del proyecto

El propósito central del proyecto es la construcción de una aplicación móvil para un congreso, en concreto para un congreso a cerca del Procesamiento del Lenguaje Natural (PLN). Sin embargo, no solo se trata de crear una aplicación para móviles para un congreso específico (como es el congreso sobre el PLN), si no que trataré de desarrollar la aplicación de la manera más general posible, pudiendo ser esta adaptada para su uso en otros tipos de congreso con pequeños cambios en la misma. Como resultado final se obtendrá una aplicación móvil que contendrá toda la información sobre el congreso del PLN totalmente usable.

1.2. Motivaciones del proyecto

La Sociedad Española para el Procesamiento del Lenguaje Natural es una sociedad científica de ámbito nacional que tiene como fin el fomento, apoyo y difusión de la investigación relacionada con las Tecnologías del Lenguaje Humano. La Sociedad organiza cada año un congreso en el mes de septiembre al que suelen asistir los investigadores más importantes a nivel nacional e internacional en el ámbito de las Tecnologías del Lenguaje Humano. El Congreso suele tener una duración de tres días, y cada día está sujeto a un programa en el que se reparten sesiones de exposiciones de artículos, de ponencias invitadas, sesiones de exposición de póster, muestra de demostraciones, y en ocasiones diversos talleres de trabajo.

Hasta la fecha, a cada asistente al Congreso se le entrega un impreso con el programa del mismo. La impresión tiene un coste considerable cuando el número de participantes es elevado, y además si se analiza se trata de un gasto superfluo ya que la mayoría de los concurrentes se deshacen del impreso con el programa nada más finalizar el Congreso.

Desde hace unos años, el uso de los teéefonos móviles táctiles (Smartphones) y las tabletas electrónicas ha incrementado exponencialmente. Actualmente todo el mundo dispone de un smartphone o una de estas tabletas electrónicas, y los usa día a día. Esto se debe a la facilidad de uso y disponibilidad de la información que presentan estos dispositivos. Debido al incremento de este mercado, practicamente todas las empresas o eventos

disponen de una aplicación móvil (Booking[1], Wizzair[2], EVO[3], McDonalds[4], aplicaciones de festivales como Arenal Sound[5] o Tomorrowland[6], etc.). De esta manera se garantiza el acceso a la información de manera gratuita, rápida y sencilla en cualquier lugar del mundo. Por lo tanto, la Sociedad se ha planteado el desarrollo de una aplicación para este tipo de dispositivos.

1.3. Objetivos del proyecto

El principal objetivo de este proyecto es el desarrollo de una aplicación móvil que dé cavidad a toda la información del congreso anual organizado por la Sociedad Española para el Procesamiento del Lenguaje Natural. Debido a que nunca he desarrollado una aplicación móvil, ya que esta disciplina no se incluye en el grado (sólo de manera optativa), este trabajo incluirá horas de trabajo previo en las que investigaré y aprenderé cómo funcionan las diferentes tecnologías móviles, además de su popularidad, ya que uno de los objetivos es que la aplicación esté disponible para el mayor número de personas posible.

En cuanto a los objetivos concretos a satisfacer por la aplicación, podemos encontrarlos enumerados a continuación:

- Ofrecer información sobre el congreso, lugar, accesos, cómo llegar a la ciudad y a la sala de conferencias, hoteles cercanos...
- Proporcionar el programa del congreso.
- Poder acceder a los documentos relacionados con el congreso, artículos, ponencias, informes...
- Poder destacar aquellos eventos del congreso que cada participante quiera asistir.
- Notificar con antelación los eventos destacados por el usuario.
- Integración con redes sociales para facilitar que los asistentes compartan su experiencia.

[1] www.booking.com
[2] www.wizzair.com
[3] www.evobanco.com
[4] www.mcdonalds.es
[5] www.arenalsound.com
[6] www.dreambeach.com

Además de estos objetivos ya presentados, se podría decir que uno de los objetivos personales del desarrollo de este proyecto es el aprendizaje a cerca del desarrollo de aplicaciones móviles. Ya que considero que es un mercado muy amplio en la actualidad y sería interesante ampliar los conocimientos obtenidos en el grado con los del desarrollo de aplicaciones móviles.

1.4. Resultados esperados

Tras finalizar el proyecto, se deben obtener como resultados:

- Aplicación móvil para el congreso anual sobre el PLN organizado por la Sociedad Española para el Procesamiento del Lenguaje Natural que satisfaga todas las peticiones propuestas por la Sociedad.
- Memoria del proyecto.
- Anexo que informe sobre el mantenimiento de la aplicación, principalmente para su actualización en congresos futuros.

1.5. Metodología del proyecto

Toda aplicación software se construye con un objetivo, y en la mayoría de los casos, por no decir siempre, ese objetivo no es otro que satisfacer una necesidad de los usuarios al que va dirigido. Para satisfacer esta necesidad previamente debemos conocer cuál es esta necesidad, por esto todo proyecto de ingeniería comienza con la definición de unos requisitos impuestos por el usuario o supuestos usuarios a los que va dirigido el producto y que el producto final debe cumplir. Para asegurar empiricamente que se conseguirá lo pactado con los clientes, los ingenieros necesitan de una metodología, de técnicas y procedimientos ideados para tal fin.

Este proyecto trata de satisfacer la necesidad de los asistentes a un congreso de disponer de la información sobre el mismo (artículos, ponencias, ubicaciones, itinerario, etc.) en todo momento y lugar. Como proyecto de ingeniería que es, sigue una metodología para que el sistema final alcance su proposito. La metodología que he escogido para el desarrollo de la aplicación es una de las más comunes en el desarrollo de software, y su nombre es Proceso Unificado. Se trata de un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes

áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. Algunas de sus características más destacables son:

- Está basado en componentes, lo cual quiere decir que el sistema en construcción está formado por componentes software interconectados a través de interfaces bien definidas.
- Utiliza el Lenguaje Unificado de Modelado, UML, para preparar todos los esquemas de un sistema software.
- Está dirigido por los casos de uso, es decir, está dirigido por las necesidades de los usuarios.
- Está centrado en la arquitectura. A partir de los casos de uso se va definiendo la arquitectura del sistema.
- Sigue un modo de trabajo iterativo e incremental. Ésto es importantísimo porque flexibiliza el desarrollo del proyecto permitiendo, en cualquier momento, modificar fases ya completadas para mejorar la calidad del producto final.

El flujo de trabajo que se va a seguir en la realización del proyecto es:

- **Análisis**, en el que se capturarán y describirán en detalle los requisitos del sistema, los perfiles de los usuarios potenciales de la aplicación, y por último se definirá y describirá la interacción de los usuarios.
- **Diseño**. A partir del análisis se describirá la solución del problema.
- **Implementación**. En esta fase se describirá la construcción del sistema.
- **Prueba**. Se indicarán las pruebas que se realizan al sistema para comprobar que los requisitos descritos en la etapa de análisis se cumplen.

Esta metodología se combina con el “Diseño centrado en el usuario”, porque el objetivo de este proyecto, como el de otros de caracter informático, es satisfacer a los usuarios, siendo estos uno de los pilares fundamentales del proyecto.

2. CONTEXTO DEL PROYECTO

En el segundo capítulo de esta memoria se tratará el ámbito de la programación de aplicaciones móviles. Puesto que el objetivo principal del proyecto es desarrollar una aplicación destinada a un congreso se describirán en este apartado las características principales de esta subsección de la ingeniería informática. Tras la lectura de dicho capítulo el lector tendrá una idea general de las diferentes tecnologías y metodologías existentes destinadas a la programación móvil, así como su frecuencia de uso y su popularidad.

2.1. Programación orientada a dispositivos móviles

En la actualidad, trabajar en el campo de la programación para dispositivos móviles se hace necesario, debido a que las empresas se deben adaptar a las tendencias del mercado y a las necesidades de sus clientes. Por lo que se debe pensar en la posibilidaad de tener acceso a la información en cualquier lugar y en cualquier instante, a través de distintos dispositivos móviles, incluidos dentro de la administración de la empresa, al igual que las soluciones informáticas para equipos de escritorio. Convirtiéndose en parte vital para el funcionamiento de los procesos empresariales.

Pero antes de adentrarnos en el mundo de la programación móvil, debemos responder a una pregunta: ¿qué entendemos por un dispositivo móvil? Y e aquí la respuesta: un dispositivo móvil es una computadora de tamaño pequeño, con capacidades de procesamiento, con conexión a Internet, con memoria, diseñado especificamente para una función, pero que pueden llevar a cabo otras funciones más generales[7].

A continuación se procederá al estudio de los diferentes dispositivos móviles existentes en el mercado actual y que han existido con anterioridad, siempre con vista al futuro de este campo que avanza tan deprisa. Se comenzará con una introducción en el mundo del hardware y software, para proseguir con un estudio de las diferentes tecnologías existentes para el desarrollo de aplicaciones móviles y un estudio de mercado en el que se podrá distinguir que tipo de dispositivo es el más empleado, el software más empleado y la tecnología más empleada.

2.1.1. Dispositivos móviles. Hardware

A la hora de estudiar el hardware existente en el campo de los dispositivos móviles debemos distinguir entre las diferentes categorías de dispositivos móviles existentes[8]:

- **Dispositivo móvil de datos limitado**: son aquellos dispositivos que disponen de una pantalla pequeña, principalmente basada en pantalla de tipo texto con servicios de datos generalmente limitados a SMS y acceso

[7] Definición obtenida de la web: http://es.wikipedia.org/wiki/Dispositivo_m%C3%B3vil
[8] Clasificación obtenida de la web: http://es.wikipedia.org/wiki/Dispositivo_m%C3%B3vil

a WAP. Un ejemplo de este tipo de dispositivo son los teléfonos móviles de antaño.

- **Dispositivo móvil de datos básico**: dispositivos que tienen una pantalla de mediano tamaño, (entre 30x120 y 240x240 píxeles), menú o navegación basada en iconos por medio de una "rueda" o cursor, y un navegador web básico. Un típico ejemplo de este tipo de dispositivos son los BlackBerry y los teléfonos inteligentes.
- **Dispositivo móvil de datos mejorado**: son los disposivos que tienen pantallas de medianas a grandes (por encima de los 240x120 píxeles), navegación de tipo stylus, y que ofrecen las mismas características que el dispositivo móvil de datos básicos más aplicaciones nativas como Microsoft Office Mobile[9] (Word, Excel, PowerPoint) y aplicaciones corporativas usuales, en versión móvil, como Dropbox[10], portales intranet, etc. Este tipo de dispositivos incluyen sistemas operativos muy desarrollados. A continuación veremos los diferentes sistemas operativos más empleados en la actualidad, un ejemplo de uno de estos sistemas es Android.

Clasificados dentro de estas categorías existen diferentes tipos de dispositivos móviles[11]:

- **Teléfono inteligente**: es un tipo de teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar actividades, semejante a la de una minicromputadora, y con una mayor conectividad que un teléfono móvil convencional.[12]

Ilustración 2.1. Samsung Galaxy S5

[9] https://products.office.com/es-es/mobile/office
[10] https://www.dropbox.com/
[11] Clasificación obtenida de la web: http://es.wikipedia.org/wiki/Dispositivo_m%C3%B3vil
[12] Definición obtenida de la web: http://es.wikipedia.org/wiki/Tel%C3%A9fono_inteligente

- **Tableta**: es una computadora portátil de mayor tamaño que un teléfono inteligente o un PDA, integrada en una pantalla táctil con la que se interactúa pimariamente con los dedos, sin necesidad de teclado físico ni ratón.[13]

Ilustración 2.2. Ipad

- **Tabléfono**: dispositvos electrónicos móviles o portátiles, con pantallas táctiles entre 5 y 7 pulgadas aproximadamente y con múltiples prestaciones de hardware y software.[14]
- **Videoconsola portátil**: es un dispositivo electrónico ligero que permite jugar videojuegos y que, a diferencia de una videoconsola clásica, los controles, la pantalla, los altavoces y la alimentación básica (baterías) están todos integrados en la misma unidad y todo ello con un pequeño tamaño, para poder llevarla y jugar en cualquier lugar o momento.[15]
- **Cámara digital**: es una cámara fotográfica que, en vez de captar y almacenar fotografías en película química como las cámaras fotográficas de película fotográfica, recurre a la fotografía digital para generar y almacenar imágenes.[16]

[13] Definición obtenida de la web: http://es.wikipedia.org/wiki/Tableta_(computadora)
[14] Definición obtenida de la web: http://es.wikipedia.org/wiki/Tabl%C3%A9fono
[15] Definición obtenida de la web: http://es.wikipedia.org/wiki/Videoconsola_port%C3%A1til
[16] Definición obtenida de la web: http://es.wikipedia.org/wiki/C%C3%A1mara_digital

Ilustración 2.3. Cámara digital junto a caja de cerillas para comparar tamaños.

- **Otros dispositivos móviles**: encontramos muchos otros tipos de dispositivos móviles en el mundo actual. Algunos de estos son: netbook, nettop, handheld, ordenador de bolsillo, PDA o el reloj inteligente.

Ilustración 2.4. Niños trabajando con netbooks.

Ilustración 2.5. Ejemplo de PDA.

2.1.2. Dispositivos móviles. Software

En cuanto al software orientado a dispositivos móviles, se va a comenzar por la definición de la capa más básica: el sistema operativo. Un sistema operativo móvil es un sistema operativo que controla un dispositivo móvil al igual

que los PCs utilizan Windows o Linux, entre otros. Sin embargo, los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos.

Algunos de los sitemas operativos utilizados en los dispositivos móviles están basados en el modelo de capas, pero antes de adentrarnos en el funcionamiento de los mismos se va a realizar un breve resumen de la historia general o “timeline” de los sistemas operativos móviles. Posteriormente se realizará una incursión en los tres sistemas operativos móviles más empleados en la actualidad, para después realizar un estudio de mercado. Finalmente, en la *Sección 2.2* se tomará la decisión de sobre que sistema operativo desarrollar la aplicación.

2.1.2.1. Historia

La historia de los sistemas operativos empleados actualmente comienza el 1 de enero de 1996, con el lanzamiento de Palm OS, sistema operativo desarrollado por Palm Inc. Para PDAs. Este sistema hacía uso de una interfaz gráfica para su control táctil, de esta manera convertía una PDA en un dispositivo de utilización sencilla. Posteriormente, en los años 1997 y 1998, este obtendría su actualización a las versiones 2.0 y 3.0 respectivamente. El 11 de enero de 1999 aparecería Blackberry OS 1.0, en el año 2000 Pocket PC y entre 2001 y 2002 surgieron varias actualizaciones para los sistemas ya mencionados anteriormente. En junio de 2003 salió a la palestra el conocido sistema operativo Windows Mobile 2003, desarrollado por Microsoft, y diseñado para su uso en teléfonos inteligentes y otros dispositivos móviles. Este estaba basado en el núcleo del sistema operativo Windows CE y cuenta con un conjunto de aplicaciones básicas utilizando las API de Microsoft Windows. Esá diseñado para ser similar a las versiones de escritorio de Windows esteticamente, además existe una gran oferta de software de terceros disponible para dicha plataforma.

En julio de 2005 Google compra Android Inc., dando así el que sería su primer paso en su exitosa inmersion en el mundo de la telefonía móvil. Por otra parte, en enero de 2007 Apple lanza su iPhone con su propio sistema operativo iPhone OS. También en 2007 aparecen algunas actualizaciones de los sistemas operativos ya expuestos con anterioridad. A finales de 2007, Google anuncia el

desarrollo de Android OS. Llega 2008, y a mediados de 2008 Apple anuncia su iOS 2.0, en agosto Blackberry saca su Blackberry OS 5.0 y sería en septiembre de 2008 cuando Google lanzara la primera versión de Android.

Durante los próximos años Android e iOS coparían el mercado de la telefonía móvil, con sus respectivas actualizaciones casi anuales de los mismos. En octubre del año 2010 Microsoft anunciaría su Windows Phone 7, su propio sistema operativo para teléfonos móviles inteligentes. Este sistema apuntaba a hacer la competencia de los dos sistemas ya mencionados anteriormente. Pero este intento fue un fracaso, ya que consiguió introducirse en el mercado pero con un éxito mínimo. Finalmente, durante los años restantes hasta la actualidad han ido apareciendo diferentes actualizaciones de los diferentes sistemas, actualmente tenemos iOS 8, Android 5.0 Lollipop y Windows Phone 8.1.

En cuanto al futuro en la historia de los sistemas operativos todo apunta a que todo seguirá tal y como está. Google y Apple seguirán actualizando sus exitosos sistemas, mientras que Microsoft seguirá tratando de desbancar a estas dos anteriores del trono de los sistemas operativos para dispositivos móviles.[17]

2.1.2.2.iPhone OS (iOS)

iOS es un Sistema operativo móvil de la multinacional Apple Inc. Originalmente desarrollado para el iPhone, después se ha usado en dispositivos como el iPod touch y el iPad. No permite la instalación de iOS en hardware de terceros. Tenía 26% de cuota de mercado de sistemas operativos móviles más vendidos en el último cuatrimestre de 2010, detrás de Google Android y Nokia.

En enero-febrero de 2014, más del 52% de todo el mundo que posee iPhone, iPod touch o iPad tienen instalado en su dispositivo iOS 8, su última versión. Este es uno de los grandes puntos a favor de este sistema operativo respecto a su máximo competidor, Android, ya que practicamente no existe fragmentación. Pero trataremos este tema más adelante.

Los elementos de control consisten de elementos que se deslizan, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslizamiento, toques, pellizcos, los cuales tienen definiciones diferentes

[17] Información extraída de la web: http://www.timetoast.com/timelines/historia-de-sistemas-operativos-moviles

dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo (por ejemplo, para el comando deshacer) o rotarlo en tres dimensiones (un resultado común es cambiar de modo vertical al apaisado u horizontal).[18]

iOS se deriva de OS X, que a su vez está basado en Darwin BSD, y por lo tanto es un sistema operativo tipo Unix. iOS cuenta con cuatro capas de abstracción[19]:

- **Capa del núcleo del sistema operativo (Core OS)**: Contiene las características de bajo nivel: ficheros del sistema, manejo de memoria, seguridad, drivers del dispositivo.
- **Capa de medios (Media)**: Provee los servicios de gráficos y multimedia a la capa superior.
- **Capa de servicios principales (Core services)**: Contiene los sevicios fundamentales del sistema que usan todas las aplicaciones.
- **Capa de "Cocoa Touch"**: Cocoa Touch es la capa más importante para el desarrollo de aplicaciones iOS. Posee un conjunto de Frameworks que proporciona el API de Cocoa para desarrollar aplicaciones. Se podría decir que Cocoa Touch proviene de Cocoa, la API ya existente en la plataforma MAC. Esta capa está formada por dos Frameworks fundamentales:
    - **UIKit**: contiene todas las clases que se necesitan para el desarrollo de una interfaz de usuario.
    - **Foundation Framework**: define las clases básicas, acceso y manejo de objetos, servicios del sistema operativo.

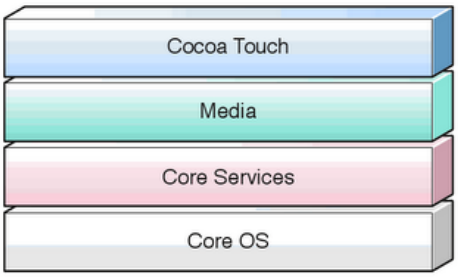

Ilustración 2.6. Esquema con las diferentes capas de iOS

[18] Información extraída de la web: https://es.wikipedia.org/wiki/IOS
[19] Clasificación obtenida de la web: https://sites.google.com/site/tecnologiaiostm/desarrollo-de-aplicaciones/arquitectura-ios

2.1.2.3. Android[20]

Android es un Sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tablets; y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde (como ya se ha expuesto en la *Sección 2.1.2.1. Historia*), en 2005, compró. Android fue presentado en 2007 junto a la fundación del Open Handset Alliance (un consorcio de compañías de hardware, software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008. Los dispositivos de Android venden más que las ventas combinadas de Windows Phone e iOS. El éxito del sistema operativo se ha convertido en objeto de litigios sobre patentes en el marco de las llamadas «Guerras por patentes de teléfonos inteligentes» (en inglés,S martphone patent wars) entre las empresas de tecnología.

La versión básica de Android es conocida como Android Open Source Project (AOSP).

El 25 de junio de 2014 en la Conferencia de Desarrolladores Google I/O, Google mostró una evolución de la marca Android, con el fin de unificar tanto el hardware como el software y ampliar mercados. Para ello mostraron nuevos productos como Android TV, Android Auto, Android Wear o una serie de “smartphones” de baja gama bajo el nombre de Android One. Esto sirvió para estabilizar la imagen de la marca de cara a los mercados y al público.

Los componentes principales del sistema operativo de Android son:

- **Aplicaciones**: las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.
- **Marco de trabajo de aplicaciones (Framework de alicaciones)**: los desarrolladores tienen acceso completo a las mismas APIs del framework usadas por las aplicaciones base. La arquitectura está diseñada para

[20] Información obtenida de la web: https://es.wikipedia.org/wiki/Android

simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

- **Librerías**: Android incluye un conjunto de librerías de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android; algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos 3D y SQLite, entre otras.
- **Runtime de Android**: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato .dex por la herramienta incluida "dx".
- **Núcleo Linux (Kernel de Linux)**: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

Ilustración 2.7. Esquema con las diferentes capas de Android.

2.1.2.4. Windows Phone[21]

Windows Phone es un Sistema operativo móvil desarrollado por Microsoft, como sucesor de Windows Mobile. A diferencia de su predecesor está enfocado en el mercado de consumo en lugar de en el mercado empresarial. Con Windows Phone; Microsoft ofrece una nueva interfaz de usuario que integra varios de sus servicios propios como OneDrive, Skype y Xbox Live en el sistema operativo. Compite directamente contra Android de Google e iOS de Apple. Su última versión disponible y definitiva es Windows 8.1, lanzado en 2014.

Debido a la evidente fragmentación de sus sistemas operativos, Microsoft anunció en 2015 que dará de baja a Windows Phone, para enfocarse en un único sistema más versátil denominado Windows 10, disponible para todo tipo de plataformas.

Puesto que este sistema está abocado a su desaparición, no se seguirá estudiando en el presente documento. La opción de desarrollar la aplicación móvil para dicho sistema operativo queda totalmente descartada, ya que con el desarrollo de la aplicación móvil para el Congreso sobre el Procesamiento del Lenguaje Natural se pretende acceder al mayor público posible, siempre con vista hacia el futuro.

2.1.3. Dispositivos móviles. Estudio de mercado

En este apartado se procederá a realizar un estudio del mercado de los dispositivos móviles, tanto de los diferentes sistemas operativos como de los diferentes dispositivos existentes, cuáles son los más usados en la actualidad y cuál es la tendencia para el futuro.

Para comenzar, vamos a ver un gráfico en el que se muestra como el uso del “smartphone” o móvil inteligente en España ha incrementado considerablemente, mientras que el uso de los móviles antiguos y el 3G ha decrementado de manera muy pronunciada:

[21] Información extraída de la web: https://es.wikipedia.org/wiki/Windows_Phone

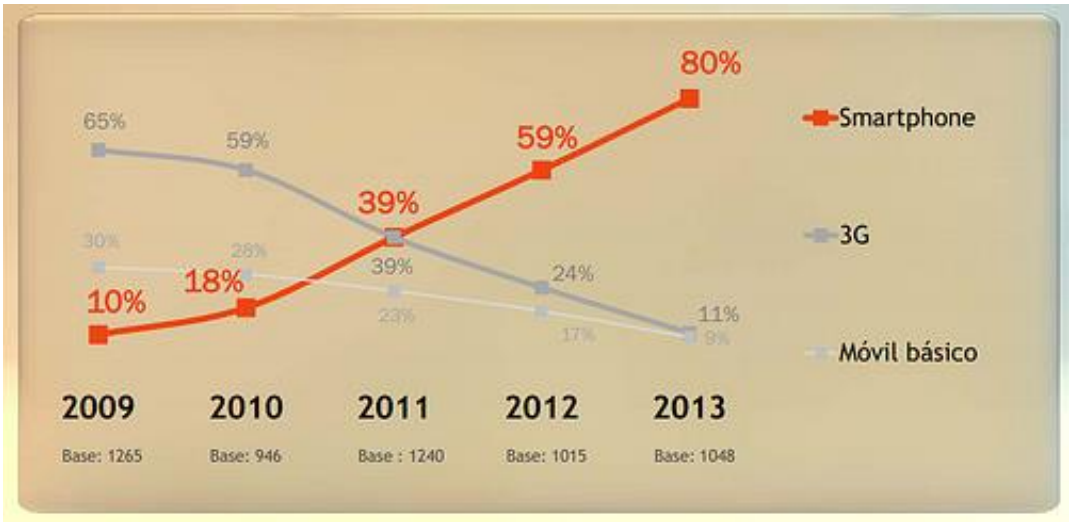

Ilustración 2.8. Gráfico uso Smartphone. Fuente: *http://www.universoabierto.com/11140/mercado-espanol-de-dispositivos-moviles/*, publicado en septiembre de 2013 y consultado en julio de 2015.

A continuación podemos ver un gráfico de barras en el que distinguimos el porcentaje de smartphones y tablets que hacen uso de diferentes aplicaciones. Por ejemplo, WhatsApp se encuentra instalada en el 70% de los smartphones, mientras que Facebook es la más instalada en las tabletas electrónicas con un 34%. Con este gráfico podemos ver que en su mayoría, los smartphones y tablets son empleados para el ocio: juegos, redes sociales, comunicación, etcétera.

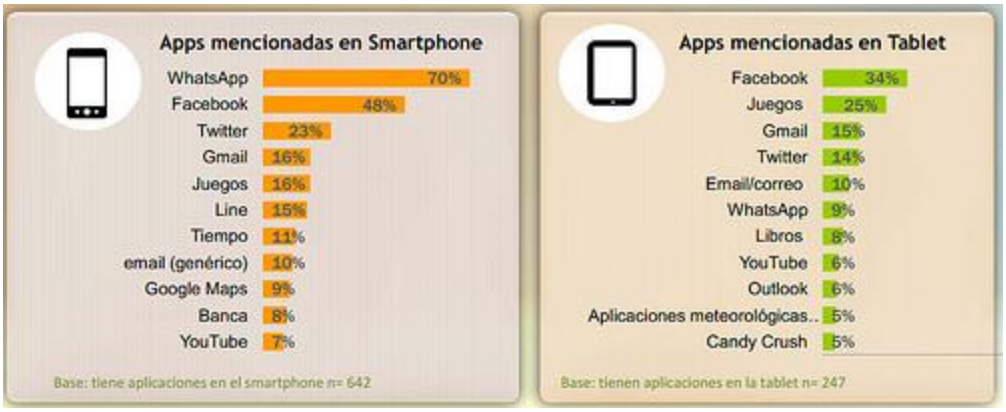

Ilustración 2.9. Gráfico aplicaciones más usadas en Smartphone y tabletas. Fuente: *http://www.universoabierto.com/11140/mercado-espanol-de-dispositivos-moviles/*, publicado en septiembre de 2013 y consultado en julio de 2015.

Como conclusión a este breve estudio de mercado de los dispositivos móviles se puede decir que en los últimos años se ha producido la consolidación del consumo de smartphones, pasando del 58% en 2012 al 80% en 2013. También se ha producido una gran subida en el consumo de tablets, se ha

pasado del 23% al 43% en un año. Por marcas, Samsung aumenta su dominio pasando del 35% en 2012 al 42% en 2013. Sube la penetración de Sony y LG y cae la de Apple, HTC, Nokia, Blackberry y Hueawei. El Smartphone se iguala con el portátil como punto de acceso diario de Internet (86% de los usuarios acceden desde el portátil, frente al 88% que accede desde el smartphone). El acceso a Internet en tablet llega al 45%. En móvil la media de conexión en usuarios diarios es de 2 horas 30 minutos. Según el estudio, 8 de cada 10 usuarios de smartphones pasa más de 1 hora conectado. El móvil se usa un 50% para comunicación social (whatssapp, redes sociales), un 20% para actividades lúdicas (noticias, contenidos), un 16% para llamadas y un 14% para navegar por Internet. Se dobla el acceso a Internet en tablet desde apps, pasando del 31% en 2012 al 62% en 2013. Por lo tanto, después de todo esto se puede decir que la aplicación se desarrollará orientada a su uso en tablets y smartphones, ya que son los dispositivos móviles más empleados en la actualidad.[22]

El siguiente paso será la realización de un breve pero muy útil estudio del mercado de los sistemas operativos móviles. Ya conocemos el funcionamiento de los mismos y ahora se precisará de conocer las diferentes cuotas de mercado de los mismos para determinar sobre que plataforma será más útil el desarrollo de la aplicación móvil para el congreso de procesamiento del lenguaje natural.

Para comenzar con este segundo estudio se va a exponer un sencillo diagrama en el que se muestran los porcentajes de uso de los diferentes sistemas operativos móviles en España. Como ya se hizo anteriormente, el estudio de mercado se realiza única y exclusivamente sobre la población española, ya que es el público al que va a ir dirigida la aplicación a desarrollar. En el gráfico vemos como Android domina el mercado sobradamente con un 88,6% de cuota de mercado. Mientras tanto, el segundo en la lista es iOS con un 8,8%, casi 80 puntos menos. La diferencia es abismal, por lo que no será muy complicado tomar una decisión. Finalmente, vemos como Windows sigue a estos dos con un 2,5%, BlackBerry totalmente desaparecido con 0% y otros sistemas toman el 0,1% del mercado. Este gráfico ha sido extraído de la web

[22] Información extraída de la web: http://www.ticbeat.com/tecnologias/android-domina-mercado-mundial-smartphones/

www.kantorworldpanel.com dónde se pueden localizar estas mismas estadísticas para diferentes paises y en diferentes momentos.

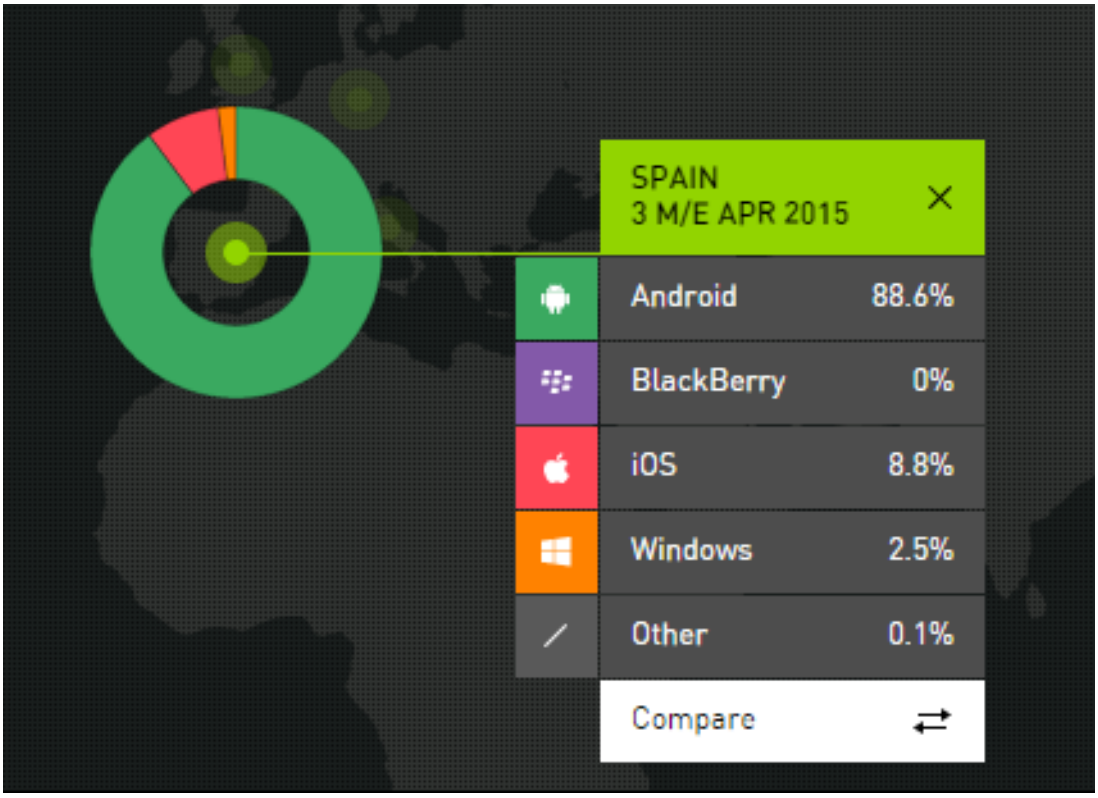

Ilustración 2.10. Gráfico cuota mercado sistemas operativos móviles

Tras este breve análisis queda claro que Android es el sistema operativo más empleado en España. Pero, como sabemos, existen diferentes versiones de los diferentes sistemas operativos. Sería interesante estudiar que porcentaje de los usuarios de los diferentes sistemas operativos utiliza cada versión.

Se comenzará por el desglose del uso de las diferentes versiones de Android, ya que es el sistema operativo más usado en España. En la siguiente tabla se puede apreciar como hay hasta nueve versiones de Android en uso. Se distingue como la versión más usada del mismo es la 4.4 KitKat con un 39,2% de cuota de mercado, le sigue la versión 4.2 de Jelly Bean con un 17,5%, después se puede ver como la siguiente es la versión 4.1 de Jelly Bean con un 14,7%, tras esta la versión 5.0 de Lollipop (una de las versiones más actuales), le siguen las versiones 2.3 Gingerbread, 4.0 Ice Cream Sandwich y 4.3 Jelly Bean con algo más de un 5% cada una, para finalizar con las versiones 2.2 Froyo y 5.1 Lollipop (la más actual) las cuales no llegan a un 1% de cuota de mercado. Con este estudio se puede ver la famosa fragmentación de Android, uno de los grandes inconvenientes del desarrollo de aplicaciones para dicho sistema

operativo. Al existir tantas versiones en uso del sistema operativo (se pueden considerar hasta tres o cuatro versiones en uso del sistema operativo, con un porcentaje de uso por encima del 10%) es más complicado desarrollar aplicaciones que aprovechen al máximo los recursos del sistema, ya que se deben desarrollar aplicaciones más generalizadas para abarcar más versiones del sistema operativo.

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.1%
4.1.x	Jelly Bean	16	14.7%
4.2.x		17	17.5%
4.3		18	5.2%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	11.6%
5.1		22	0.8%

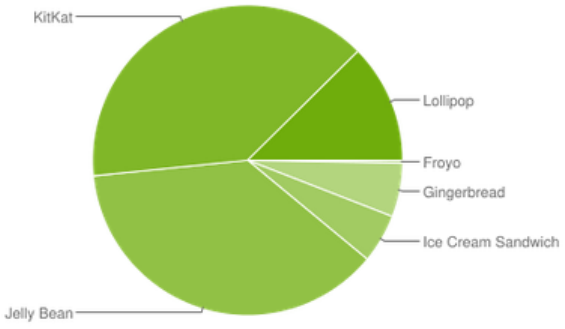

Ilustración 2.11. Gráfico uso versiones Android extraído de la web *http://developer.android.com/about/dashboards/index.html*, publicado en junio de 2015 y consultado en julio de 2015.

Para continuar se va a exponer el gráfico de uso de las diferentes versiones de iOS, para el cual, a diferencia de Android, no existe ese nivel de segmentación que se comentaba anteriormente. Como se puede apreciar en el gráfico, la última versión del sistema operativo, iOS8, ya se encuentra instalada en más de la mitad de los dispositivos (52%), practicamente la otra mitad (43%) utilizan iOS7, mientras que únicamente el 5% hace uso de una versión inferior. Esto es una gran ventaja de este sistema operativo sobre Android, ya que a la hora de diseñar una aplicación, esta se desarrollada orientada únicamente a las versiones que emplean los usuarios, que en este caso son practicamente solo dos.

Ilustración 2.12. Gráfico uso versiones iOS

Finalmente se va a repasar el uso de las diferentes versiones de Windows Phone. Como se aprecia en el gráfico, solo existen cuatro versiones del sistema operativo, y una amplia mayoría emplea la versión 8.1 del mismo, seguidos del 16% que usan Windows Phone 8.0, después aparecen los usuarios de alguna de las versiones de Windows Phone 7 con un 7,5% de cuota de mercado y para finalizar los usuarios de Windows 10 (la versión más actual) con un porcentaje de uso del 2,3%. Se espera un incremento de uso de este sistema operativo con el lanzamiento de su versión 10, ya que tratará de aunar todos los dispositivos que se emplean en la vida cotidiana (al igual que ya hace Apple con su iOS).

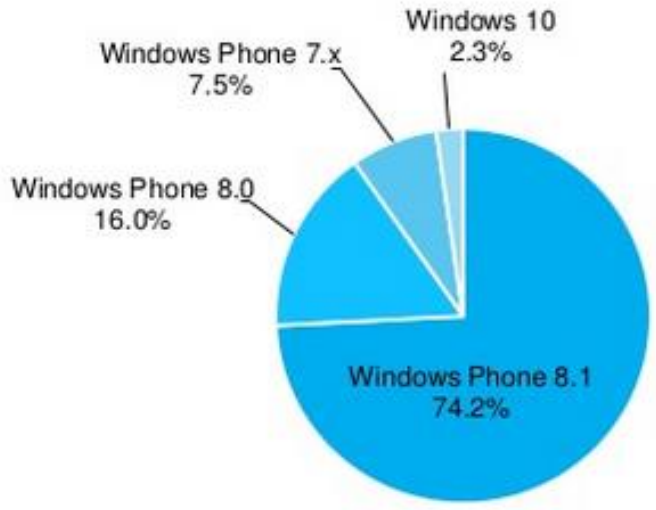

Ilustración 2.13. Gráfico uso versiones Windows Phone

2.2. Decisión

Para concluir este apartado de estudio previo al análisis, desarrollo y fase de prueba de la aplicación se va a tomar una decisión a cerca de sobre que plataforma se desenvolverá la aplicación móvil.

Repasando todo lo estudiado en este *Capítulo 2*, se puede afirmar con total rotundidad que Android es el sistema operativo móvil más empleado en la actualidad con mucha difirencia. Y, si no se produce ninguna catástrofe durante los años venideros, este lo seguirá siendo por unos cuantos años más. Además, para mi, es una de las plataformas sobre las que es más sencillo desarrollar una aplicación, puesto que dispongo de un dispositivo Android y conozco su funcionamiento. También la programación para Android está basada en Java, lenguaje de programación que sí que se ha tratado durante el desarrollo del grado, por lo que es otro punto a favor.

También se debe decidir sobre cuál de las versiones existentes de Android se desarrollará la aplicación. Para cada aplicación Android se debe definir una versión objetivo (target) y una versión mínima sobre las que desarrollar. Esto funciona de la siguiente manera: Google dispone de unas librerías de compatibilidad, de manera que si se implementa algo disponible solo a partir de la versión 5 de Android, esta característica se “autoimplementará” en las versiones anteriores mediante estas librerías, asegurando así la retrocompatibilidad. Por lo que la aplicación se desarrollará sobre la última versión dispoible (5.1, API 22 en este caso) teniendo en cuenta la versión mínima, ya que hay elementos que las librerías de compatibilidad no implementan. Por lo tanto, escogeré la versión mínima basándome en el diagrama expuesto en la *Ilustración 2.11*, *Sección 2.1.3.* En este gráfico se puede apreciar cuál es la cuota de mercado de cada una de las versiones de Android existentes. Lo más lógico es escoger la versión 4.0.3 (API 15), ya que con esta decisión se abarcaría el 94.1% de los usuarios, y las versiones anteriores presentan multitud de dificultades para implementar muchos elementos nuevos en Android, ya que el sistema cambió mucho de las versiones 2.x a las 4.x. Por lo tanto, se escogerá la versión 4.0.3 (API 15) como la versión mínima para el desarrollo.

Finalmente, se debe escoger sobre qué plataforma de desarrollo se comenzará a programar. Navegando por la web he encontrado muchos tutoriales explicando como desarrollar aplicaciones Android y la mayoría de los mismos emplean Eclipse con el plugin orientado a los desarrolladores de aplicaciones Android, sin embargo este IDE· ya no dispone de soporte por parte de Google. Por lo que me decantaré por el uso del IDE propio de Android, Android Studio.

3. ANÁLISIS

A partir de este capítulo comienza la parte técnica del proyecto.

Como se ha estudiado durante las diferentes asignaturas que componen el Grado en Ingeniería Informática, todo proyecto informático debe estar compuesto por cuatro fases principales: análisis del problema, diseño de la solución, implementación y prueba del producto. Como se describe en el título del capitulo, aquí se tratará el análisis del problema, desde un punto de vista conceptual, sin pensar en como soluionarlo o en las posibles tecnologías y componentes software específicos que se puedan utilizar en la construcción del producto final. Se puede decir que este apartado de la memoria tratará el “qué”, mientras que posteriormente, en los capítulos de diseño e implementación se tratará el “cómo”.

3.1. Planificación del proyecto

La planificación de un proyecto es una de las etapas más importantes, ya que de su mala o buena realización depende que el proyecto se pueda terminar con éxito a tiempo. La planificación consiste en la división del objetivo del proyecto en diversas tareas, a las que se les asignan recursos, como son tiempo, personal o dinero. El fin de la planificación es la óptima identificación y asignación de recursos a esas tareas. Para ello, se va a dividir el proceso de planificación en las siguientes etapas: división del trabajo, estimación de tiempo, identificación de hitos y planificación temporal.

3.1.1. División del trabajo

En este apartado de la memoria se identificarán las tareas que se tienen que desarrollar para alcanzar el objetivo principal del proyecto. Las actividades a realizar serán:

1. Búsqueda bibliográfica sobre dispositivos móviles, su software y desarrollo de aplicaciones orientados a los mismos.
2. Análisis del problema.
3. Diseño de la solución al problema.
4. Construcción de prototipo sobre papel.
5. Implementación.
6. Realización de pruebas al sistema.
7. Redacción de la memoria del proyecto.

Estas tareas serán realizadas siguiendo la metodología del Proceso Unificado, explicada con anterioridad en el *Capítulo 1*.

3.1.2. Estimación de tiempo

Una vez identificadas las tareas a realizar durante el proyecto, el siguiente paso será estimar el tiempo que va a llevar ejecutar cada una de ellas, de manera que se aproxime la duración del proyecto.

Actividad	Tiempo estimado
Búsqueda bibliográfica sobre dispositivos móviles, su software y desarrollo de aplicaciones orientados a los mismos.	7 días
Análisis del problema	3 días
Diseño de la solución al problema	6 días
Construcción de prototipo sobre papel.	2 días
Implementación	20 días
Realización de pruebas al sistema	2 días
Redacción de la memoria del proyecto	20 días
ESFUERZO TOTAL:	60 días (2 meses aprox)

Tabla 3.1. Tabla de estimación de tiempos

3.1.3. Identificación de hitos

Un hito es un “miniobjetivo” intermedio en el proceso de desarrollo de un proyecto y constituyen los pasos previos para la consecución de la meta final, es decir, son puntos intermedios en el camino de la realización del proyecto.

A partir de las tareas expuestas anteriormente se establecen los siguientes hitos:

- **Hito 1:** Conocimiento del mundo de los dispositivos móviles, tanto hardware como software existente, estudios de mercado y desarrollo de aplicaciones para los mismos. Este hito se alcanzará cuando se haya estudiado la bibliografía relacionada y se tome la decisión de sobre que software se desarrollará aplicación.
- **Hito 2:** Definición y análisis del problema. Se superará cuando se hayan especificado los requerimientos y casos de uso del sistema.
- **Hito 3:** Diseño de la solución. Se alcanzará este hito cuando se obtenga el diseño de clases definitivo.

- **Hito 4:** Construcción de prototipos. Se alcanzará este hito cuando se hayan desarrollado los storyboards representando las diferentes pantallas y situaciones posibles de la aplciación.
- **Hito 5:** Implementación y prueba del sistema. Se superará cuando se disponga de una versión definitiva del sistema.
- **Hito 6:** Redacción de la memoria. La superación de este hito se obtendrá cuando se haya completado la memoria del proyecto.

3.1.4. Planificación temporal

Para representar la planificacion temporal se va a utilizar un diagrama de Gantt. En este tipo de diagramas, cada nodo representa una actividad a realizar y su tamaño representa el tiempo que se empleará en ella. Los hitos se representan como rombos y, por medio de flechas se unen los nodos, expresando la dependencia de unas actividades respecto de otras ya realizadas. De este modo, se puede ver como el no cumplimiento de los tiempos estimados se traduce en un retraso en el resto de etapas del proyecto.

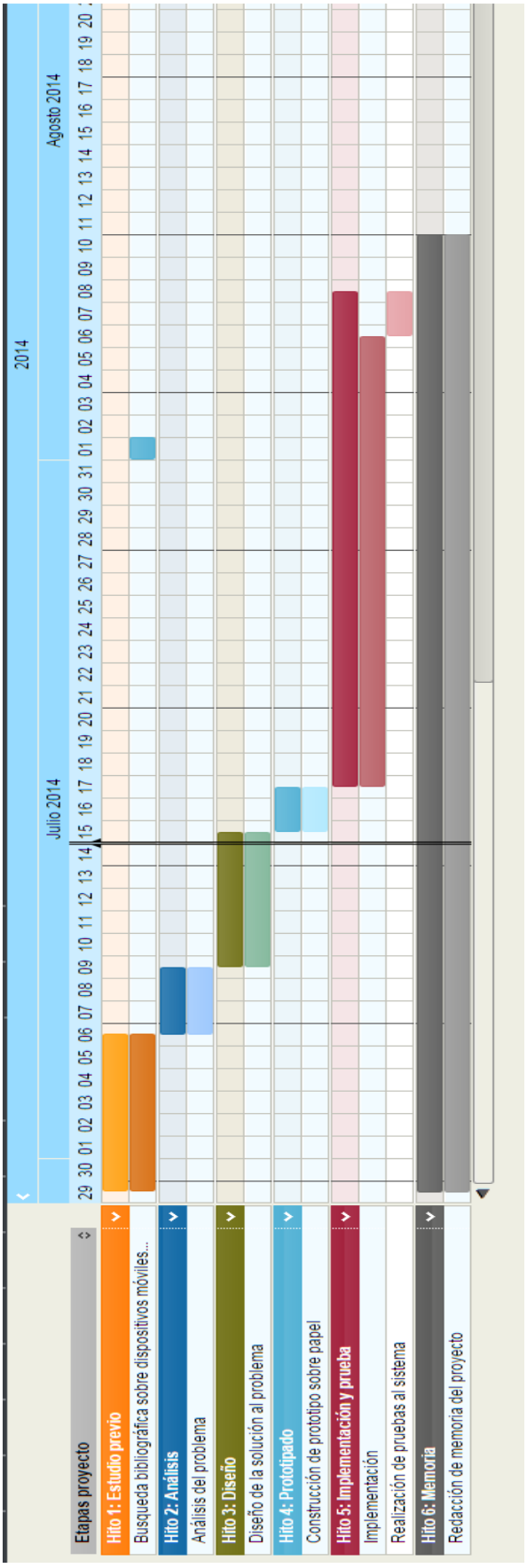

Ilustración 3.1. Diagrama de Gantt

3.2. Modelo de casos de uso

De forma general ya se definió cuál es el problema a resolver así como los objetivos a alcanzar durante el capítulo primero. Pero en este apartado se van a presentar, de una forma entendible, tanto para personas sin conocimientos en Ingenieria Informatica, como para aquellas que si los tienen, los requisitos que debe de cumplir el sistema final.

La metodología que se está siguiendo en el desarrollo del proyecto sugiere utilizar una herramienta llamada Casos de Uso para describir la funcionalidad del sistema. Estos, son fragmentos de funcionalidad de valor del sistema, es decir, cada uno representa una función del sistema que tiene un beneficio para quien lo utiliza, y a esos se les llama actores. Se podría pensar que los casos de uso se corresponden con los requisitos que debe de cumplir el sistema. Pero, aunque en el 95% de los casos es asi, existen ocasiones en que se incluyen requisitos que, aunque no son verdaderas funcionalidades del sistema, son caracteristicas tan importantes que se incluyen como requisitos funcionales.

En los siguientes apartados se listarán los requisitos funcionales y no funcionales, posteriormente se describirán los actores que intervendrán en los casos de uso y se mostrarán esquematicamente las relaciones entre estos y los actores.

3.2.1. Requisitos funcionales

Los requisitos funcionales se pueden definir como las funcionalidades que el cliente y el equipo de desarrollo acuerdan que debe tener el sistema final. En el desarrollo de software esta etapa es muy importante, porque si no se definen bien lo más probable es que el producto final no sea del agrado del cliente.

Al tratarse este de un Trabajo de Fin e Grado (TFG), realmente no existe un cliente con el que pactar los requisitos que debe de cumplir el sistema. Las funciones que tiene que tener el sistema las acordé con los directores del proyecto, y son las siguientes:

- Será posible consultar la ruta desde la ubicación del dispositivo hasta la ubicación del congreso.
- Se podrá consultar el programa del congreso.
- Se podrán consultar los diferentes articulos relativos al congreso.

- Será posible la activación de notificaciones para avisar al usuario de aquellas ponencias a las que este esté interesado en asistir.
- Se podrán compartir las diferentes experiencias del congreso a través de diferentes redes sociales.

3.2.2. Requisitos no funcionales

Los requisitos no funcionales son aquellos que especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento o facilidad de uso. Aunque no describan funcionalidades que tiene que tener el sistema son igual de importantes que los requisitos funcionales, y su incumplimiento puede provocar la no aceptación del producto por parte del cliente.

Los requisitos no funcionales que debe cumplir la aplicación software que surja de este proyecto son:

1. **Requisitos de usabilidad:**
    - **Interfaz de usuario familiar:** El conocimiento que los usuarios tengan de haber utilizado otras aplicaciones android o aplicaciones móviles debe ser suficiente para poder usar dicha app.
    - **Facilidad de aprendizaje**: El tiempo de aprendizaje que necesita el usuario para poder emplear plenamente la aplicación debe ser mínimo.
    - **Tiempo de respuesta**: El tiempo que el usuario ha de esperar para obtener la información que desea encontrar en la aplicación debe ser mínimo.
2. **Accesibilidad:** Para acceder al sistema, el usuario debe disponer de un dispositivo móvil con Android 4.0.3 o superior.
3. **Disponibilidad:** Si el dispositivo móvil dispone de conexión a internet, el sistema debe poder utilizarse al completo.
4. **Extensible:** El sistema debe construirse de tal forma que la incorporación de nuevas funcionalidades no requiera esfuerzo.
5. **Licencias software:** Las imágenes y audios que se utilicen tienen que ser de licencias libres, para evitar posibles problemas con los formatos propietarios en el futuro.

6. **Requisitos económicos:** EL proyecto no cuenta con ningún tipo de financiación, por lo que todos los componentes software que se utilicen deben de suponer un gasto cero para el proyecto.
7. **Recursos informáticos:** Debido a las limitaciones económicas del proyecto, el equipo informático que se utilizará para el desarrollo del proyecto será el ordenador portátil con el que habitualmente trabajo:
    - **Hardware**: Acer Aspire 5750G, con CPU Intel Core i5 2410M 2.3GHz y RAM de 8GB.
    - **Software**: Sistema Operativo Windows 7, entorno de programación Android Studio, editor de diagramas UML Visual Paradigm, editor de texto Microsoft Word, editor de imágenes Gimp.

3.2.3. Actores del Sistema

Una vez conocidos cuales son los requisitos del sistema, en este y en los dos apartados siguientes se describirán de una forma más gráfica. En este concretamente, se definen los actores, es decir, los elementos que interaccionarán con el sistema, siendo los siguientes:

- **Actor 1: Usuario**. Cualquier persona que descargue la aplicación de la Play Store de la que todos los dispositivos Android disponen. Este usuario podría ser, por ejemplo, un congresista, un ponente o cualquier interesado en el tema que trata el congreso (Procesamiento del Lenguaje Natural), aunque este no pueda asistir al congreso.

3.2.4. Diagrama de casos de uso

A continuación se presenta de forma esquemática la relación entre los actores y los casos de uso del sistema de una manera general y sencilla.

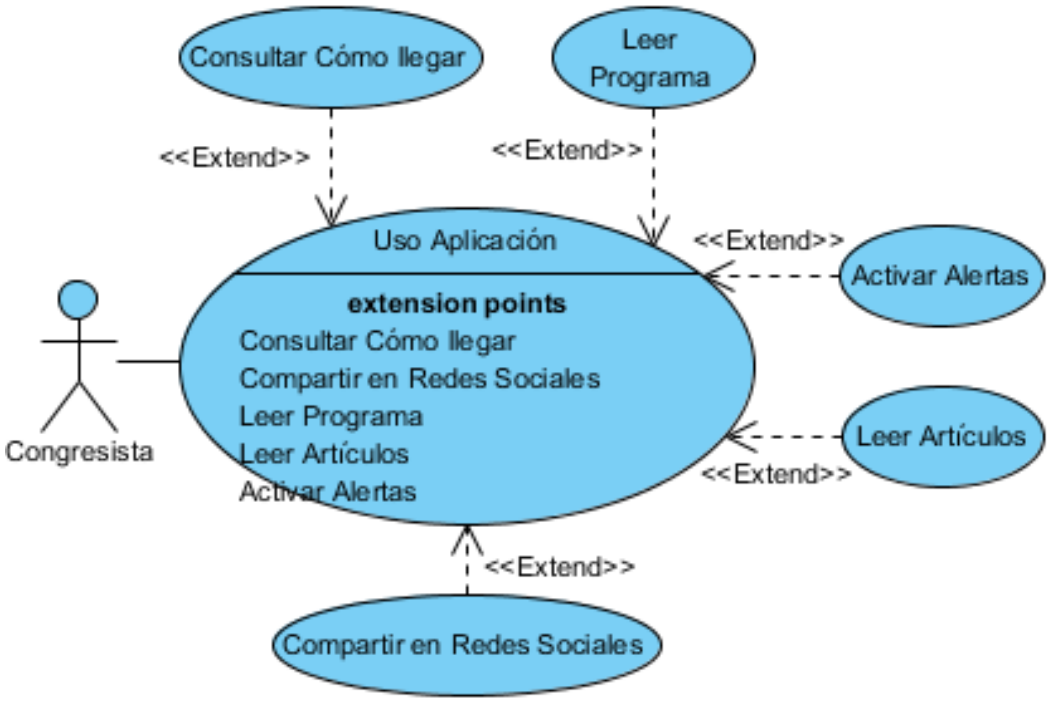

Ilustración 3.2. Diagrama de caso de uso general "Uso APP"

Como se puede ver en el diagrama, el sistema dispone de un caso de uso principal, **Uso Aplicación**. Este extiende las diferentes funcionalidades de la aplicación: **Leer Programa**, **Activar Alertas**, **Leer Artículos**, **Compartir en Redes Sociales** y **Consultar Cómo llegar**.

El siguiente paso será desarrollar cada uno de los casos de uso que extienden al caso de uso principal y, posteriormente, en la próxima subsección se describirán las diferentes narrativas de cada uno de los casos de uso.

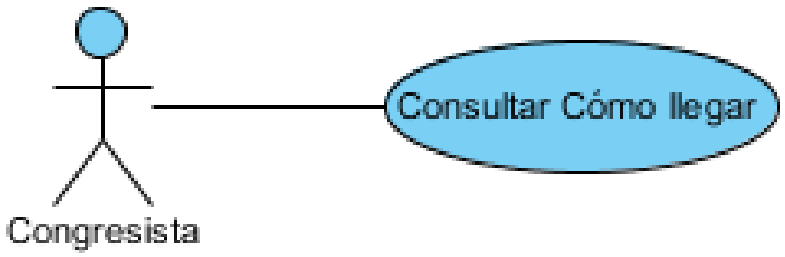

Ilustración 3.3. Diagrama de caso de uso específico Consultar Cómo llegar

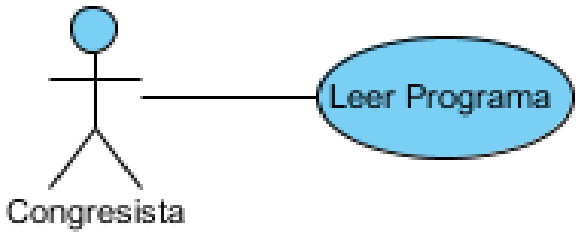

Ilustración 3.4. Diagrama de caso de uso específico Leer Programa

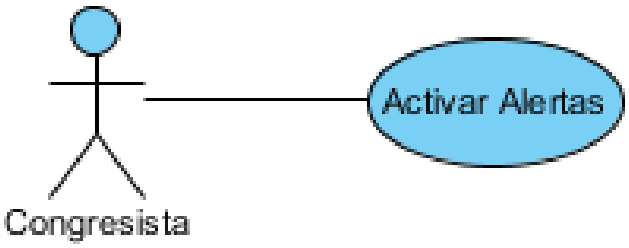

Ilustración 3.5. Diagrama de caso de uso específico Activar Alertas

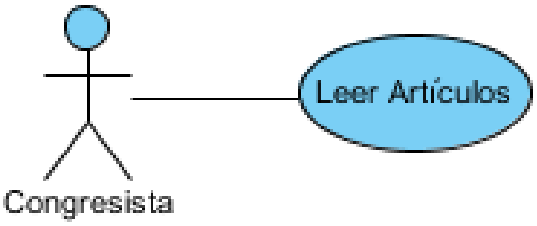

Ilustración 3.6. Diagrama de caso de uso específico Leer Artículos

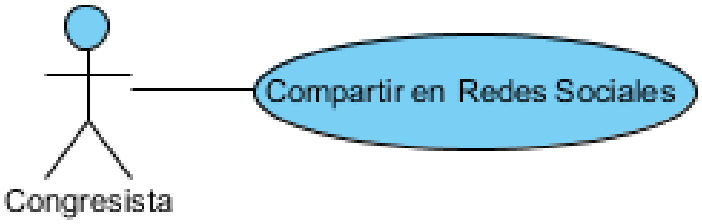

Ilustración 3.7. Diagrama de caso de uso específico Compartir en Redes Sociales

3.2.5. Narrativas de casos de uso

En esta subsección se procederá a describir cada uno de los casos de uso expuestos anteriormente mediante lenguaje natural. Para ello se empleará una plantilla que incluye cada uno de los aspectos necesarios para la completa

descripción de los mismos. A continuación se enumeran cada uno de los casos de uso importantes con su respectiva narrativa:

- **Uso Aplicación**

Actor Principal	Congresista
Actores Apoyo	Aplicación SEPLN 2015
Precondiciones	Asistencia al congreso
Condición salida	El congresista accede a todos los servicios que ofrece la app
Flujo básico de eventos	
1	El congresista descarga la aplicación del congreso
2	El congresista accede a la aplicación
3	El congresista hace uso de la app: leer artículos, programa, accede a los puntos de interés del congreso, activa alertas y comparte su experiencia en las redes sociales (casos de uso descritos a continuación).

Tabla 3.2. Narrativa del caso de uso Aplicación

- **Consultar Cómo llegar**

Actor Principal	Congresista
Actores de Apoyo	Aplicación SEPLN 2015
Precondiciones	Tener la aplicación instalada
Condición de salida	El congresista obtiene la ruta desde su ubicción hasta el punto de interés seleccionado
Flujo básico de eventos	
1	El congresista accede al mapa de la aplicación
2	La aplicación muestra el mapa con las diferentes ubicaciones (puntos de interés)
3	El congresista escoge una de las ubicaciones que aparecen (puntos de interés)
4	El congresista escoge ver la ruta (¿Cómo llegar?)
5	El congresista ve la ruta desde la aplicación de mapas correspondiente en su dispositivo
Flujos alternativos de eventos	
4a	El congresista decide que no quiere ver la ruta
1	El congresista escoge la opción de volver al mapa
2	Vuelve al paso 2 del flujo básico

Tabla 3.3. Narrativa del caso de uso Cómo llegar

- **Leer Programa**

Actor Principal	Congresista
Actores de Apoyo	Aplicación SEPLN 2015
Precondiciones	Tener la aplicación instalada
Condición de salida	El congresista accede al programa completo del congreso
Flujo básico de eventos	
1	El congresista accede al programa en la aplicación
2	La aplicación muestra el programa
3	El congresista navega por el programa del congreso

Tabla 3.4. Narrativa del caso de uso Leer Programa

- **Activar Alertas**

Actor Principal	Congresista
Actores de Apoyo	Aplicación SEPLN 2015
Precondiciones	Tener la aplicación instalada
Condición de salida	El congresista gestiona su tiempo activando alertas de aquellos eventos a los que desee asistir
Flujo básico de eventos	
1	El congresista se encuentra en el programa de la aplicación (caso de uso anterior *Leer programa*)
2	El congresista escoge un evento al que desea asistir para activar alerta
3	El sistema activa la alerta
Flujos alternativos de eventos	
3a	La alerta ya está activa y esta no se puede activar
1	Vuelve al paso 1 del flujo básico
3b	Es imposible activar la alerta, evento pasado
1	Vuelve al paso 1 del flujo básico

Tabla 3.5. Narrativa del caso de uso Activar Alertas

- **Leer Artículos**

Actor Principal	Congresista
Actores de Apoyo	Aplicación SEPLN 2015
Precondiciones	Tener la aplicación instalada
Condición de salida	El congresista accede a los artículos relacionados con el congreso
Flujo básico de eventos	
1	El congresista accede a los artículos en la aplicación
2	La aplicación la lista de artículos
3	El congresista escoge un artículo que desea ver
4	El congresista descarga dicho artículo
5	El congresista lee el artículo descargado con la aplicación correspondiente en su dispositivo

Tabla 3.6. Narrativa del caso de uso Leer Artículos

- **Compartir en Redes Sociales**

Actor Principal	Congresista
Actores de Apoyo	Aplicación SEPLN 2015
Precondiciones	Tener la aplicación instalada
Condición de salida	El congresista comparte su experiencia a través de las redes sociales
Flujo básico de eventos	
1	El congresista acciona alguno de los botones *Compartir* disponibles en las diferentes pantallas de la aplicación
2	El sistema le ofrece las posibles plataformas sobre las que desea compartir
3	El congresista escoge una de las plataformas
4	El sistema abre dicha plataforma para que el usuario comparta su experiencia

Tabla 3.7. Narrativa del caso de uso Compartir en Redes Sociales

3.3. Modelo del dominio

Una vez esquematizado y desglosado el problema con el diseño de los casos de uso anteriores, se procederá a la realización del modelo de dominio.

Esto es un modelo conceptual en el que se incluyen todos los temas relacionados con el problema específico. En él se describen las distintas entidades, sus atributos, papeles y relaciones, además de las restricciones que rigen el dominio del problema. Así se obtendrá el vocabulario y los conceptos clave.

Se puede decir que este es un paso previo al diseño del diagrama de clases, el cual será llevado a cabo posteriormente en el apartado de diseño de la aplicación.

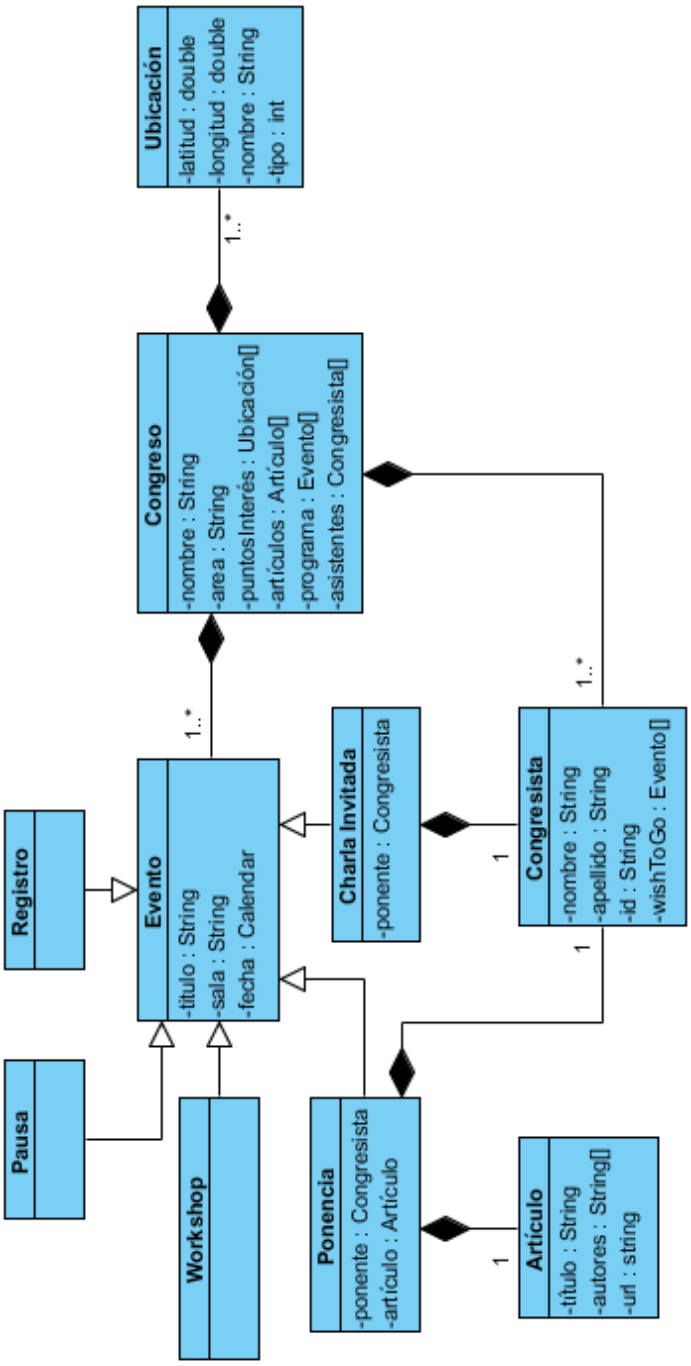

Ilustración 3.8. Modelo de Dominio de la aplicación

Como se puede apreciar en el diagrama, existen cinco conceptos diferentes:

- **Congreso**: es el concepto principal, junto a **Congresista**. Este es único y alrededor suyo se desarrollan el resto de conceptos. Está compuesto por un conjunto de asistentes (**Congresistas**), un conjunto de puntos de interés (**Ubicaciónes**) y varios **Eventos** de diferentes tipos.

- **Congresista:** representa a los asistentes al Congreso. Además algunos de estos congresistas serán los encargados de llevar a cabo las **Ponencias** y **Charlas invitadas** (tipos de **Evento**). Posteriormente, veremos en que se diferencian cada uno de los eventos.
- **Ubicación:** es un punto en el mapa definido por su latitud y longitud, un nombre y un tipo (lugar del congreso, monumento, restaurante, etc.).
- **Artículo:** representa los artículos relacionados con el congreso.
- **Evento**: representa los diferentes acontecimientos del congreso, existen diferentes tipos de eventos: **Pausa**, simplemente es un descanso; **Registro**, también es simple y trivial; **Workshop**, es un taller llevado a cabo en el Congreso; **Ponencia**, es una charla con un **Artículo** asociado; y **Charla invitada**, como su propio nombre indica es una charla llevada a cabo por un invitado, sin un **Artículo** asociado concreto.

3.4. Diagrama de actividades

El diagrama de actividades o diagrama de flujo es la representación gráfica del flujo de trabajo paso a paso, de negocio y de operaciones del sistema[23]. Con este diagrama se conseguirá una idea general de la sucesión de acontecimientos del sistema que compone la aplicación a desarrollar.

En este diagrama se emplearán símbolos con significados definidos que representan los pasos que sigue el proceso a describir, y se representa el flujo de ejecución mediante flechas que conectan los puntos de inicio y de fin del proceso.

A continuación se modelará el diagrama de actividades que representa la participación en el Congreso.

[23] Definición obtenida de la web: https://es.wikipedia.org/wiki/Diagrama_de_flujo

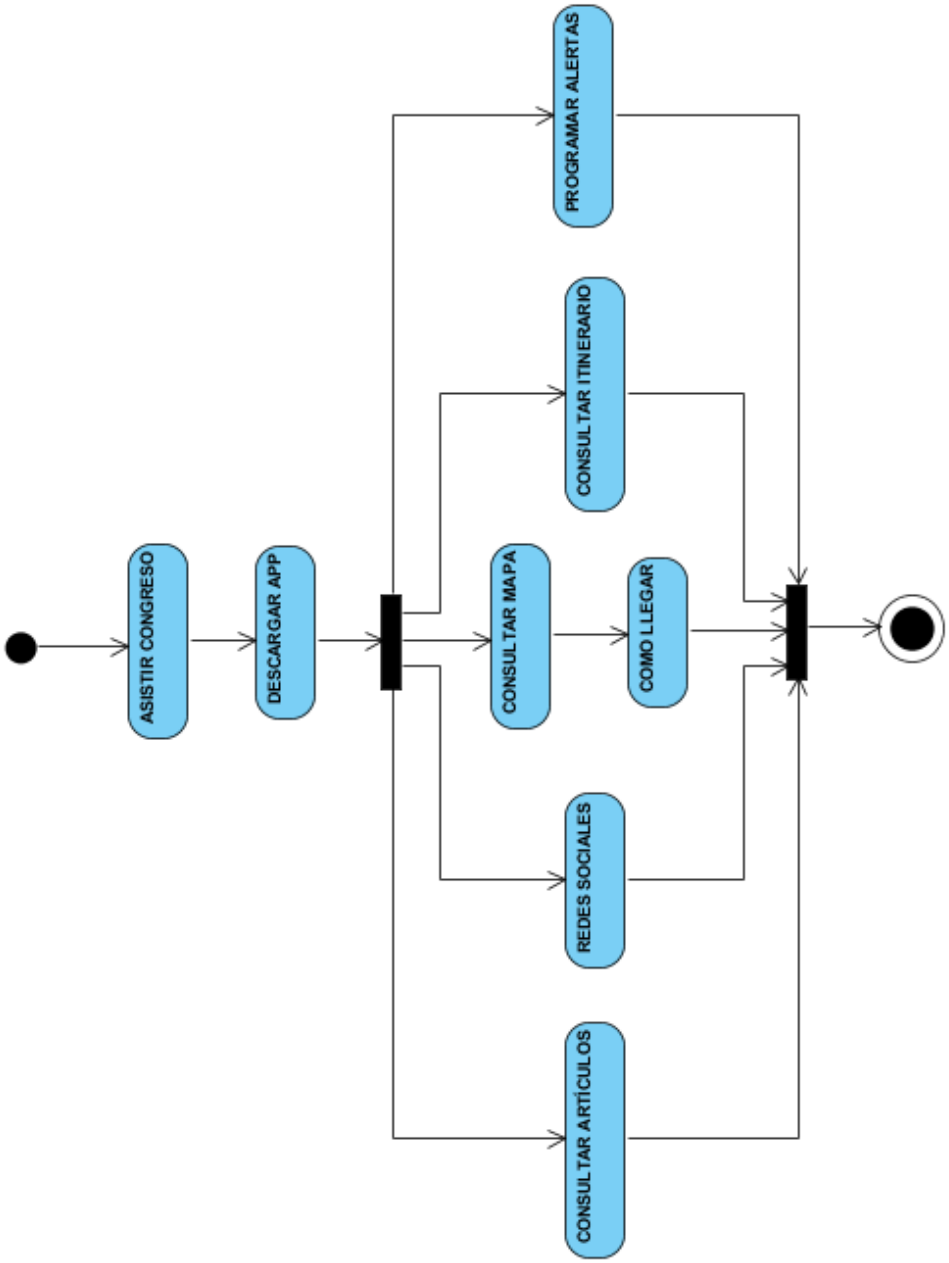

Ilustración 3.9. Diagrama de actividades

4. DISEÑO

Una vez completado el Análisis (definición del problema), es hora de darle una solución, siendo este el objetivo del Capítulo 4. Se comenzará con una descripción textual de la solución al problema, luego se continuará con una explicación de los componentes software que se utilizarán en la construcción del sistema. Una vez terminados estos dos apartados introductorios, se utilizarán distintos artefactos del Proceso Unificado para ir especificando de forma incremental la arquitectura y el diseño de la solución. Una vez presentados los distintos artefactos, el lector podra ver el diseño de la interfaz del usuario. El último apartado del capítulo corresponde con el diseño de clases definitivo de la solución propuesta, que constituirá la entrada a la siguiente fase en el proceso de desarrollo software, la implementación.

4.1. Descripción de la solución

Del análisis anterior, se puede concluir que lo que se debe desarrollar es una forma sencilla para que todo congresista tenga acceso a la información que necesite de manera previa al congreso, durante el mismo y de manera posterior si se desea, y todo esto de una manera sencilla.

Por ello, como indica el título del proyecto, se desarrollará una aplicación orientada a dispositivos móviles con tal propósito. Con esto se conseguirá que todo sea mucho más cómodo para el congresista medio. Podrá acceder a la información sobre el congreso incluso antes de su comienzo, ahorrándose asi el proceso de buscar información esencial previa como podría ser, por ejemplo, la forma de llegar al edificio del congreso. También podrá organizar su tiempo con una mayor antelación, haciendo así su trabajo más eficiente.

A continuación se procederá a describir los componentes software que serán empleados durante el diseño de la aplicación y la metodología que se seguirá durante el desarrollo:

4.1.1. Visual Paradigm

Visual Paradigm para UML es una herramienta CASE que soporta UML 2, SysML y el Modelo y Notación de Procesos de Negocio (BPMN en inglés), perteneciente al Object Management Group (OMG). Además de proveer herramientas para el modelado, también dispone de generación de informes y capacidad de generación de código. También puede generar diagramas a través del código.[24]

Se ha empleado esta herramienta para el diseño de cada uno de los diagramas necesarios para el desarrollo de la aplicación para el Congreso sobre el Procesamiento del Lenguaje Natural.

Decidí emplear esta herramienta y no otra ya que esta fue la usada durante el desarrollo del Grado. Concretamente, se empleó está aplicación para llevar a cabo las prácticas de la asignatura Fundamentos de Ingeniería del Software. Posteriormente, he empleado dicha aplicación para diseñar cada uno de los diagramas UML que he necesitado. Por lo que es una herramienta familiar para mí, y la considero la mejor opción para el diseño de dichos diagramas.

[24] Información obtenida de la web en inglés: https://en.wikipedia.org/wiki/Visual_Paradigm_for_UML

4.1.2. Metodología y lenguaje de programación

Antes de comenzar con el diseño propiamente dicho, se mencionará la metodología y el lenguaje de programación que se va a seguir, porque el diseño, aunque se intentará hacer lo más genérico posible, va a depender de esta elección. Como ya se mencionó en el *Capítulo 2*, la plataforma escogida a la que irá orientada la aplicación a desarrollar como objeto del TFG será Android. Una vez tomada esta decisión, la metodología y el lenguaje de programación no es algo que haya que decidir o escoger. Para la programación de aplicaciones para Android se sigue la metodología de programación Orientada a Objetos y con el lenguaje de programación Java.

Concretamente, la programación para dispositivos Android consta de dos partes: la parte lógica de la aplicación, para la cual se emplea la ya mencionada programación Orientada a Objetos con lenguaje Java, y la parte de diseño de la aplicación, un conjunto de ficheros .XML que definirán la distribución del contenido de las diferentes pantallas de nuestra aplicación.

Por lo tanto, la metodología a seguir será la siguiente: para cada una de las pantallas se diseñará el fichero .XML correspondiente y, posteriormente, se implementará parte lógica de la misma mediante lenguaje Java. La parte lógica consistirá en la carga de información desde una base de datos, cambiar contenido de manera dinámica, programación de alertas, carga de mapas mediante la API anteriormente descrita, etc.

4.2. Diseño de datos

Una vez introducidas cada una de las herramientas software que se van a emplear durante el diseño de la aplicación cometido de este TFG, se va a proceder a explicar la parte de los datos: cómo son los datos que tratará la aplicación, su procedencia, cómo se van a obtener, el uso que se les va a dar y su destino.

4.2.1. Definición de los datos de la aplicación

Para comenzar se van a definir qué datos debe manejar la aplicación. A partir del modelo del dominio expuesto en el *Capítulo 3. Análisis* se pueden apreciar una serie de entidades de manera muy clara: Congresista, Congreso, Evento (y todos sus tipos), Artículo y Ubicación.

A partir de los requisitos también expuestos en dicho capítulo, se puede afirmar que de estas entidades, la aplicación tratará una serie de objetos de tipo **Artículo**, una serie de objetos de tipo **Evento** y una serie de objetos de tipo **Ubicación**. Tanto el concepto **Congresista**, como el concepto **Congreso**, son empleados para modelar el sistema, pero estos no serán necesarios explícitamente en la aplicación. Ya que el **Congreso** es el ente alrededor del que se modelan el resto de entidades, por lo tanto este podría ser análogo a la aplicación, porque esta contiene toda la información a cerca de un único **Congreso**. Mientras que el concepto **Congresista** tampoco será necesario para implementar los requisitos funcionales expuestos.

Para continuar, se va a proceder al modelado de las tres tipos de datos que la aplicación va a manejar.

4.2.2. Modelado de los Artículos

Como ya se ha mencionado en el párrafo anterior, la aplicación dispondrá de una serie de **Artículos**, los cuales el usuario podrá consultar. Cada uno de estos **Artículos** corresponde con un **Evento** de tipo **Ponencia** del Congreso y se calcula que habrá un número reducido de **Artículos** (entre treinta y cuarenta) Cada dato de tipo **Artículo** debería disponer de un autor, un título y el artículo en sí. El autor y el titulo está claro que son datos simples, cadenas de caracteres, mientras que para el artículo en sí se deben estudiar las diferentes posibilidades de las que se dispone para almacenarlo.

1. Una opción sería almacenar cada uno de los artículos en formato .PDF e incluirlos en la aplicación, por lo que el peso de la misma aumentaría considerablemente, además de que para una posible actualización futura de la app todos los artículos deberían ser sustituidos por los nuevos. La ventaja de esta opción es que el acceso a los mismos sería más rápida y no se precisaría de conexión a Internet para leer artículos.
2. La segunda opción sería incluir una dirección web y que estos fueran descargados para su posterior consulta mediante alguna app específica instalada en el terminal. Esta opción tiene la ventaja de que hace a la aplicación más liviana (solo debería almacenar la dirección web de los artículos y no el artículo completo) aunque obliga al usuario de disponer de conexión a Internet para leer artículos.

Tras ver las dos posibilidades de las que se dispone, he decidido almacenar tan solo la dirección web del artículo, ya que la principal desventaja que esta opción conlleva es que es necesaria la conexión a Internet para leer los artículos, pero hoy en día prácticamente todo usuario de un dispositivo móvil dispone de una conexión a Internet en todo momento. De esta manera la aplicación será más liviana y accesible.

Además, los artículos concretos de dicha app serán extraídos de la web de la revista del SEPLN, en este caso la aplicación mostrará aquellos artículos que serán tratados en el Congreso. Tras toda esta información, aun quedaría por tomar la decisión más importante, y esta es de qué manera se van a obtener los datos por parte de la aplicación. Para esta decisión he considerado dos posibilidades:

1. La primera posibilidad sería la implementación de una base de datos simple, con una entidad Artículo de cuatro columnas: un id, una URL, el título y el autor del Artículo.
2. La segunda opción sería realizar un análisis de la web de la que se ha comentado que se extraerán los artículos, para que las direcciones de los artículos sean extraídas directamente y de manera automática de dicha web.

A primera vista, puede parecer que la segunda solución es la ideal, puesto que obtiene las direcciones web de manera automática y no sería necesario su almacenamiento en una base de datos para su posterior consulta. Sin embargo, la implementación de la segunda opción es bastante más compleja, además de suponer una gran desventaja a la hora de que la web podría cambiar, y por lo tanto la función que obtiene los artículos de manera automática también debería cambiar. También se debe considerar la cantidad de Artículos que se va a manejar, que como ya se expuso anteriormente estará entre treinta y cuarenta. Por lo que la obtención manual de las direcciones web será una tarea laboriosa, pero que no debería tomar más de 15 minutos o media hora. Por lo tanto decido implementar una base de datos para almacenar los Artículos, el diseño concreto, arquitectura, tecnología e implementación de la misma serán tratados en posteriores secciones de esta memoria.

4.2.3. Modelado de los Eventos

A priori, los Eventos son la entidad más difícil de modelar, ya que, como se expuso con anterioridad en el Modelo del Dominio en el *Capítulo 3: Análisis*, estos disponen de diferentes tipos. Analizando los diferentes tipos de Eventos, se puede ver que se trata más de algo conceptual que de una diferenciación real. Únicamente los eventos de tipo Ponencia y de tipo Charla invitada incluyen una diferenciación, y esta es que disponen de uno o varios Congresistas que llevan a cabo el Evento. Además las Ponencias dispondrán de un Artículo asociado.

Por lo tanto, los Eventos se modelarán de manera parecida a los Artículos, ya que son datos similares. Un Evento es un dato compuesto por varios más simples: un título, una duración, un día, un mes, un año, una hora, unos minutos, una posible notificación activada (en forma de entero representando los minutos de antelación con los que se notificará), una sala y, de manera opcional se incluirán, un artículo asociado y uno o varios congresistas.

En el caso de esta aplicación concreta, los Eventos serán consultados en la web del congreso del SEPLN 2015 en Alicante (*http://gplsi.dlsi.ua.es/sepln15/*). Además se tomará la opción de almacenarlos en la base de datos y no obtenerlos directamente de la web del congreso (la justificación de esta decisión es análoga a la de los Artículos). El diseño, arquitectura, tecnología e implementación de la base de datos serán tratados posteriormente.

4.2.4. Modelado de las Ubicaciones

Estas constan básicamente de una latitud, una longitud, un nombre y un tipo (indicando si esta es un monumento, un restaurante o es la localización del congreso).

En el caso de esta aplicación, la Ubicación principal del congreso será consultada en la web del congreso del SEPLN 2015, en Alicante (*http://gplsi.dlsi.ua.es/sepln15/*), existirá un tipo de punto de interés que representará algunos monumentos de la ciudad de Alicante de mi propia elección y el otro tipo de puntos de interés serán aquellos en los que se

desarrollan las diferentes actividades culturales del congreso, consultados en el programa del congreso (el cual aparece en la web del congreso también).

Para almacenar estas ubicaciones se diseñará una tabla con los campos correspondientes. Posteriormente se mostrará el diseño de la base de datos, además de definir la tecnología a emplear y el modo en que se implementará.

4.2.5. Diseño de la base de datos

Tras las decisiones estudiadas, justificadas y tomadas anteriormente, para completar este diseño de los datos solo faltaría definir la base de datos: su diseño, tecnología que empleará, etc.

El primer paso será definir qué es una base de datos y decidir qué tipo de base de datos se va a implementar en la aplicación objeto de este proyecto (base de datos relacional o base de datos distribuida).

Se le llama base de datos a los bancos de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto.[25] Existen diferentes tipos de bases de datos. Para mi base de datos concreta consideraré tres posibilidades:

1. **Base de datos relacional no distribuida**[26]: es un tipo de base de datos que cumple con el modelo relacional, el modelo más utilizado actualmente para implementar bases de datos ya planificadas. Permite establecer interconexiones o relaciones entre los datos (guardados en tablas), y a través de dichas conexiones relacionar los datos de dos tablas. Se van a exponer a continuación una serie de ventajas y desventajas de dichas bases de datos:
    - **Ventajas**: provee herramientas que garantizan evitar la duplicidad de registros, garantiza la integridad referencial y favorece la normalización por ser más comprensible y aplicable.
    - **Desventajas**: presentan deficiencias con datos gráficos y no se manipulan de forma adecuada los bloques de texto como tipo de dato.

[25] Definición obtenida de la web: https://es.wikipedia.org/wiki/Base_de_datos
[26] Información extraída de la web: https://es.wikipedia.org/wiki/Base_de_datos_relacional

2. **Base de datos relacional distribuida**[27]: es un conjunto de múltiples bases de datos lógicamente relacionadas las cuales se encuentran distribuidas en diferentes espacios lógicos e interconectados por una red de comunicaciones. Estas bases de datos permiten realizar operaciones tanto locales como distribuidas. Se van a exponer a continuación una serie de ventajas y desventajas de dichas bases de datos:
    - **Ventajas**: refleja una estructura organizacional, autonomía local, disponibilidad, rendimiento, economía (muchas computadoras pequeñas más barato que una muy potente) y modularidad.
    - **Desventajas**: complejidad, economía (mayor mano de obra), seguridad, integridad, falta de experiencia, carencia de estándares y diseño de la base datos complejo.
3. **Base de datos no relacional**: es un concepto de base de datos que surgió con la web y la necesidad de las exitosas "startups" de millones de usuarios. Con ello llegaron los evidentes problemas de alta escalabilidad.

    Los sistemas no relacionales intentan atacar este problema proponiendo una estructura de almacenamiento más versátil, aunque sea a costa de perder ciertas funcionalidades como las transacciones que engloban operaciones en más de una colección de datos, o la incapacidad de ejecutar el producto cartesiano de dos tablas teniendo que recurrir a la desnormalización de datos.

    Algunas implementaciones que se consideran no relacionales son: CouchDB, MongoDB, RavenDB, Neo4j, Cassandra, BigTable, Dynamo, Riak, Hadoop, [28]y otras muchas.

Tras la definición de dichos tipos de bases de datos, decido emplear la base de datos tipo relacional no distribuida, ya que el número de datos a almacenar será pequeño, además de ser datos de una complejidad bastante simple, y su implementación será más sencilla. Para el número de datos que se van a almacenar no merece la pena la implementación de una base de datos relacional distribuida o una base de datos no relacional.

[27] Información extraída de la web: https://es.wikipedia.org/wiki/Bases_de_datos_distribuidas
[28] Ejemplos consultados en la web: http://www.genbetadev.com/bases-de-datos/el-concepto-nosql-o-como-almacenar-tus-datos-en-una-base-de-datos-no-relacional

Una vez decidida la implementación de una base de datos relacional, será necesario su diseño. Para ello se va a dibujar el diagrama entidad-relación de dicha base de datos. Esto es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades[29]. Se muestra a continuación:

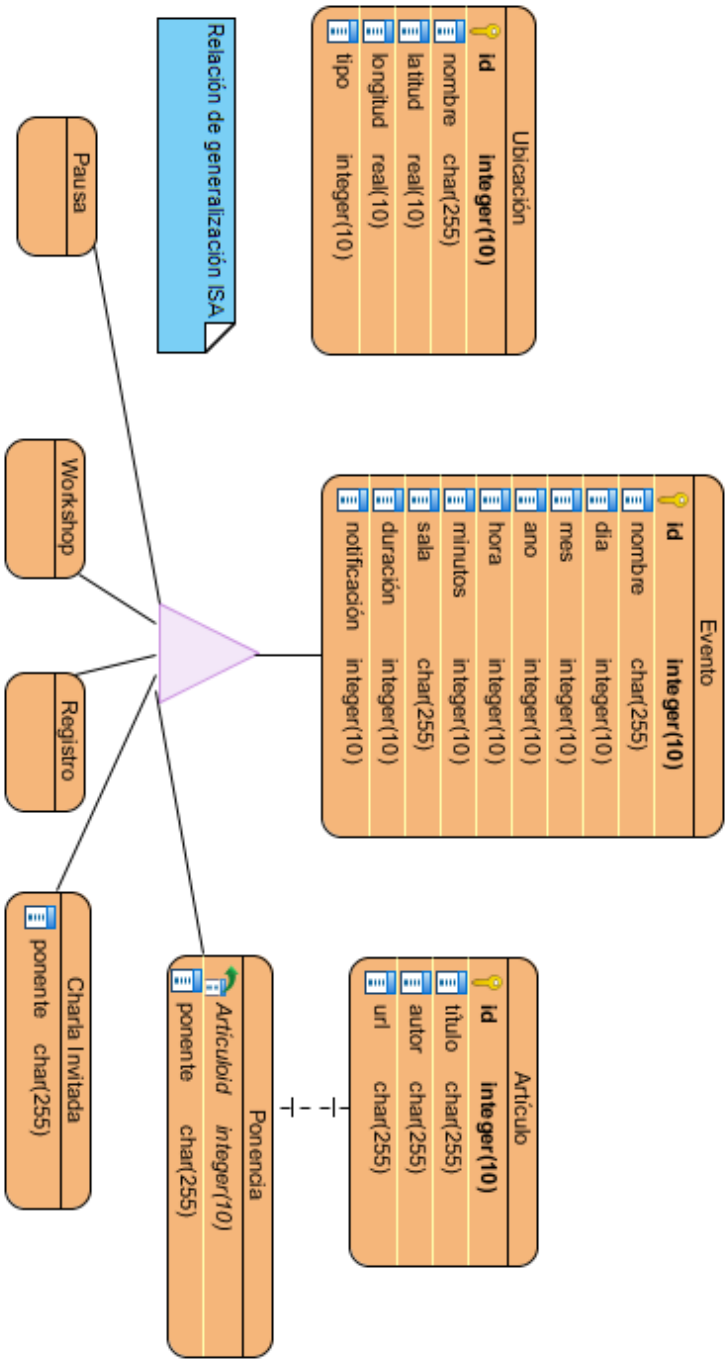

Ilustración 4.1. Diagrama entidad-relación

[29] Definición procedente de la web: https://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n

En el diagrama anterior se puede apreciar el diseño de la base de datos que posteriormente será implementada en la aplicación. Aquí podemos ver como este diseño consta de tres entidades principales, una entidad Ubicación, es independiente a las demás y dispone de un id (clave principal), un nombre, una latitud, una longitud y un tipo; se puede apreciar también una entidad Artículo la cual representa, como su nombre indica, cada uno de los Artículos a mostrar por la aplicación con su respectivo id (clave principal), título, autor y dirección URL; y, finalmente, otra entidad Evento, con todos los atributos correspondientes a un Evento (id, el cual será la clave principal de dicha entidad, título, sala, duración, día, mes, año, hora, minutos, notificación). A su vez, se puede apreciar como existe una relación de generalización ISA para Evento, de esta manera se modelarán los diferentes tipos de Evento existentes, además, en particular Ponencia y Charla invitada dispondrán de un atributo ponente (el/los que darán la charla o ponencia), además las Ponencias estarán asociadas con un artículo. Todos los tipos de Evento serán entidades hijo de la entidad padre Evento, obteniendo así cada uno de sus atributos y relaciones.

Por último, para terminar con esta sección del capítulo cuatro solo queda la elección de la tecnología para implementar la base de datos. Investigando un poco en la web encontramos que para la implementación de una base de datos de no mucho peso en una aplicación Android, la mejor opción es emplear SQLite. Esta tecnología se describirá en el *Capítulo 5: Implementación*. Con esto queda concluida la sección *Diseño de datos.* A continuación se procederá a diseñar los casos de usos correspondientes a la aplicación.

4.3. Diseño de casos de uso

Después de describir la arquitectura del sistema, el Proceso Unificado indica que el siguiente paso es el diseño de casos de uso. Consiste en la construcción iterativa del diagrama de clases general (incluyendo todos los casos de uso existentes) y de secuencia de cada caso de uso. Para alcanzar ese objetivo se aconseja ir refinando iterativamente ambos diagramas hasta alcanzar la versión definitiva para cada caso de uso.

4.3.1. Diagrama de clases

Este sistema cuenta con diferentes casos de uso sencillos, diseñados en el Capítulo anterior (Análisis). A continuación se muestra el diagrama de clases general que aúna todos los casos de uso, el cual está disponible también en este enlace: *https://www.dropbox.com/s/tucclpjpt2zz33n/diagramaClases.PNG?dl=0*.

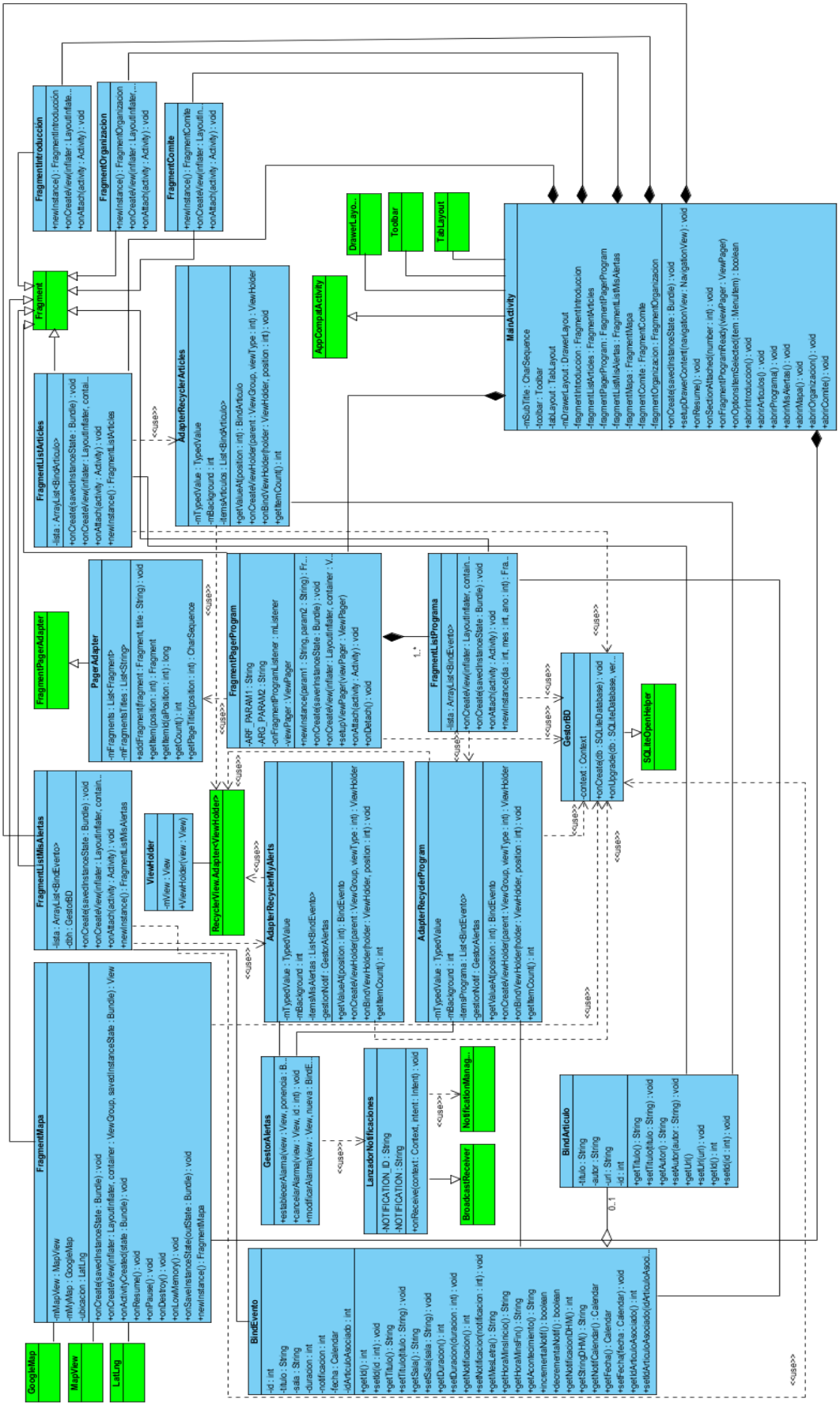

Ilustración 4.2. Diagrama de clases

A simple vista, el diagrama de clases general de la aplicación puede parecer algo engorroso, ya que este incluye todas y cada una de las relaciones del modelo lógico de la aplicación objeto de dicho proyecto. Sin embargo, a continuación se va a explicar el diseño del mismo, de manera que este sea completamente entendible.

Como se aprecia en el diagrama existen dos tipos de clases:

- **Verdes**: Aquellas definidas por Android, las cuales se emplean para extender de ellas obteniendo diferentes funcionalidades o se emplean a modo de atributo de alguna clase de la aplicación.
- **Azules**: Son las clases de la aplicación y que serán implementadas en el desarrollo de la aplicación.

Tras esta aclaración, se va a proceder a desglosar cada una de las clases principales e interacciones entre clases que forman parte del diagrama.

- **MainActivity**: es la encargada de controlar cada uno de los elementos en la aplicación. Esta implementa la clase de Android AppCompatActivity, por lo tanto se puede afirmar que es una actividad de la aplicación. También contiene la Toolbar (barra de acción superior presente en todo momento durante la ejecución de la app y que permite interactuar con la misma), el DrawerLayout (menú lateral desplegable desde la barra de acción para cambiar de pantalla), el TabLayout asociado a la barra de acción (esto es el conjunto de pestañas que aparecerá junto a la barra de acción y que solo será visible en la pantalla que mostrará el programa del congreso) y, por último, cada uno de los Fragment de la aplicación (se podría decir que un Fragment en Android es una porción de pantalla, en este caso será toda la pantalla a excepción de la Toolbar y el TabLayout; por lo tanto cada Fragment representa una sección de la aplicación). Además esta clase dispone de las funciones necesarias para hacer funcionar el DrawerLayout, lanzar cada uno de los Fragment e interactuar con la Toolbar.
- **GestorBD**: es la clase encargada de toda la funcionalidad correspondiente a la base de datos. Esta implementa la clase SQLiteOpenHelper. Lo que obliga a sobrecargar los métodos onCreate

(crea la base de datos) y onUpgrade (actualiza la base de datos en caso de que se desee cambiar la versión de la misma).

- **GestorAlertas**: es la clase encargada de la gestión de las alertas (como su propio nombre indica). Implementa los métodos para activar una alerta, cancelar una alerta o modificar una alerta. Para la implementación de estos métodos se hará uso de una clase auxiliar: **LanzadorDeNotificaciones**, encargada de implementar el BroadcastReceiver. Esto, explicado a grosso modo, es un "escuchador" el cual estará "escuchando" al sistema para lanzar la alerta cuando la fecha y hora sea la correspondiente a la establecida.
- **FragmentIntroducción, FragmentOrganización y FragmentComité**: estas tres clases son análogas, por lo que explicaré su uso en un mismo apartado. Cada una de estas clases implementa un Fragment de la aplicación, de ahí su extensión de la clase Fragment. Este será un Fragment muy simple en el que se mostrará un texto y unas imágenes predefinidas en la parte de la vista de la aplicación. El método onCreate se encargará de inflar el Layout correspondiente al Fragment (esto no es otra cosa que crear lo que se verá por pantalla al seleccionar este Fragment). El método newInstance(), será el empleado por la MainActivity para lanzar el Fragment. Estos dos métodos son comunes a todos los Fragment y su función es la misma para cada uno.
- **FragmentListArticles y FragmentListMisAlertas**: estos Fragment representan respectivamente la lista de Artículos y la lista de Alertas activas que se mostrarán en la aplicación. Estas clases son muy similares a las descritas anteriormente, aunque con la particularidad de que poseen una lista de los elementos que van a mostrar (FragmentListArticles dispone de una lista de **bindArtículo** y FragmentListMisAlertas dispone de una lista de **bindEvento**). En el método onCreate de cada una de las clases se accederá a la base de datos, haciendo uso de la clase anteriormente descrita **GestorBD**, para cargar las respectivas listas. Posteriormente, en dicho método se pasará la lista ya cargada a la pantalla del dispositivo. Para este último paso ambas clases hacen uso de su adaptador correspondiente (**AdapterRecyclerArticles** y

AdapterRecyclerMyAlerts). Un adaptador es una clase que se encarga de inflar el layout correspondiente a cada ítem de una lista, cambiando la información para cada ítem si fuera necesario. En este caso se coge la información de la lista precargada correspondiente, de manera que cada ítem de la lista reflejará la información de una fila de la base de datos.

- **FragmentPagerProgram**: este Fragment es uno de los más complejos de la aplicación. Implementa un ViewPager, un paginador de vistas. Esto junto con el TabLayout dará como resultado un conjunto de Fragments de tipo **FragmentListPrograma** asociados a un conjunto de pestañas que aparecerán debajo de la Toolbar principal. Esta clase debe crear instancias a tantos **FragmentListPrograma** (clase cuya funcionalidad es análoga a las explicadas en el apartado anterior, FragmentListArticles y FragmentListMisAlertas) como días tenga el congreso, ya que cada uno de estos Fragments mostrará la lista de eventos correspondiente a un día del congreso. Para esto se accede a la base de datos haciendo uso de la clase GestorBD descrita anteriormente.
- **FragmentMapa**: este Fragment es diferente a los anteriores, aunque dispone de las mismas funciones para su manejo a parte de algunas más para el uso de la API de Google Maps. Esta clase dispone de un MapView, el cual enlaza la vista del mapa con un objeto GoogleMap. El objeto GoogleMap será el usado para realizar las diferentes operaciones en la creación del Fragment. Para la creación de este Fragment se accederá a la base de datos mediante la clase GestorBD para colocar un marcador en el mapa correspondiente a cada una de las diferentes Ubicaciones que allí se encuentran. Este marcador será de diferente color dependiendo del tipo de punto de interés. Además se guardarán en un atributo de la clase de tipo LatLng la latitud y longitud de la ubicación del Congreso.

4.3.2. Diagramas de secuencia de interacción

El diagrama de clases es una visión estática del sistema, y es una entrada fundamental para la implementación de este. Pero en ocasiones, es necesario mostrar cómo interaccionan los objetos entre sí, para ello se utilizan los Diagramas de Secuencia.

El diagrama de secuencia es un tipo de diagrama usado para modelar interacción entre objetos en un sistema según UML. Muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso:

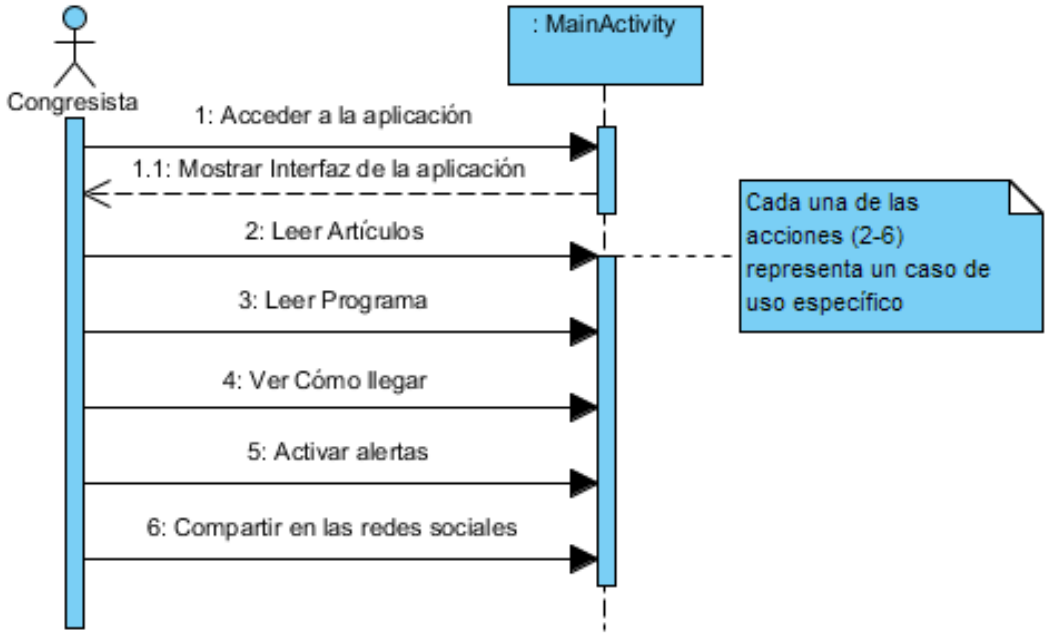

Ilustración 4.3. Diagrama de secuencia caso de uso general

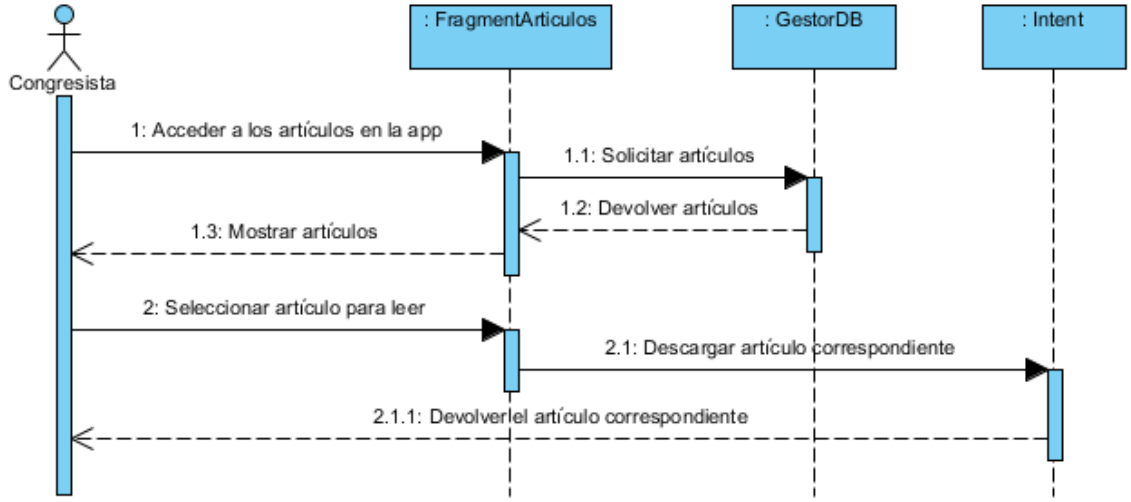

Ilustración 4.4. Diagrama de secuencia del caso de uso Leer Artículo

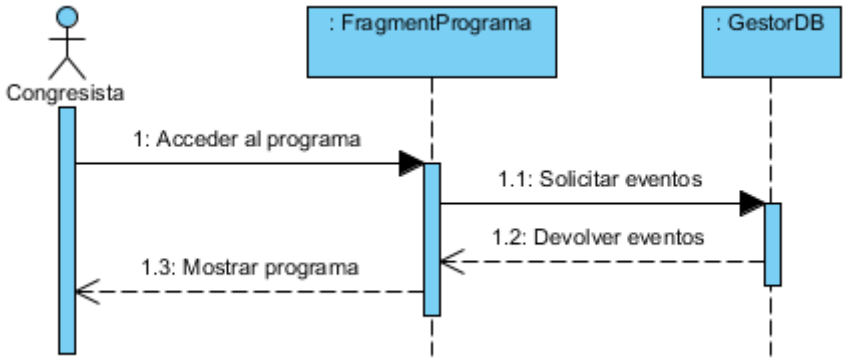

Ilustración 4.5. Diagrama de secuencia del caso de uso Leer Programa

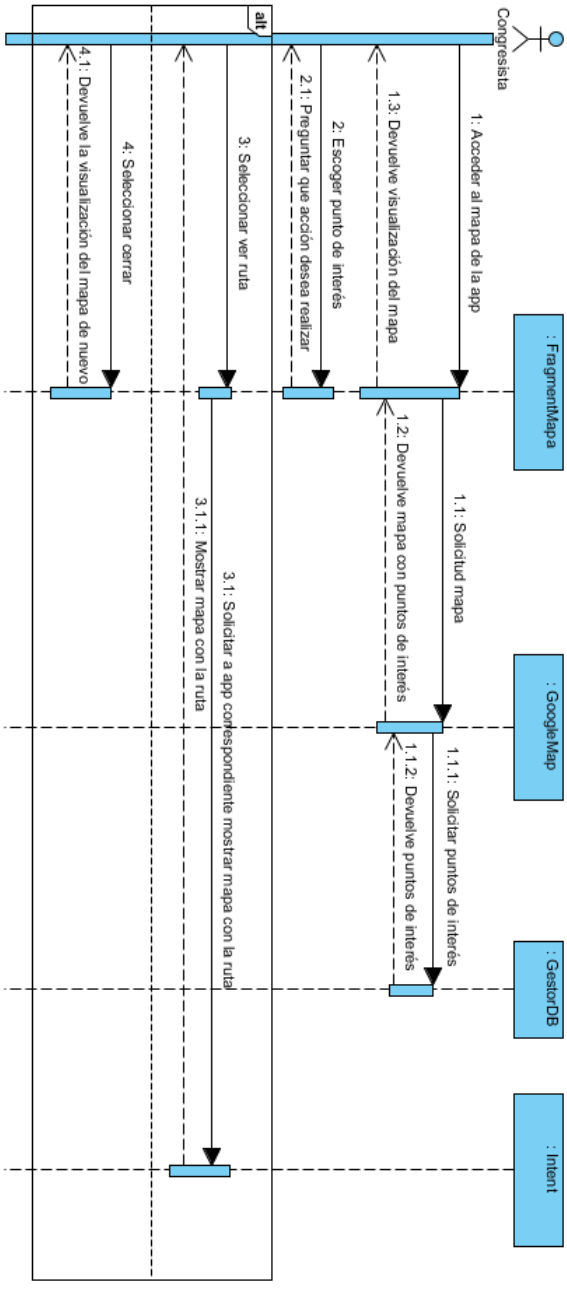

Ilustración 4.6. Diagrama de secuencia del caso de uso Consultar Cómo llegar

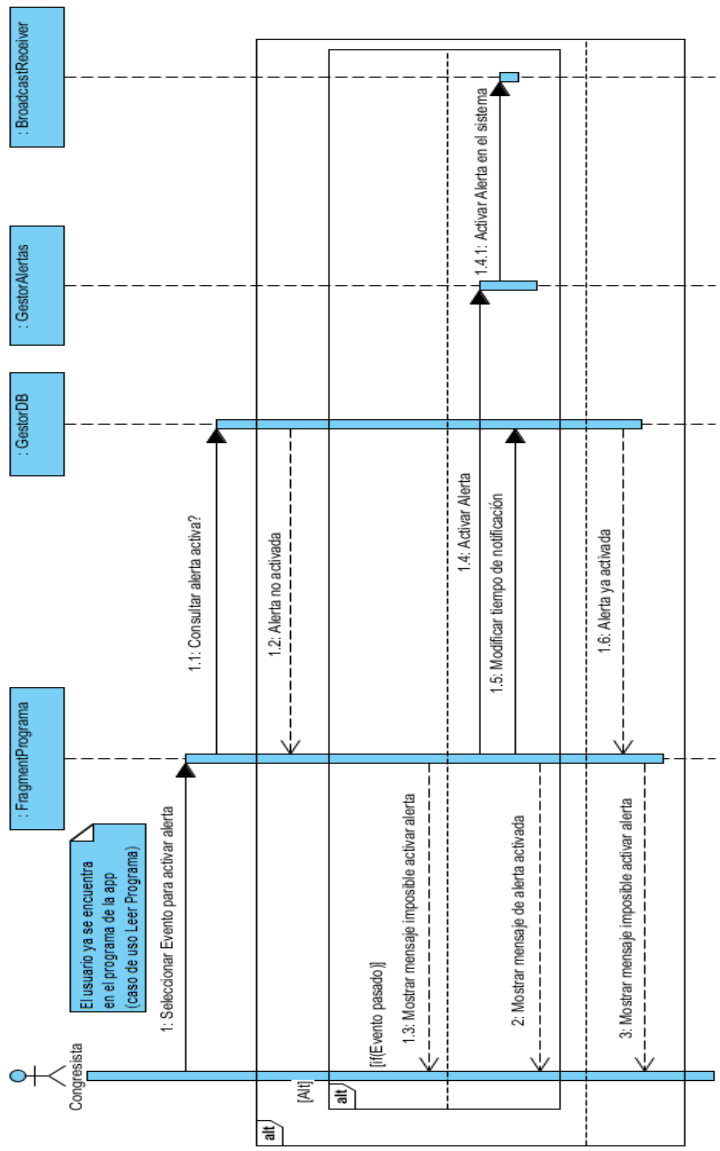

Ilustración 4.7. Diagrama de secuencia del caso de uso Activar Alertas

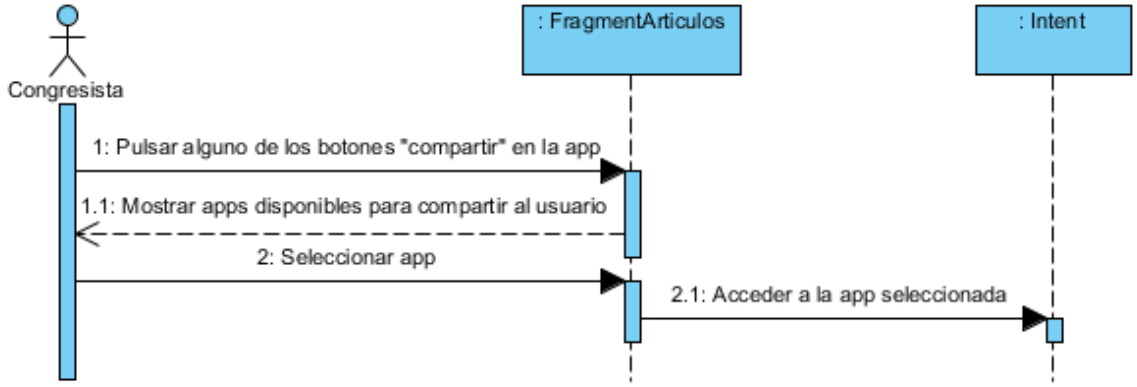

Ilustración 4.8. Diagrama de secuencia del caso de uso Activar Alertas

4.4. Diseño de la interfaz de usuario

Hasta ahora, el diseño de la solución se ha centrado en la parte del sistema, lo que no ve el usuario, lo que para los ingenieros es lo más importante, es decir, el modelo de la aplicación. En este apartado se va a diseñar lo más importante para los usuarios, lo que ellos ven, la interfaz de usuario. Esta es la parte que permite que los usuarios puedan beneficiarse de los servicios del sistema, por lo que debe de ser mucho más relevante para los ingenieros de lo que es, ya que se puede haber construido el sistema más eficiente, flexible, y en el que se han utilizado las mejores y más novedosas tecnologías, que si al usuario no le gusta, o no la entiende, no podrá alcanzar sus objetivos, por lo que el sistema no tendrá ninguna valía para él.

El diseño de la interfaz no solo estará compuesto del diseño de las pantallas, sino también por la definición de su estilo y por el diseño de los llamados storyboards o caminos de navegación.

4.4.1. Definición del estilo de la interfaz

Para que una interfaz de usuario tenga éxito, se entiende que tiene que satisfacer dos condiciones: ser usable y encandilar al usuario. La segunda condición se refiere el aspecto visual de la interfaz, el cual debe gustarle al usuario, transmitirle seguridad, tranquilidad, en definitiva, tiene que captar su atención.

Para el desarrollo de una aplicación Android, el Google dispone una serie de normas de estilo, las cuales son aconsejables satisfacer.

Durante el año 2014, Google lanzó el concepto Material Design, este es el sucesor al diseño Holo. Material Design recibe su nombre por estar basado en objetos materiales. Piezas colocadas en un espacio y con un tiempo determinado. Es un diseño donde la profundidad, las superficies, los bordes, las sombras y los colores juegan un papel principal. Precisamente, esta es una manera de aproximarse a la realidad, algo que en un mundo donde todo es táctil y virtual es difícil. Material Design se guía por la leyes físicas, las animaciones son lógicas, los objetos se superponen pero no pueden atravesarse, etc.

Para todo ello Android pone a disposición del desarrollador todas las facilidades posibles. Ya que las animaciones, los colores, y en cierta medida el diseño vienen predefinidos, sin cohibir la flexibilidad.

En el caso de mi aplicación se ha decidido implementar un diseño sencillo formado por una barra superior con el título de la aplicación y la pantalla en la que se encuentra el usuario. Esta barra será de uno de los colores que Material Design pone a disposición en su web de estilo[30]. Tanto desde esta barra como deslizando desde la parte izquierda de la pantalla hacia el centro de la misma se lanzará el menú principal de la aplicación, el cual será un NavigationDrawer, lo cual está a la orden del día en toda aplicación Android. Aquí se visualizarán todas las secciones de la aplicación, además de permitir al usuario cambiar de una a otra.

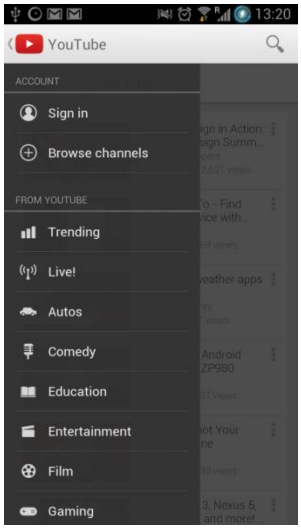

Ilustración 4.9. *NavigationDrawer* en app de YouTube

El estilo concreto dentro del contenido de cada una de las pantallas o secciones de la aplicación será definido en el diseño de las pantallas.

4.4.2. Diccionario del sistema y diseño de contenidos

La definición del estilo tiene el objetivo de homogeneizar el criterio sobre la apariencia de la interfaz. El resultado de la homogeneización del aspecto es una interfaz coherente. Para conseguir una mayor coherencia se deben definir los términos que se utilizarán en botones, mensajes de error y otros elementos similares. También se deben definir guías de estilo para la redacción de los contenidos.

La interfaz del sistema que tiene por objeto este proyecto no tiene muchos elementos a los que haya que poner un nombre, sin embargo, aunque sean pocos, deben seguir unas reglas. Estos son principalmente, los nombres de las

[30] https://www.google.com/design/spec/style/color.html

secciones, el nombre de la aplicación, algunos mensajes para avisar de la actualización de la base de datos, los títulos y datos sobre las ponencias y artículos del congreso y algunos botones para interactuar con dichos datos.

En la redacción de todos estos mensajes o nombres se deberán seguir las siguientes reglas:

- Los mensajes y nombres deberán ser claros, cortos y directos. Ya que se está hablando de una aplicación móvil que puede ser ejecutada hasta en terminales de 4 pulgadas.
- El vocabulario que se utilice debe ser sencillo y entendido por el público en general. La aplicación será utilizada por gente de una inteligencia y educación supuestamente alta, aun así se trata de una aplicación móvil orientada a Android y que cualquiera podría descargar desde la Play Store de Google.
- Los mensajes y nombres deben estar redactados en un tono cordial y amigable, siempre intentando agradar al usuario.
- Los mensajes y nombres deben encajar en todo dispositivo móvil, tanto en una Tablet de 10 pulgadas como en un teléfono móvil de 4. Ya se ha comentado algo anteriormente a cerca de este tema pero es preferible reiterar, ya que es un aspecto importante a cumplir por los mensajes y nombres de la aplicación.

4.4.3. Definición de las pantallas

En este apartado se presentan cuáles serán las pantallas que formarán la interfaz de usuario de la aplicación objeto de este proyecto. Ya se expusieron una serie de directrices generales que Google propone para todas aquellas aplicaciones orientadas a su sistema operativo, estas eran las denominadas bajo el concepto de *Material Design*.

La aplicación constará de cinco pantallas principales, además de la pantalla con el NavigationDrawer (mencionado también en la sección sobre el estilo de la aplicación). Por lo que se puede decir que se deben diseñar seis pantallas para la aplicación. Cada una de estas pantallas tendrá un diseño parecido, como ya se expuso en la sección de estilo, dispondrán de una barra de acción superior. Esta barra ocupará un espacio mínimo en la pantalla de la

aplicación, el resto de la pantalla mostrará el contenido de la sección, cuyo diseño visual se explicará a continuación:

1. **Introducción**: En esta pantalla se mostrará un breve texto, como el mismo nombre de la sección adelanta, introduciendo el Congreso sobre el Procesamiento del Lenguaje Natural. Posteriormente, se mostrará una imagen representativa de la ciudad en la que se celebrará el congreso.

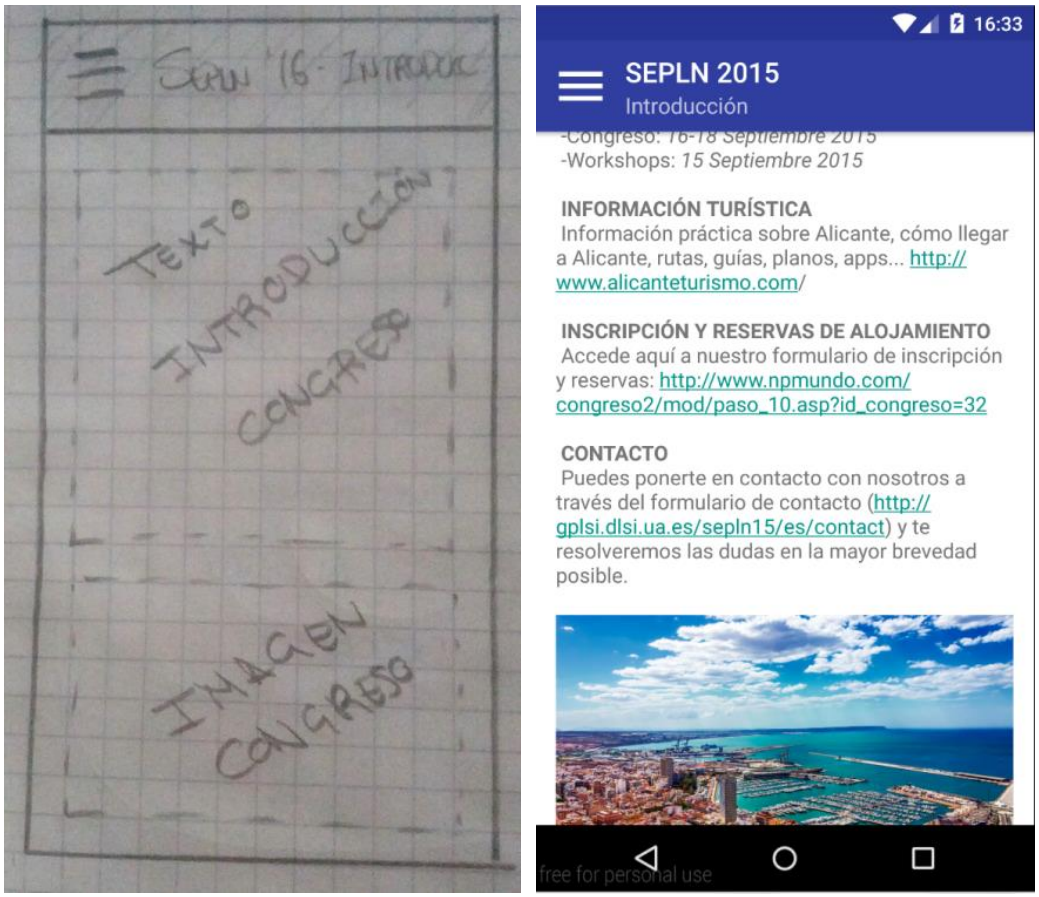

Ilustración 4.10. Diseño y resultado sección "Introducción"

2. **Artículos**: Esta pantalla mostrará una lista. Cada ítem de dicha lista contendrá la información almacenada sobre un artículo en concreto (título y autor). Además cada ítem dispondrá de dos botones en su parte inferior para descargar el artículo correspondiente o compartirlo a través de mensajería o redes sociales.

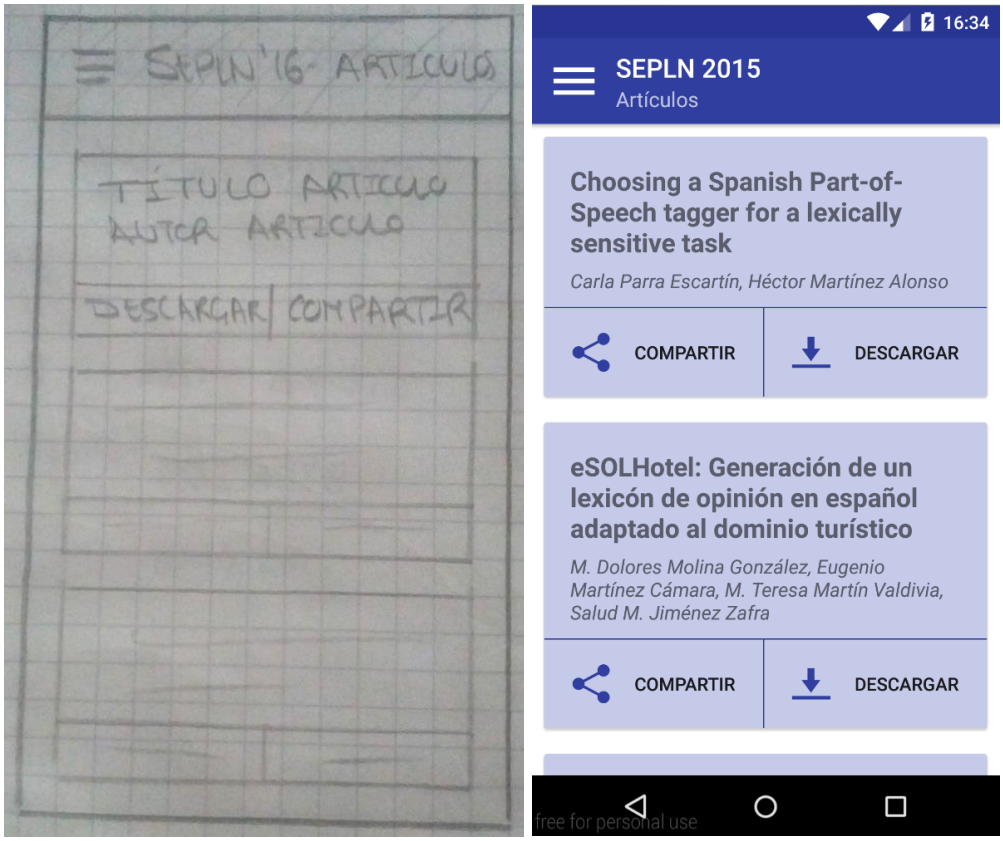

Ilustración 4.11. Diseño y resultado sección "Artículos"

3. **Programa**: Esta pantalla mostrará una serie de pestañas, cada una correspondiente aun día del Congreso. En cada pestaña se listarán los eventos correspondientes a dicho día. Como sabemos existen diferentes tipos de Eventos por lo que existirán diferentes tipos de ítems en el programa. Cada ítem de dicha lista contendrá la información almacenada sobre un evento en concreto (título, duración, sala, fecha, etc.). Además si el Evento dispone de un artículo asociado la información sobre este será mostrada y se dará la opción de descargarlo mediante un botón. Esta lista aparecerá ordenada de manera cronológica a modo de programa de eventos, como el propio nombre de la sección indica. Además cada ítem dispondrá de un botón para activar alerta.

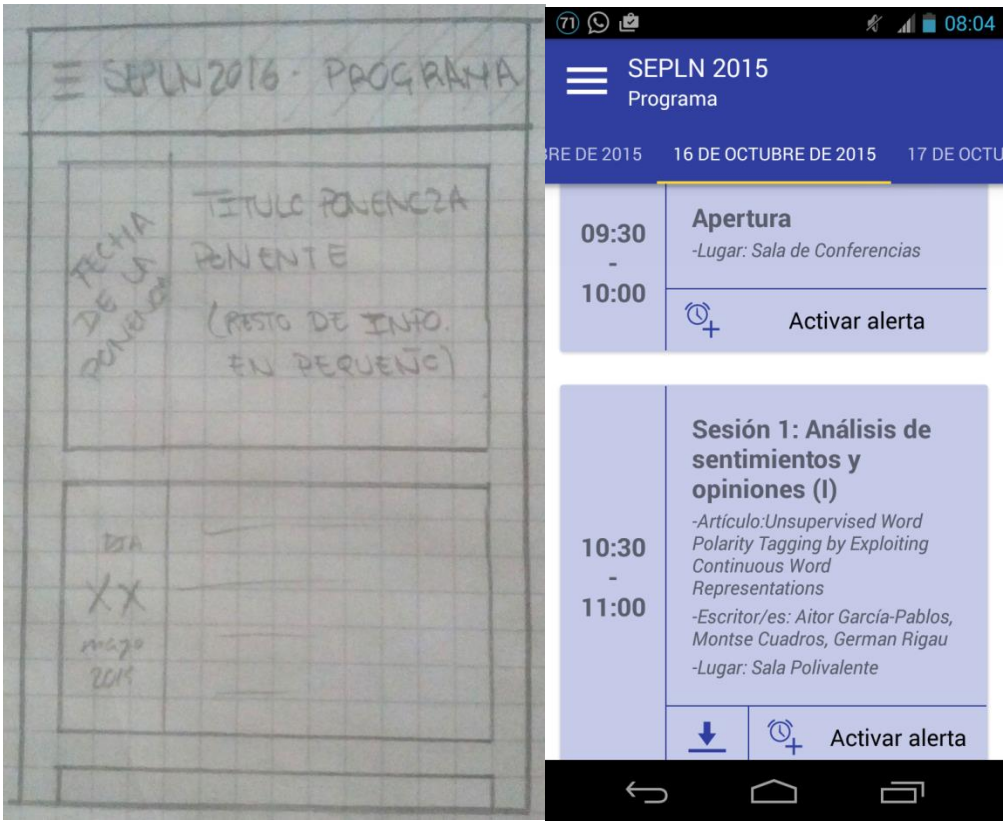

Ilustración 4.12. Diseño y resultado sección "Programa"

4. **Mis Alertas**: Esta pantalla mostrará una lista. Cada ítem de dicha lista contendrá la información almacenada sobre una alerta o notificación previamente activada por el usuario. Además cada ítem dispondrá de dos botones en su parte inferior para eliminar la notificación o alerta o compartirlo a través de mensajería o redes sociales. También dispondrá de un tiempo de notificación en pantalla que podrá ser aumentado o disminuido de manera dinámica con dos botones que se mostrarán en la parte superior e inferior del tiempo de notificación correspondientemente. Y si el evento cuya alerta está activa dispone de un artículo asociado la información sobre este será también mostrada.

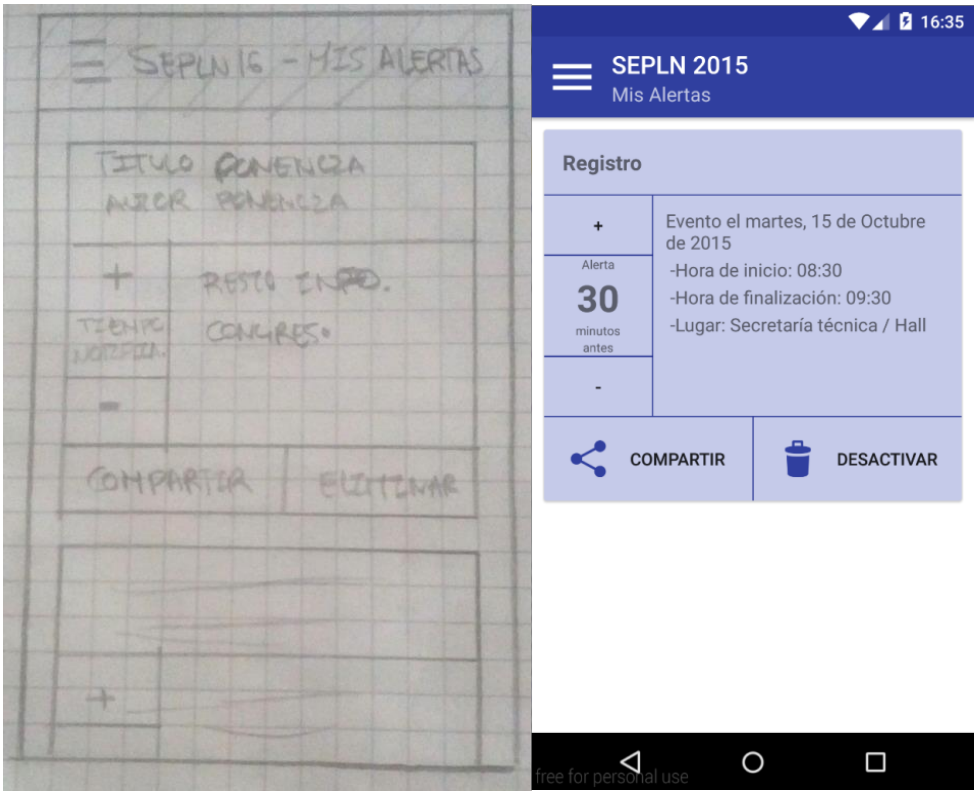

Ilustración 4.13. Diseño y resultado sección "Mis Alertas"

5. **Ubicación**: Esta pantalla será una de las más simples y a su vez una de las más complejas de implementar, ya que mostrará un mapa con la ubicación concreta del congreso y los diferentes puntos de interés. Además al pulsar sobre cada marcador se accionará un diálogo para dar la opción al usuario de ver la ruta hasta cualquiera de los puntos de interés.

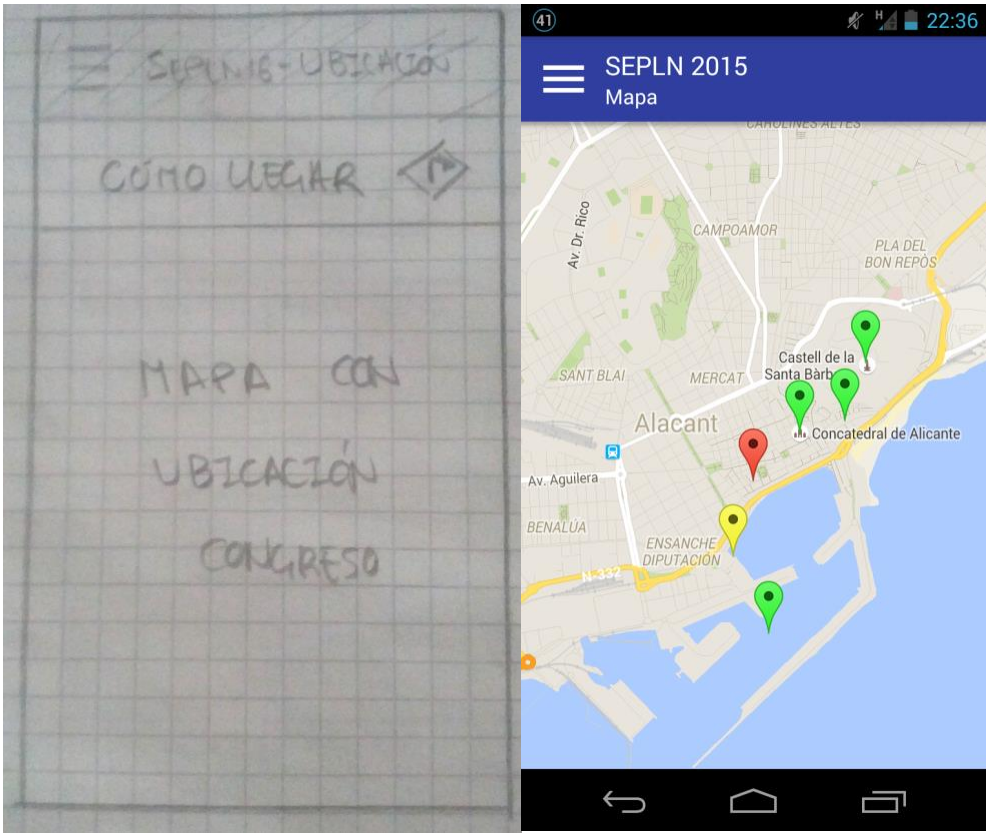

Ilustración 4.14. Diseño y resultado sección "Ubicación"

6. **Navigation Drawer**: Cómo pantalla extra, se expondrá a continuación el modo en que se mostrará en el dispositivo el menú de selección de sección. Este se mostrará de manera idéntica en cada una de las pantallas, sobre el contenido de la misma, y dispondrá de una imagen superior representativa del congreso organizado por el SEPLN y una serie de opciones representando cada una de las secciones de la aplicación.

   Cambios: tras la finalización de la primera versión de la app se han incluido dos nuevas secciones con información de la Organización y el Comité Científico del mismo, de ahí que aparezcan más opciones en el menú.

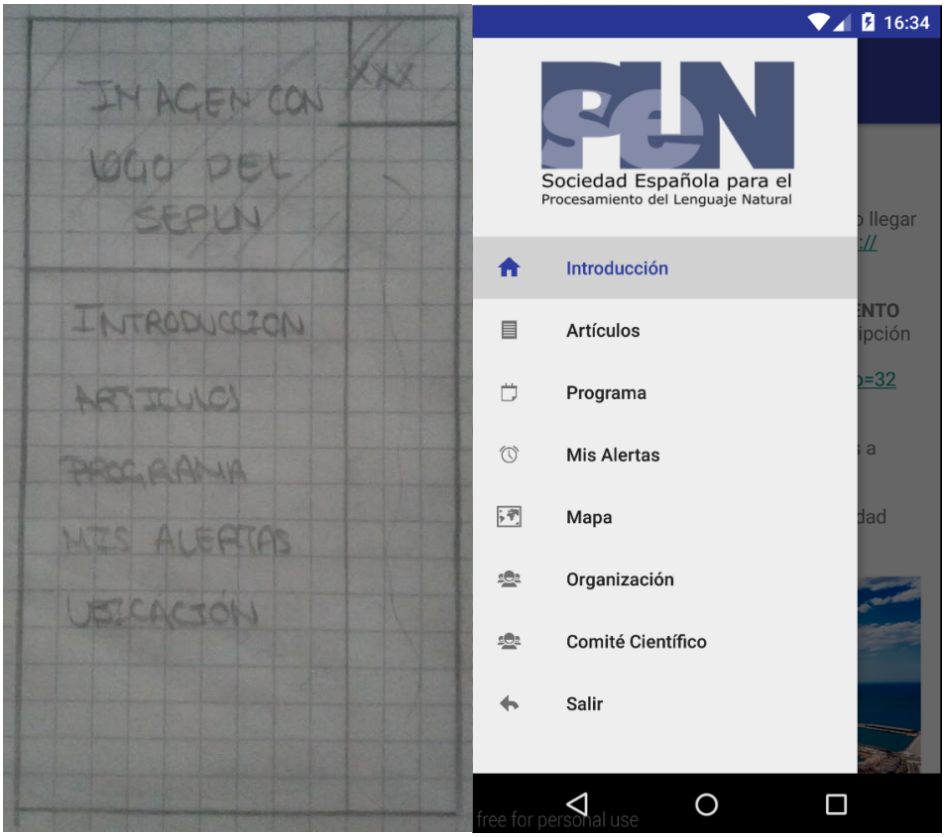

Ilustración 4.15. Diseño y resultado menú lateral

Estas son las dos pantallas anteriormente mencionadas:

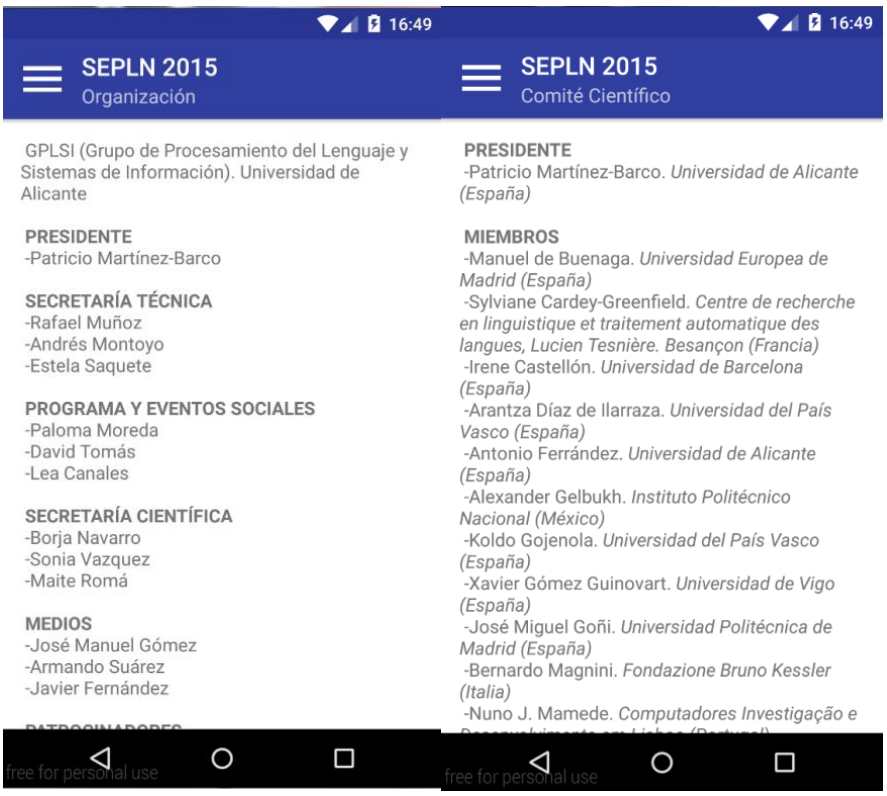

Ilustración 4.16. Resultado de las dos pantallas añadidas en la última actualización

4.5. Conclusiones del diseño

En el primer capítulo se decía que este proyecto tiene el objetivo de desarrollar una aplicación móvil que contendiera toda la información acerca del congreso anual del SEPLN. Mientras que en el capítulo dos se decía que otro de los objetivos era de realizarlo de la manera más genérica posible, de manera que esta aplicación pudiera adaptarse para un congreso futuro sobre cualquier otra materia o para el SEPLN pero de un año diferente. Por ello, se ha intentado que la solución al problema planteado en este capítulo fuera lo más genérica posible, de manera que la introducción de cambios, actualización de información y la inclusión de alguna nueva funcionalidad no suponga un gran esfuerzo. Para conseguir esto, se ha procurado que el sistema y la interfaz de usuario fuera prácticamente independiente de los datos a mostrar por la aplicación. De esta forma, para actualizar la información para un congreso de otro año del SEPLN solo sería necesario cambiar la información en la base de datos, la ubicación en la clase correspondiente y el contenido de la sección de Introducción, para adaptarlo a la nueva localización del congreso. Por otro lado, si se desea cambiar el temario del que tratará el congreso, sería necesario cambiar el título de la aplicación (ya que este incluye la temática del mismo), y la imagen del NavigationDrawer la cual es la única muestra de personalización que indica que el congreso es organizado por el SEPLN. Además, se pueden añadir nuevas secciones de manera sencilla, añadiendo una nueva opción al NavigationDrawer y enlazándola al correspondiente fragmento que implemente dicha sección.

Al seguir la arquitectura que Android plantea (modelo vista controlador), también se han independizado la interfaz gráfica (ficheros .XML) de la lógica del programa (paquete JAVA).

Una vez terminado el diseño, el siguiente paso es la construcción del sistema, es decir, la implementación, y esa es la temática del siguiente capítulo.

5. IMPLEMENTACIÓN

Llegado a este punto ya está definido el problema, y su solución, por lo que ya es momento de construirla.

En este capítulo se describirá cómo se ha construido cada una las partes de la solución descrita en el anterior capítulo.

5.1. Herramientas

En esta primera sección del *Capítulo 5* se describirán las herramientas concretas que se emplearán para la implementación de la solución al problema planteado por este proyecto.

5.1.1. Android Studio

Es un entorno de desarrollo integrado especificamente para la plataforma Android. Fue anunciado el 16 de mayo de 2013, por lo que es un IDE bastante actual.

Características principales:

- Renderización en tiempo real
- Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Soporte para construcción basada en Gradle.
- Refactorización especifica de Android y arreglos rápidos.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones, y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para Android Wear.[31]

Otra opción para la implementación de aplicaciones Android es Eclipse junto a su plugin para desarrolladores de Android, sin embargo se ha anunciado que esta plataforma dejará de tener soporte pronto. Por lo que Android Studio es el presente y futuro en el desarrollo de aplicaciones Android y por ello de su elección.

5.1.2. Google Maps API

Todo el mundo hoy en día conoce Google Maps, pero aquí va una breve definición: Google Maps es un servidor de aplicaciones de mapas en la web que pertenece a Google. Ofrece imágenes de mapas desplazables, así como fotografías por satélite del mundo e incluso la ruta entre diferentes ubicaciones o imágenes a pie de calle.[32]

[31] Información obtenida de la web: https://es.wikipedia.org/wiki/Android_Studio
[32] Definición obtenida de la web: https://es.wikipedia.org/wiki/Google_Maps

Ahora se va a proceder a definir qué es una API, lo que ayudará a comprender mejor qué es la Google Maps API. Una interfaz de programación de aplicaciones, abreviada como API, es el conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas de programación.[33]

En junio de 2005 Google lanzó su API de Google Maps, haciendo oficialmente modificable casi cualquier aspecto de la interfaz original. Con la contraseña oficial de desarrollador, la API es libre de uso para cualquier sitio web o aplicación móvil.

Una vez comprendido el concepto que representa la Google Maps API, sólo queda comentar su uso en la aplicación para el congreso anual del SEPLN. Principalmente se va a emplear esta API para la implementación de uno de los requisitos funcionales principales de la aplicación que ya se expusieron en el *Capítulo 3*, en la *Sección 3.2.1* Este requisito es "*Será posible consultar la ruta desde la ubicación del dispositivo hasta la ubicación del congreso*".

5.1.3. SQLite

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275~kiB) biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D-Richard Hipp.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada

[33] Definición obtenida de la web: https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

transacción. En su versión 3, SQLite permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.[34]

5.1.4. Gimp

GIMP o GNU Image Manipulation Program, es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Es un programa libre y gratuito. Forma parte del proyecto GNU y está disponible bajo la *Licencia pública general de GNU* y *GNU Lesser General Public License*. Es el programa de manipulación de gráficos disponible en más sistemas operativos, además de estar disponible en varios idiomas.[35]

Durante el desarrollo del TFG se emplea este programa para la modificación de las imágenes que en el mismo aparecen (tanto en la aplicación como en la memoria). Decidí emplear este programa ya que es el que aprendí a utilizar cuando era más pequeño, además de ser software libre (al contrario que ocurre con otras aplicaciones de modificación de imágenes cómo podría ser el archiconocido Photoshop).

5.2. Construcción de la interfaz y menús

Para continuar con el Capítulo de Implementación se va a explicar cómo ha sido la construcción de la interfaz de la aplicación, es decir la parte que será visible al usuario.

Se decidió comenzar por la implementación de este apartado ya que para una aplicación móvil, esta es una de las partes más importantes del desarrollo. La interfaz determinará si un usuario gusta de la aplicación o, sin embargo, este la deshecha en menos de un minuto.

Para la construcción de la interfaz de la aplicación de la app, se han seguido los criterios de diseño expuestos por Google (Material Design), como ya se dijo en el Capítulo de Diseño. Por lo tanto esta estará formada por los siguientes componentes:

[34] Información extraída de la web: https://es.wikipedia.org/wiki/SQLite
[35] Información obtenida de la web: https://es.wikipedia.org/wiki/GIMP

5.2.1. Actionbar superior

Según Material Design esta debe aparecer en la parte superior de la aplicación en cada una de las pantallas. Esta barra mostrará el título de la app y el lugar en el que nos encontramos de la misma. Además desde esta se podrá abrir el menú lateral desde el que se cambiará de sección.

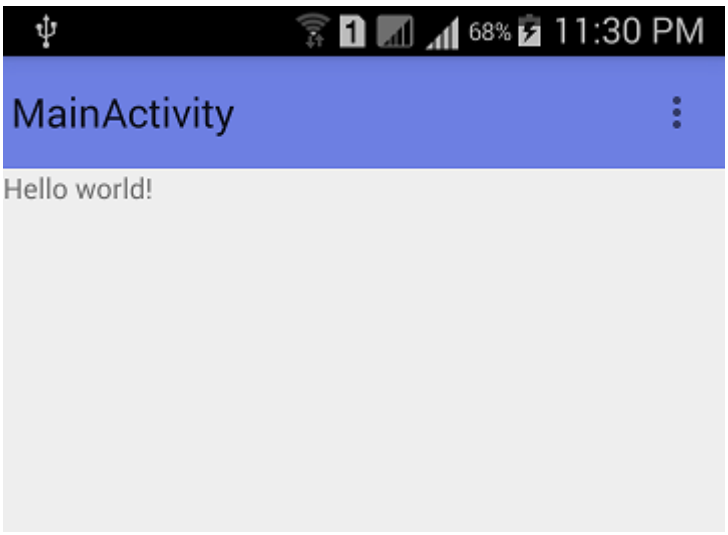

Ilustración 5.1. Ejemplo de toolbar por defecto

Android ofrece por defecto esta barra. Aunque se decidió implementar una personalizada (con su layout correspondiente), ya que en la sección Programa será necesaria la inclusión de pestañas y para integrar las pestañas con la Toolbar será necesario personalizar dicha barra. Además se eliminará el botón del menú de la derecha ya que este no será empleado en la aplicación y se añadirá el botón de menú a la izquierda del texto para posteriormente asociarlo con el menú lateral. También será gestionado el cambio de subtítulo, el cual irá cambiando en función de la sección de la sección que se esté mostrando en cada momento.

Toda la funcionalidad correspondiente a la Actionbar superior es implementada en la clase MainActivity. Esta controlará el cambio de subtitulo, si se debe mostrar la barra de pestañas o no en función de la pantalla en la que se encuentre el usuario, se controlará la apertura y cierre del menú, etc. Esto ya se explicó en el *Capítulo 4: Diseño*, tras la exposición del diagrama de clases de la aplicación.

5.2.2. Menú lateral

El menú lateral, también llamado DrawerLayout, estará compuesto por una imagen representativa de la Sociedad del Procesamiento del Lenguaje Natural en la parte superior y una serie de opciones mostradas en una lista de manera descendiente. Estas opciones serán cada una de las pantallas de la aplicación, además de una opción para salir de la misma en la parte inferior del menú.

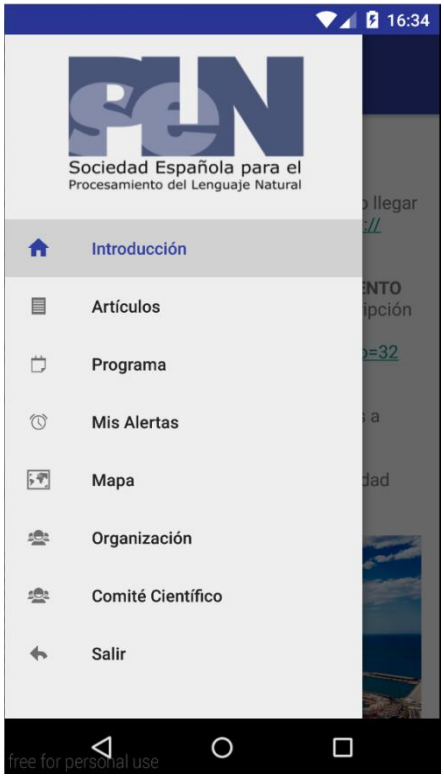

Ilustración 5.2. Menú lateral de la aplicación

Este menú estará asociado a la Toolbar anteriormente descrita, y toda la gestión del mismo será incluida en la clase MainActivity. Se deberá asociar el Layout del menú con el DrawerLayout y el DrawerLayout con el botón en la Toolbar. Además se debe asociar a este un fichero de la carpeta menú (un recurso), el cual contendrá todas las opciones que se mostrarán en el menú.

Finalmente, se deberá indicar la acción a realizar al pulsar cada una de las opciones del menú. En este caso se ha optado por implementar una función para cada una de las secciones la cual se encargará de crear una instancia del Fragment correspondiente e intercambiarlo por el actual. Además de mostrar u ocultar la barra de pestañas en caso de que el Fragment corresponda a la sección Programa o a otra de las secciones.

Con esto queda concluido el apartado de la interfaz. Tras llevar a cabo lo anteriormente descrito ya se dispondrá de una versión de la app en la que se podrá navegar por los diferentes Fragments de la misma. Ahora solo quedaría por delante la implementación de la funcionalidad de dichos Fragments.

5.3. Construcción de los datos

En el párrafo anterior se dijo que ya solo quedaría por delante la implementación de la funcionalidad de los diferentes Fragments. Sin embargo nos encontramos con esta sección primero. Esto es porque para la implementación de la funcionalidad de cada uno de los Fragments de la aplicación previamente habrá que construir el sistema de gestión de los datos, ya que algunos de los Fragments harán uso del mismo.

5.3.1. Implementación del diagrama entidad-relación

Para comenzar se va a explicar cómo se ha implementado el diagrama entidad-relación diseñado en el *Capítulo 4*.

Si volvemos a mirar el diagrama vemos que hay una tabla con Ubicaciones independiente a las demás, por lo tanto se creará una tabla Ubicaciones. Esta dispondrá de un id, dos reales representando la latitud y la longitud correspondientemente, un nombre y un tipo. En este caso el tipo será siempre 0, 1 o 2. Se debe tener cuidado a la hora de la inserción de que solo haya una ubicación de tipo 0, ya que esta será la Ubicación principal del congreso

Después vamos a la parte complicada, podemos ver como existe una entidad Evento. Esta entidad tiene una relación de generalización ISA con cada uno de los tipos de Eventos existentes en el Congreso. Para hacer la implementación mucho más simple y sencilla se va a realizar una abstracción de los tipos de Evento: existirán tan solo dos tipos de Evento, Eventos simples formados por un título, una sala, una duración, una fecha y una notificación; y Eventos con artículo asociado, los cuales dispondrán de todo lo anterior más un artículo (nombre), un autor/es del mismo (encargados también de su presentación, evidentemente) y una url al artículo en formato pdf.

De esta manera, para la implementación de todo el diagrama solo serán necesarias dos tablas: tabla Ubicaciones y tabla Eventos (de esta se extraerán los Artículos también).

5.3.2. Proceso de implementación de los datos

Como se explicó en el *Capítulo 4* se utilizará la herramienta que Android proporciona para la gestión de base de datos. Esta es SQLite, la cual también fue descrita con anterioridad. Para ello será necesaria la construcción de una clase que extienda a SQLiteOpenHelper. Esta clase, en este proyecto, será llamada GestionBD e implementará los métodos onCreate y onUpdate. La funcionalidad de estas no es otra que crear la base de datos y actualizarla en caso de cambio de versión de la misma respectivamente, como los propios nombres de los métodos indican.

Para la construcción de estas dos funciones se ejecutan una serie de scripts .SQL, los cuales contendrán las acciones SQL a ejecutar. Ambos métodos leerán los ficheros .SQL correspondientes línea a línea y las irán ejecutando mediante el uso de la función execSQL.

Para proseguir, se procederá a describir cada uno de estos ficheros .SQL. Existirán principalmente tres scripts: uno con las acciones necesarias para crear las dos tablas que forman el sistema, otro con las acciones correspondientes al borrado y uno más con todas las inserciones necesarias para llenar la base de datos.

Finalizando este capítulo, se explicará el método de empleo de esta clase en los Fragments que de ello requieran. Para el uso de la base de datos será necesario realizar una conexión a la misma, posteriormente realizar las consultas o cambios oportunos y para finalizar cerrar dicha conexión. Esto se realiza en Android de la siguiente manera:

1. Creación de un objeto de la clase GestorDB, para la cual debemos utilizar la actividad actual, el nombre de la base de datos (el cuál será almacenado en el fichero de recursos strings de manera que será más fácil mantener la integridad en la implementación de la aplicación) y la versión de la base de datos. De esta manera se detectará si la base de datos con este nombre está creada, en caso negativo se llamará al método goncearte y en caso afirmativo se comprobará si la versión es igual a la que se ha indicado, en caso contrario se llamará al método onUpgrade.

2. Obtención de un escritor o lector de la misma. Para escribir o consultar en la base de datos será necesario un objeto de tipo SQLiteDatabase obtenido a partir de la conexión a la base de datos realizada en el paso 1 con la creación del objeto de la clase GestorDB.
3. Se emplea este escritor o lector para realizar las acciones oportunas en la base de datos.
4. Se cierran tanto el lector o escritor cómo la conexión a la base de datos realizada por el objeto de tipo GestorDB.

5.4. Construcción de las secciones

Ya se realizó una descripción de cada una de las clases que representaba cada uno de los Fragments de la aplicación en el *Capítulo 4: Diseño*. Sin embargo, al igual que en la sección anterior (Construcción de los datos), en esta se expondrán los pasos seguidos para la implementación de cada una de las secciones.

Para comenzar se debe señalar que cada uno de estos Fragments son mostrados en un FrameLayout de la aplicación. La MainActivity dispone de un layout en el que se incluyen el menú lateral (DrawerLayout), la barra superior (Toolbar y TabLayout) y dicho FrameLayout. De esta manera se define la composición de cada una de las pantallas de la aplicación, en la cual solo cambiará el contenido de este FrameLayout en función de la sección seleccionada.

Una vez explicado esto, se van a presentar los pasos seguidos para la implementación de cada una de las secciones o Fragments:

5.4.1. Introducción

El Fragment que se mostrará en pantalla cuando el usuario escoja la sección de Introducción posee una de las construcciones más sencillas.

El primer paso será la definición de la composición de la pantalla. Esta se realizará mediante el layout correspondiente. Este estará compuesto por un LinearLayout, esto es un tipo de contenedor en Android en el cual se sitúan los diferentes elementos o vistas de manera secuencial uno detrás del anterior. Este se puede definir vertical u horizontal, en este caso será vertical. Además estará incluido en un ScrollView, para hacer posible el desplazamiento de la pantalla en

vertical si la información excede el tamaño de la pantalla. Dentro del LinearLayout se ubicarán los dos elementos principales: un TextView, esto es un campo de texto, y un ImageView, esto es una imagen la cual al encontrarse en un LinearLayout vertical se mostrará a continuación del texto. El texto será extraído del fichero de recurso string (donde se encuentran todas las cadenas de caracteres que usará la app). La imagen será un recurso de tipo imagen (drawable en Android).

El siguiente paso será la construcción de la parte lógica del Fragment. Esta sección es muy sencilla, y su implementación e prácticamente trivial. Además ya fue explicada en el Capítulo de Diseño.

5.4.2. Artículos

El Fragment que se mostrará en pantalla cuando el usuario escoja la sección de Artículos posee una construcción algo más compleja que la expuesta anteriormente. Ya que esta incluirá una lista con los Artículos del congreso.

El primer paso será la definición de la composición de la pantalla (esto no cambia). En este caso se utilizarán dos layouts. El primero de ellos (fragment_recycler_articulos) será el que contenga el elemento RecyclerView, esto no es otra cosa más que la lista de Artículos. El segundo layout será el que definirá la vista de cada una de las celdas de la lista de Artículos. Su nombre es item_articulo y dispone de un texto con el título del artículo, otro texto debajo de este con el autor o autores del mismo, y, una vez más, debajo de este aparecerán dos botones: uno para descargar el artículo y el otro para compartirlo en las redes sociales.

El siguiente paso será la construcción de la parte lógica del Fragment. Este está compuesto por una clase que extiende a Fragment y otra clase de tipo Adapter, encargada de rellenar los datos de cada una de las entradas de la lista. El funcionamiento de esto ya se comentó en el *Capítulo 4: Análisis*. El Fragment cargará la lista de artículos desde la base de datos y asociará el RecyclerView mencionado en el párrafo anterior con el Adapter. Este se encargará de inflar cada uno de los elementos que componen un ítem de la lista asignándole el valor correspondiente de la lista de Artículos precargada de la base de datos anteriormente.

5.4.3. Programa

La construcción de esta sección es muy similar a la construcción de la sección Artículos anteriormente descrita. Aunque introduce una particularidad, esta dispondrá de las pestañas para separar los diferentes Eventos del Congreso por días.

Para ello se definirán tres layouts: el primero será Fragment_program_pager, el cuál contendrá un ViewPager (ya definido en secciones anteriores), el segundo layout será fragment_recycler_programa, este será análogo al fragment_recycler_articulos de la Sección Artículos, y finalmente un item_programa, este contendrá las vistas que formarán cada uno de los elementos de la lista que forma el Programa. Este último estará formado por dos "tipos" de vistas, ya que la forma de modelar el sistema es con varios tipos de Eventos y estos se mostrarán utilizando este mismo layout pero serán diferentes:

- Vistas que se mostrarán siempre, tanto en los Eventos más simples como en los Eventos con Artículo asociado: título del Evento, lugar del Evento, hora del Evento y un botón para activar alerta.
- Vistas que solo serán mostradas en Eventos con un Artículo asociado (serán ocultados en los Eventos simples). Estos serán un texto con el título del Artículo asociado, un texto con el autor o autores del Artículo asociado y un botón de descarga del Artículo asociado.

Todo esto se gestionará en el apartado lógico de la aplicación. Existen tres clases para manejar esta sección:

- **FragmenPagerProgram**: maneja el ViewPager. En el se extraen de la base de datos los días del congreso para crear las instancias del FragmentListPrograma que correspondan. Esto se hace accediendo a la base de datos y realizando un SELECT DISTINCT sobre la tabla de Eventos. Posteriormente se asociarán dichas instancias con cada una de las pestañas del TabLayout.
- **FragmentListPrograma**: maneja la lista de Eventos correspondiente a un día determinado. Existirán tantas instancias de este Fragment cómo días tenga el Congreso. En la creación de cada Fragment se accederá a la

base de datos y se extraerán en una lista los Eventos correspondientes al día del Fragment. Posteriormente se llamará al adaptador.

- **AdapterRecyclerProgram**: este es el adaptador para la lista de Eventos de cada uno de los días del Congreso. Se encargará de inflar el layout item_programa para cada una de las celdas, asociando la información contenida en la lista correspondiente al día del Fragment con las vistas de un item_programa. Para realizar la distinción entre Eventos, se comprueba si el Evento dispone de un Artículo asociado, si así es se inflarán también las vistas particulares de este tipo de Eventos, en caso contrario estas serán ocultad con el método setVisible.

  También en esta clase se implementa la funcionalidad de los botones correspondientes. Por lo que se hará uso de la clase GestorAlertas para activar las alertas correspondientes en caso de que el usuario pulse el botón. Antes de activar cada una de las alertas se realizan varias comprobaciones: primero se compruebe que el Evento no haya pasado, ya que si el Evento se encuentra en el pasado no tiene mucho sentido activar una alerta, posteriormente se comprueba que la alerta no está ya activa, tampoco tendría sentido activarla si esta ya está activa. Y, finalmente, se activará la alerta si ninguna de las condiciones anteriores se cumple. Además durante el proceso se lanzarán notificaciones al usuario de tipo SnackBar (con una acción) para informarle del resultado de su acción y ofreciéndole una alternativa, como eliminar una alerta si esta ya ha sido activada o deshacer la activación de una alerta que acaba de activar.

5.4.4. Mis Alertas

La construcción de esta sección es una analogía a una combinación de las secciones anteriores, por lo que se tratará de explicar con la mayor brevedad posible.

Lo primero, como siempre, la definición de los layouts: se definirán dos layouts, uno con la lista de Mis Alertas y otro que definirá cada ítem mostrado en esta lista. Este segundo layout es algo más complejo que los anteriores, dispondrá de una sección superior con el título del Evento, y el título y autores del Artículo en caso de que el Evento tenga un Artículo asociado; una sección inferior con dos botones, para

desactivar la alerta o compartir el Evento en las redes sociales; una sección lateral con un botón superior de aumento (+), un botón inferior de disminución (-), y un número en el centro indicando el tiempo de notificación el cual será modificado por los botones mencionados realizando las comprobaciones pertinentes previamente.

La parte lógica de esta sección es análoga a la de la sección de Artículos, con la particularidad de comprobar si el Evento dispone o no de Artículo asociado en el adaptador, para inflar u ocultar las vistas correspondientes (de manera similar a cómo se hacía en la sección de Programa).

5.4.5. Mapa

El Fragment que se mostrará en pantalla cuando el usuario escoja la sección de Mapa posee una de las construcciones más complejas en la aplicación, ya que requiere del paso previo de inclusión de la API de Google Maps.

Por lo tanto, el primer paso será incluir la API en la aplicación, esto se realiza de una manera relativamente sencilla, aunque yo me encontré con un problema durante el proceso de inclusión de la misma. Para incluir la API es necesario acceder a la consola de desarrollo de Google, para ello se debe poseer una cuenta de Google. Una vez allí se debe crear un nuevo proyecto. Tras esto se accede a la sección de APIs y una vez ahí activar la de Google Maps para Android. Tras esto se debe obtener una clave para incluirla en la aplicación. Para la generación de esta clave es necesaria una huella digital. Y aquí estuvo mi problema. Ya que no sabía cómo obtener dicha clave y me tomó bastante tiempo. Una vez obtenida la clave se incluye en el archivo Manifiest de la aplicación, además de los permisos correspondientes que Google recomienda pedir para el funcionamiento de la API.

Tras esto, paso a la definición de la composición de la pantalla. Esta se realizará mediante el layout correspondiente. Este estará compuesto por un LinearLayout vertical. En el que se encontrará un MapView al que se le asigna un id, para posteriormente configurarlo en la parte lógica correspondiente de este Fragment.

Por último se ha de definir el apartado lógico de esta sección. Esto es la clase FragmentMapa. En el método onActivityCreated se configurará la vista MapView definida en el layout de la sección. Lo primero será identificar esta vista

en el código java mediante un objeto de tipo MapView. Posteriormente se crea una instancia del MapView y se asocia a un objeto de tipo GoogleMap, el cual será empleado para realizar la configuración pertinente. El siguiente paso será acceder a la base de datos (como ya se explicó anteriormente) y definir un marcador para cada una de las Ubicaciones (puntos de interés) que allí encontramos. Para esto se recorre la tabla Ubicación y se crea el marcador en función del tipo. Existen tres tipos de puntos de interés en esta aplicación:

- 0: Únicamente hay uno, y es la Ubicación del congreso. Por lo que en este caso se almacenará su LatLng en un atributo del Fragment (que será empleado posteriormente) y se crea un marcador de color rojo (por defecto) con el LatLng y nombre correspondiente.
- 1: Es un punto de interés de tipo monumento. Puede haber varios. En este caso se asigna al mapa un marcador de color verde con el LatLng y nombre correspondiente.
- 2: Es un punto de interés de tipo actividad cultural (actividades culturales en el programa del congreso que son desarrolladas en lugares diferentes a la ubicación principal del congreso). Puede haber varios. En este caso se asigna al mapa un marcador de color amarillo con el LatLng y nombre correspondiente.

Para continuar, una vez asignados todos los marcadores, se indicará la acción al pulsar un marcador. Esta no será otra más que mostrar un diálogo personalizado mostrando el nombre asignado al marcador (nombre del punto de interés) y dos opciones: volver al mapa o ver ¿cómo llegar?. En el caso de la segunda opción se lanza un Intent para abrir otra actividad, en este caso se trata de una URL que el sistema interpreta y abre con una aplicación externa dedicada a tal cometido (Google Maps, navegador, etc.).

Finalmente, se crea un objeto de tipo CameraPosition para situar la cámara centrada en la ubicación del congreso (anteriormente guardada) y con un zoom correspondiente. Después se llama al método moveCamera desde el objeto GoogleMap atributo de la clase.

5.4.6. Organización

El Fragment que se mostrará en pantalla cuando el usuario escoja la sección de Organización posee una de las construcciones más sencillas.

El primer paso, como en los casos anteriores, será la definición de la composición de la pantalla. Esta se realizará mediante el layout correspondiente. Este estará compuesto por un LinearLayout vertical contenido en un ScrollView. Dentro del LinearLayout se ubicará únicamente un TextView, esto es un campo de texto. El texto será extraído del fichero de recurso string y será una lista con los diferentes representantes que forman la organización del congreso extraída de la web *http://gplsi.dlsi.ua.es/sepln15/es/organizacion*.

El siguiente paso será la construcción de la parte lógica del Fragment. Esta sección es muy sencilla, y su implementación es prácticamente trivial.

5.4.7. Comité

El Fragment que se mostrará en pantalla cuando el usuario escoja la sección de Comité posee una de las construcciones más sencillas y es análoga a la construcción de las secciones Introducción y Organización.

El primer paso, como en los casos anteriores, será la definición de la composición de la pantalla. Esta se realizará mediante el layout correspondiente. Este estará compuesto por un LinearLayout vertical contenido en un ScrollView (ya se describió lo que era esto en la explicación de la sección Introducción). Dentro del LinearLayout se ubicará únicamente un TextView, esto es un campo de texto. El texto será extraído del fichero de recurso string y será una lista con los diferentes representantes que forman el comité científico del congreso extraída de la web *http://gplsi.dlsi.ua.es/sepln15/es/node/30*.

El siguiente paso será la construcción de la parte lógica del Fragment. Esta sección es muy sencilla, y su implementación es prácticamente trivial.

5.5. Internacionalización

Android proporciona un sistema muy sencillo para la internacionalización de sus aplicaciones. Toda aplicación Android dispone de un directorio de recursos. Estos pueden ser imágenes, cadenas de caracteres, menús, layouts, ficheros de texto, etc. Y Android proporciona la posibilidad de realizar duplicados de los mismos y clasificarlos en diferentes directorios. Esto es más fácil de

entender con un caso real. En la aplicación objeto de este TFG se dispondrá de un directorio configurado para el idioma español y otro para el contenido en caso de que el idioma escogido sea el inglés. El sistema se encargará de escoger los ficheros de uno u otro en función del idioma del dispositivo.

Por lo tanto, para la internacionalización de la aplicación será tan simple como traducir el fichero que contiene todas las cadenas de caracteres en español al inglés y situar este nuevo fichero en el directorio de inglés. Además también se han traducido los scripts de la base de datos, para que la información que ellos contienen también aparezca en ambos idiomas.

6. PRUEBAS

En este capítulo se describen las pruebas a las que se ha sometido el software para comprobar que se cumplen los requerimientos, tanto los funcionales como los no funcionales.

6.1. Introducción

En el desarrollo de cualquier software, una de las actividades asociadas a este proceso es la prueba. Las pruebas del software son un elemento crítico para la garantía de calidad del software y representan una revisión final del diseño y de la codificación.

La prueba es el proceso de ejecución de un programa con el intento deliberado de encontrar errores. Su objetivo es la verificación y validación del producto. La verificación se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica, y la validación se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requisitos del cliente.

De acuerdo con la "IEEE Standard for Software Test Documentation", el concepto de prueba se define como:

"Una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente."

Junto con esta definición aparece también el concepto de caso de prueba:

"Un conjunto de entradas, condiciones de ejecución y resultados esperados diseñados para un objetivo particular."

6.2. Técnicas de prueba genéricas

Se va a comenzar esta Sección realizando una breve introducción a las pruebas genéricas para todo producto software. Posteriormente se desarrollarán algunas de estas pruebas sobre la aplicación.

Las técnicas de prueba del software se pueden clasificar en:

- **Pruebas de caja blanca**: La prueba de caja blanca, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que examina la estructura de control interna del programa para obtener los casos de prueba. Existen distintos tipos de pruebas que se consideran de caja blanca:
    - **Prueba de camino básico**: La mayoría de los programas cuentan con distintos caminos de ejecución. El camino básico es un mecanismo para comprobar que todos esos caminos son

accesibles desde la interfaz de usuario, y que se ejecutan sin errores.

- **Pruebas sobre las estructuras de control**: Se centran en la evaluación de las estructuras de control, como son: bucles y estructuras condicionales.

Ninguna de estas pruebas será expuesta en esta memoria, ya que con la cantidad de secciones y funcionalidades de la app obtendríamos una gran cantidad de caminos básicos. También se debe reseñar que la aplicación no incluye ningún algoritmo de alta complejidad, por lo que no se considera necesario realizar pruebas sobre las estructuras de control. Además, en el flujo de implementación de cada uno de los aspectos de la aplicación se han ido realizando pruebas de este tipo, para comprobar que todo funciona correctamente. Así que se podría afirmar que se han realizado pruebas de caja blanca en la aplicación de manera continua durante el desarrollo de la misma.

- **Pruebas de caja negra**: Las pruebas de caja negra, también denominada prueba de comportamiento, permiten obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Las pruebas de caja negra no es una alternativa a las de caja blanca, más bien se trata de un enfoque complementario.

  En la ingeniería del software, los casos de prueba o Test Case son un conjunto de condiciones o variables bajo las cuáles el equipo de desarrollo determinará si el requisito de una aplicación es parcial o completamente satisfactorio.

  Se pueden realizar muchos casos de prueba para determinar que un requisito es completamente satisfactorio. Con el propósito de comprobar que todos los requisitos de una aplicación son revisados, debe haber al menos un caso de prueba para cada requisito. Por lo tanto se desarrollarán tantos bloques de prueba como requisitos tenga una aplicación. En este caso los requisitos funcionales (los no funcionales serán verificados mediante pruebas concretas para Android, *Sección 6.3*) y sus respectivos casos de prueba son los siguientes:

6.2.1. Probar: Será posible consultar la ruta desde la ubicación del dispositivo hasta la ubicación del congreso.

Para comprobar este requisito se realizará un único caso de prueba. Este será muy sencillo y estará formado por los siguientes pasos a realizar manualmente en la aplicación:

1. Acceder a la aplicación.
2. Abrir el menú lateral.

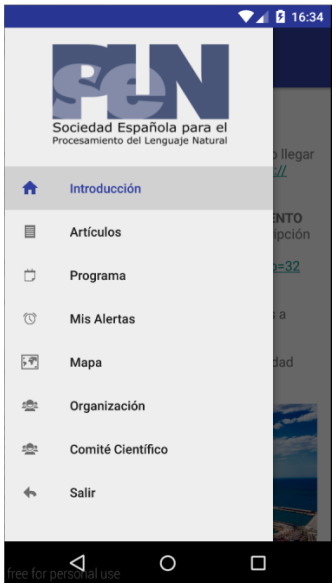

Ilustración 6.1. Menú lateral de la aplicación.

3. Seleccionar la sección Mapa.
4. Comprobar que el mapa se abre.

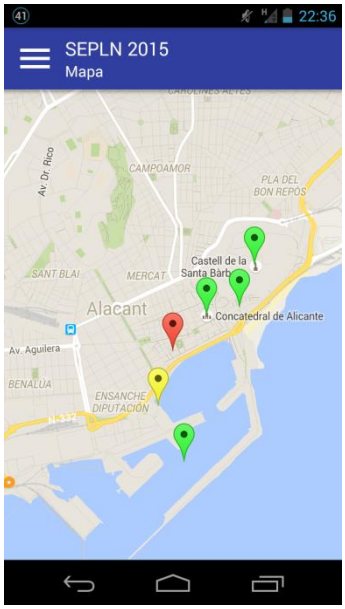

Ilustración 6.2. Se observa que el mapa de la aplicación se abre correctamente.

5. Seleccionar el marcador que señala la ubicación del congreso.
6. Escoger la opción “¿Cómo llegar?” del cuadro de diálogo.

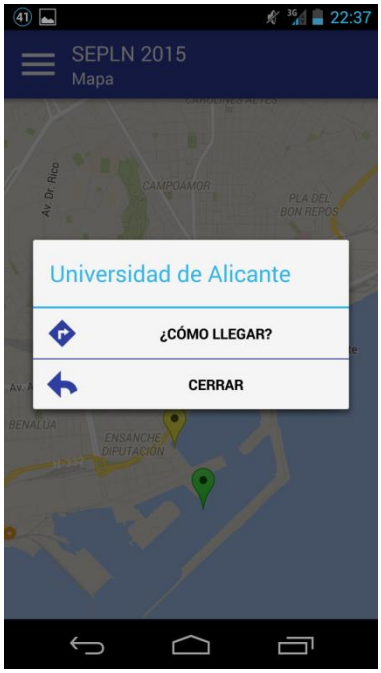

Ilustración 6.3. Cuadro de diálogo en el que se debe escoger la opción ¿Cómo llegar?

7. El sistema se encargará de mostrar una lista de aplicaciones con las que se puede mostrar la ruta.
8. En este caso se seleccionará Google Maps.
9. Google Maps se encargará de mostrar la ruta.

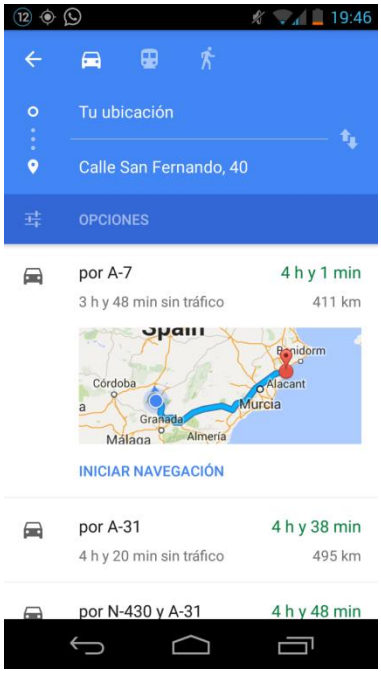

Ilustración 6.4. Captura de pantalla de como se muestra la ruta en Google Maps (1)

Ilustración 6.5. Captura de pantalla de como se muestra la ruta en Google Maps (2)

Por lo que se puede afirmar que esta funcionalidad se cumple de manera satisfactoria.

6.2.2. Probar: Se podrá consultar el programa del congreso.

Para comprobar este requisito se satisface se desarrollará un único caso de prueba. Al igual que el anterior este será muy sencillo y estará formado por los siguientes pasos a realizar manualmente en la aplicación:

1. Acceder a la aplicación.
2. Abrir el menú de la aplicación.
3. Seleccionar la opción “Programa”.
4. Comprobar como se abre el Programa del Congreso.

Ilustración 6.6. Se comprueba como se ve la pantalla del congreso

5. Hacer scroll hacia abajo para ver los Eventos que no caben en pantalla y comprobar el correcto funcionamiento del scroll.
6. Navegar por las diferentes pestañas para comprobar que se muestran los Eventos de manera satisfactoria.

Se puede afirmar que este requisito funcional también es implemenado satisfactoriamente por la aplicación.

6.2.3. Probar: Se podrán consultar los diferentes articulos relativos al congreso.

Una vez más, para la comprobación de que este requisito es cumplido satisfactoriamente, se realizará una única prueba, la cuál también es de un nivel de sencillez muy alto. Además, esta prueba también será realizada manualmente. Se deberán seguir los siguientes pasos:

1. Acceder a la aplicación.
2. Abrir el menú de la aplicación.
3. Seleccionar la opción “Artículos”.
4. Comprobar como se abre la lista de Artículos del Congreso.

Ilustración 6.7. Pantalla de la sección Artículos de la aplicación

5. Hacer scroll para ver el resto de artículos que no son mostrados al quedar fuera de la pantalla.
6. Descargar un Artículo pulsando en el botón “Descargar”.

7. Acceder al Artículo, el cuál aparecerá en la barra de notificaciones superior al completar su descarga.

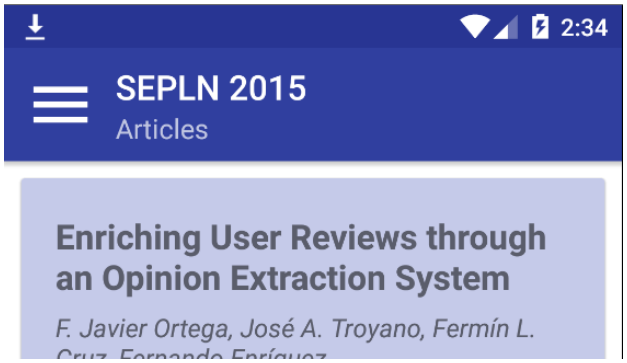

Ilustración 6.8. Se aprecia la notificación de descarga en la barra superior.

8. Abrir el artículo con la aplicación externa que se desee.
9. Comprobar como es posible leer el artículo.

Se puede afirmar una vez más que el requisito es implementado de manera satisfactoria.

6.2.4. Probar: Será posible la activación de notificaciones para avisar al usuario de aquellas ponencias a las que este esté interesado en asistir.

Este requisito será algo más complejo de comprobar. Esto es porque los Eventos que aparecen en la aplicación, y sobre los que se activan las alertas, son Eventos reales y que se desenvolverán en un futuro. Por ello se insertará un Evento para hoy mismo en la base de datos y se comprobará cómo se activa la alerta y aparece la notificación en la barra superior de notificaciones. Se seguirán los siguientes pasos:

1. Insertar un Evento de ejemplo para el mismo día de la prueba a través de AndroidStudio, con una hora de celebración posterior al momento de la prueba.
2. Acceder a la aplicación.
3. Abrir el menú lateral.
4. Seleccionar la sección Programa.
5. Activar la alerta para el Evento recién creado.
6. Abrir el menú lateral.
7. Seleccionar la sección Mis Alertas.

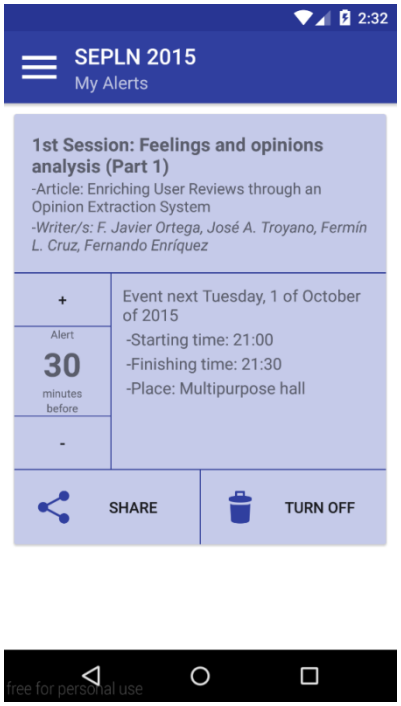

Ilustración 6.9. Sección de Mis Alertas con la Alerta activa.

8. Comprobar como aparece la alerta que se acaba de activar.
9. Aumentar el tiempo de notificación hasta que sea posible (de esta manera habrá que esperar menos).
10. Salir de la aplicación y esperar a que llegue la hora de la notificación.
11. Comprobar que la notificación aparece con la antelación que se había seleccionado.

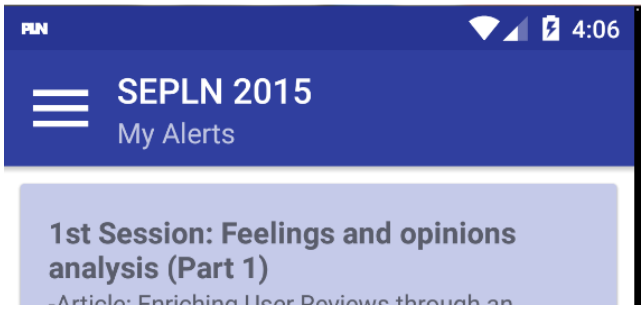

Ilustración 6.10. Captura en la que se puede apreciar la notificación en la barra de acción superior

Se puede afirmar que se cumple el requisito de manera satisfactoria. Ya se puede eliminar este Evento de la base de datos de la aplicación.

6.2.5. Se podrán compartir las diferentes experiencias del congreso a través de diferentes redes sociales.

Hay diferentes secciones en las que la aplicación permite al usuario compartir su experiencia en las redes sociales. Sin embargo, se realizará un único caso de prueba, ya que en todas las secciones esta funcionalidad es implementada de la misma manera. Se llevarán a cabo los siguientes pasos manualmente:

1. Acceder a la aplicación.
2. Abrir el menú lateral.
3. Seleccionar la sección de Artículos.
4. Pulsar sobre un botón “Compartir” de uno de los Artículos que se muestran en la pantalla.
5. El sistema mostrará un diálogo con las aplicaciones de las que dispone en el dispositivo para compartir.

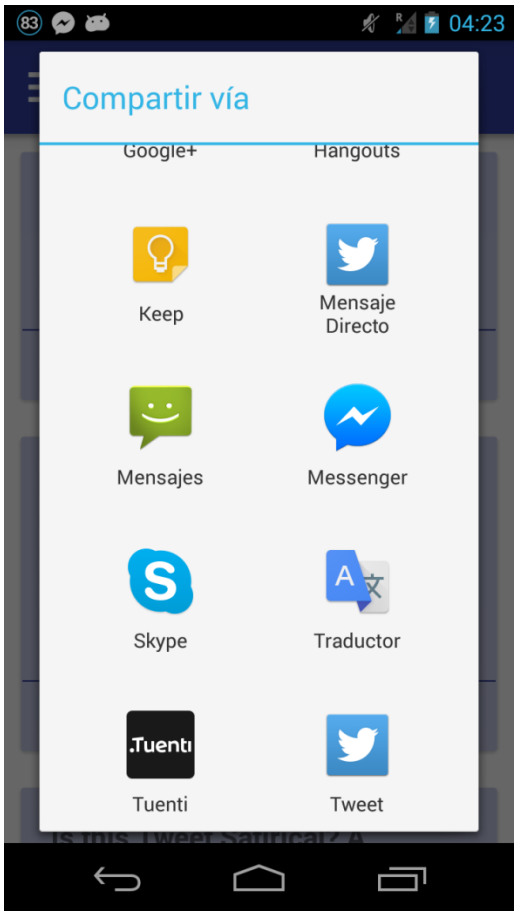

Ilustración 6.11. Pantalla mostrando diálogo Compartir vía

6. Seleccionar una de ellas, se selecciona Twitter en este caso particular.

7. Se abre la aplicación y ofrece la posibilidad de modificar el texto por defecto que en ella aparece.

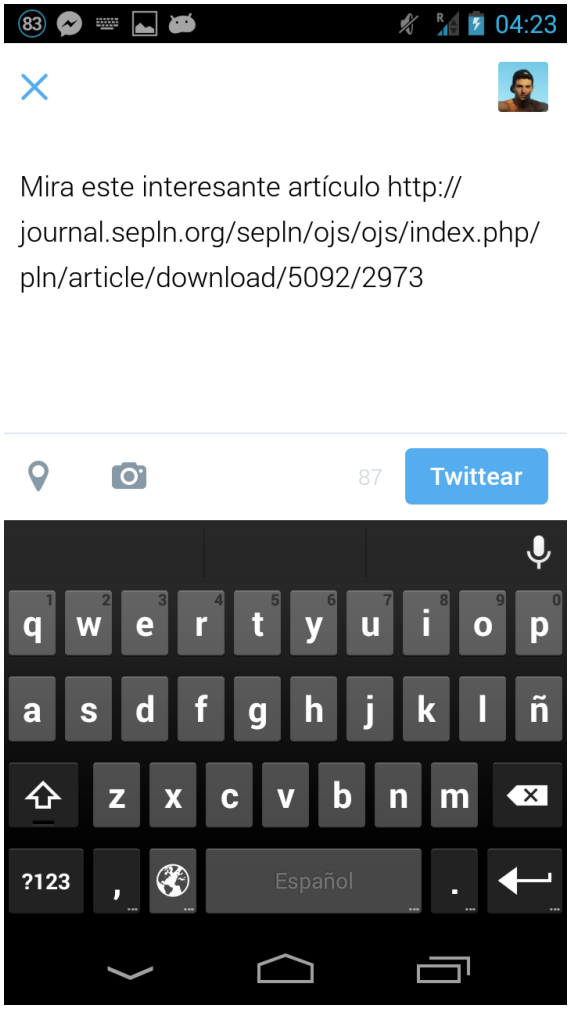

Ilustración 6.12. Pantalla de Twitter con el tweet por defecto

8. Pulsar sobre el botón correspondiente para enviar el tweet.
9. Comprobar que la experiencia ha sido compartida en el timeline del usuario.

   Se puede afirmar que se cumple el requisito de manera satisfactoria.

6.3. Técnicas de prueba específicas para Android

En esta sección se va realizar un repaso por diferentes pruebas y sistemas de pruebas más específicos de Android. Además, para finalizar se expondrá la justificación por la cual se satisfacen todos y cada uno de los requisitos no funcionales expuestos durante el *Capítulo 3: Análisis*.

6.3.1. Pruebas en Android

En esta subsección se va a estudiar el caso particular de las pruebas en Android. Ya que, aunque se describieron anteriormente los tipos de pruebas genéricos para toda aplicación software, existen algunos aspectos que se deben tener en cuenta a la hora de probar un sistema software orientado a Android. Estos aspectos son los siguientes:

- Existen multitud de **dispositivos** diferentes que usan Android, por lo que habrá muchos dispositivos sobre los que probar la aplicación. En este caso concreto se ha probado la aplicación sobre un teléfono móvil Samsung Galaxy Nexus con versión de Android 4.3, sobre un emulador de un Nexus 5 con versión de Android 5.1 y sobre una Tablet (modelo desconocido, china) con una versión de Android 4.0.3. La aplicación funcionó de manera correcta en cada una de las versiones sobre las que se probó.
- Multitud de **formatos** diferentes. En este caso se ha probado sobre dos teléfonos móviles (uno virtual y otro físico) de similar tamaño, alrededor de las 5 pulgadas, y sobre una Tablet de 10 pulgadas. En todos los casos la aplicación funcionó de manera adecuada, aunque sí es verdad que se podría aprovechar mejor la pantalla de la Tablet real.
- **Retrocompatibilidad** hacia atrás. Como ya se comentó anteriormente, una aplicación Android se desarrolla normalmente contra la última API disponible (actualmente la 22) y el desarrollador establece la versión mínima (la API 15 en este caso), y el mismo desarrollador debe encargarse de que todos los elementos implementados funcionen en todas las versiones hasta la mínima establecida. Para ello Google proporciona una librería de compatibilidad, en la que implementa las nuevas funcionalidades de las últimas versiones en versiones anteriores. En este caso se ha probado la app en diferentes versiones y todo funciona como debe funcionar, sea cual sea la versión del sistema operativo.
- Integración con el **hardware**. En el mundo de los dispositivos Android existen multitud de configuraciones hardware diferentes: procesadores mononúcleo, multinucleo, memorias RAM de hasta 3 gigas, etc. Se debe

probar que la aplicación funciona de manera fluida y correctamente tanto en dispositivos con configuraciones hardware tope de gama, cómo en dispositivos con configuraciones hardware más reducidas. Cómo ya se ha comentado la app se ha probado en un móvil físico de gama media actual, la aplicación funciona fluida y de manera notable; un dispositivo virtual de gama superior en el que evidentemente todo iba sobre ruedas; y una Tablet de origen chino, de la cual desconozco el hardware (las pruebas en esta Tablet fueron realizadas en casa de un familiar ya que en mi familia no disponemos de una Tablet, y no se tomaron datos sobre la misma, ni capturas), aunque se puede afirmar que la aplicación funcionaba adecuadamente.

6.3.2. Aspectos a probar en una aplicación Android

Existen una serie de aspectos que en una aplicación móvil, en concreto Android, deben ser probados sí o sí, estos son los siguientes:

- **Interfaz de usuario**: este es un aspecto difícil de probar y bastante relativo. Pero existen unos estándares que se deben seguir. Es importante que la interfaz de usuario de una aplicación móvil sea sencilla y atractiva para el usuario al mismo tiempo. Para probar este apartado se han comprobado que los botones son accionables, la navegación entre las distintas pantallas es adecuada, se ha comprobado que ningún elemento de las diferentes secciones queda fuera del alcance del usuario, se ha comprobado el funcionamiento del scrolling vertical en las diferentes pantallas, la apertura y cierra del menú lateral tanto desde la Toolbar cómo deslizando desde la parte izquierda de la pantalla hacia la derecha, se ha comprobado la navegación de las pestañas donde corresponde, etc. Además, para solucionar el problema al rotar la pantalla (muchos elementos aparecían descolocados) se ha omitido esta acción al usuario, ya que solo se utiliza un terminal con la pantalla en horizontal en un porcentaje de ocasiones mínimo. Por lo tanto se puede afirmar que la interfaz de usuario es sencilla, vistosa, intuitiva y no obstaculiza ninguna acción al usuario.
- **Usabilidad**: para comprobar la usabilidad de un producto la mejor opción es siempre mostrárselo a alguien totalmente ajeno al mismo, y esperar la

sinceridad y crítica constructiva del mismo. En este caso se han utilizado a mis familiares más cercanos como conejillos de indias. Ellos me han ayudado a encontrar algunos fallos en cuanto a usabilidad, por ejemplo la inclusión de botones en las listas en lugar de utilizar la acción al pulsar sobre un ítem de la misma. Además, también mis tutores han probado la aplicación durante el desarrollo y han ido sugiriendo mejoras que se han ido implementando en la medida de lo posible. Finalmente, ha quedado una aplicación bastante usable, cualquier persona ajena a la informática y al Congreso sobre el Procesamiento del Lenguaje Natural podría utilizarla sin problema alguno.

- **Rendimiento**: en cuanto al rendimiento es recomendable comprobar la redundancia de código, los ciclos de la CPU, el consumo de batería, las fugas de memoria, el uso del Wifi, 4G y 3G. Sin embargo no se han realizado pruebas respecto a este apartado, debido a la falta de tiempo para las mismas.
- **Seguridad**: para que la aplicación desarrollada sea segura, es importante comprobar la protección de la información y el mantenimiento de la misma que hace la aplicación, comprobar la confidencialidad, integridad, autentificación, autorización, disponibilidad, etc. En este apartado tampoco se han realizado comprobaciones, ya que la aplicación no trata datos de gran delicadeza.

6.3.3. Cumplimiento de los requisitos no funcionales

Para finalizar con este Capítulo de Pruebas se van a exponer los requisitos no funcionales (extraídos del Capítulo 3: Análisis) para justificar su satisfacción:

1. **Requisitos de usabilidad:**
    - **Interfaz de usuario familiar:** la prueba de usuarios ajenos totalmente al Congreso y al mundo de la informática, y la comprobaciónd e que son totalmente capaces de usar la aplicación justifica la satisfacción de este requisito..
    - **Facilidad de aprendizaje**: al igual que el anterior, con la prueba de usuarios ajenos al Congreso y al mundo de la informática se ha comprobado que estos son capaces de aprender a usar la aplicación practicamente al instante.

- **Tiempo de respuesta**: en este aspecto la aplicación aun puede mejorar a la hora de cargar algunas de las secciones, como ocurre con la sección de Mapa en su primera ejecución. Se podría decir que este requisito está parcialmente satisfecho.

2. **Accesibilidad:** mediante la prueba de la app en diferentes dispositivos, con diferente hardware y software, se ha comprobado que la accesibilidad se cumple de manera muy satisfactoria.
3. **Disponibilidad:** siempre que el dispositivo dispone de acceso a Internet todas las funcionalidades están disponibles, incluso sin acceso la algunas de las funcionalidades pueden seguir realizándose.
4. **Extensible:** la alicación no se construyó tal y como es desde el primer momento. Se realizó una primera versión, y sobre esta se han ido implementando nuevas funcionalidades sin un mayor esfuerzo. En el Anexo A de esta memoria se expondrá todo lo correspondiente a la actualización y mantenimiento de la aplicación.
5. **Licencias software:** se disponia licencia de todas las herramientas empleadas, o estas eran libres. Además los iconos empleados han sido extraídos, en su mayoría, del propio IDE de desarrollo AndroidStudio. A excepción del logo de la SEPLN.
6. **Requisitos económicos:** este requisito es dificil demostrable, aunque se considera evidente que no he dispuesto de recursos económicos para el desarrollo de la aplicación.
7. **Recursos informáticos:** estos, al igual que el anterior también es dificilmente demostrable.

7. CONCLUSIONES

Resumen

En este último capítulo describo principalmente lo que ha significado para mí el proyecto.

7.1. Valoración personal

Llegado a este punto, he completado todas las fases que la ingeniería del software define para el desarrollo de un proyecto informático, y doy por concluido el proyecto, ya que se han alcanzado los objetivos que D. Luis Alfonso Ureña López, Eugenio Martínez Cámara y yo nos marcamos para mi trabajo fin de grado (TFG), cuando hablamos por primera vez sobre la posibilidad de realizar una aplicación orientada a dispositivos móviles para el Congreso anual de la Sociedad del Procesamiento del Lenguaje Natural.

Como ya he mencionado anteriormente, esta era mi primera experiencia en el mundo de la programación para dispositivos móviles y no me podría haber ido mejor. Me he sentido muy bien desarrollando la aplicación y es muy gratificante ver que tu trabajo va a tener resultado, ya que la aplicación tendrá un uso real en el congreso de la SEPLN de este año. Esto solo me motiva más para continuar desarrollando software.

Ya se tuvieron primeros contactos en cuanto a desarrollo de interfaces de usuario en las asignaturas de Interacción Persona-Ordenador, y este es uno de los aspectos más importantes en cuanto al desarrollo de una aplicación móvil. Una aplicación móvil debe disponer de una interfaz muy usable, en la que el usuario tenga acceso a todo en tres toques. Y la verdad que el diseño de interfaces de usuario es uno de los temas que más me llama la atención. Además en este TFG también hay mucho de Ingeniería del software. Ya que una aplicación móvil es un proyecto software más. Me han sido bastante útiles asignaturas como Fundamentos de Ingeniería del Software, Calidad del Software e Ingeniería de Requisitos. También se ha tocado la parte de diseño, implementación y gestión de base de datos, de la que también tuve contacto durante el desarrollo del grado en dos asignaturas. Y, como no, la Programación Orientada a Objetos con Java. En definitiva puedo decir que del TFG, a parte de el aprendizaje para desarrollar aplicaciones orientadas a dispositivos móviles, también me llevo este pequeño resumen de lo aprendido durante el desarrollo del grado.

Como conclusión final me gustaría decir que, aunque he dispuesto de muy poco tiempo para el desarrollo de este TFG (algo más de dos meses), he realizado un gran esfuerzo. Ha sido un verano muy intenso de trabajo pero puedo afirmar que ha merecido la pena. Al fin y al cabo esto solo refleja como ha sido mi paso por el Grado en Ingeniería Informática, han sido cuatro años breves pero intensos. Esta etapa ya llega a su fin y solo puedo tener buenos recuerdos de ella. Espero que esta nueva etapa que comienza ahora sea al menos igual de intensa y apasionante.

Anexo A: Actualización de la aplicación

1. Introducción

Una vez finalizada la aplicación para el Congreso sobre el Procesamiento del Lenguaje Natural, el organismo organizador (la SEPLN) ha decidido utilizar la aplicación para el Congreso del presente año. Por lo que esta estará disponible en la Play Store para su descarga por parte de los asistentes al Congreso.

Si la aplicación tuviera éxito, podría decidirse emplearla de nuevo para el Congreso de un año futuro. Para ello sería necesaria la actualización de los datos de la aplicación.

Por lo tanto en este Anexo se van a exponer los pasos a seguir para actualizar los datos de la aplicación dando por hecho que se ha leído la memoria y se tienen las nociones de cómo funciona el sistema. Además se propondrán algunas mejoras futuras para la aplicación.

2. Actualización de los datos de la aplicación

1. Modificar script de inserción de tuplas: Esto será una tarea más engorrosa que compleja por así decirlo. Encontramos este script en el proyecto de la aplicación en la siguiente ruta: \app\src\main\res\raw con el nombre insertdb.sql. Accedemos a él mediante cualquier editor de texto o el mismo Android Studio y se sustituyen los eventos que ahí aparecen por los eventos nuevos del presente año.

2. Modificar comité, organización y texto de introducción: Para esto se debe acceder al recurso strings del proyecto. Este se encuentra en el proyecto en la ruta: \app\src\main\res\values con el nombre strings.xml. Accedemos desde cualquier editor de texto o desde el mismo Android Studio y modificamos los textos correspondientes sustituyendo los actuales por los correspondientes al nuevo congreso.

```
<!-- Strings seccion inicio!-->
<string name="txtbienvenida">
    The SEPLN conference, is the best conference in <b>Natural Language Processing</b>.
    for the Spanish language. Here, national and international specialists, treat the
    advances in these areas, covering the new chalenges of Human Language Technologies
    and their applications.\n
    \n
    The 31rd edition of the SEPLN will have place in Alicante in September 16th, 17th
    and 18th, 2015.\n
    \n
    <b>IMPORTANT DATES</b>\n
    The conference will be held from <b>Wednesday 16 September to Friday 18 September, 2015</b>.\n
    \n
    -Early registration deadline: <i>July 1 2015</i>\n
    -Deadline for paper submission: <i>March 27 2015</i>\n
    -Notifications: <i>May 1 2015</i>\n
    -Camera Ready: <i>May 15 2015</i>\n
    -Conference: <i>September 16-18 2015</i>\n
    -Workshops and Tutorials: <i>September 15 2015</i>\n
    \n
    <b>TOURIST INFO</b>\n
    Practical information about Alicante, getting Alicante, city walks, brochures,
    maps, apps... http://www.alicanteturismo.com/?lang=en\n
    \n
    <b>REGISTRATION AND ACCOMODATION</b>\n
    Please use our registration and accomodation form:
    http://www.npmundo.com/congreso2/mod/paso_10.asp?id_congreso=32\n
    \n
    <b>CONTACT</b>\n
    You can contact us via the contact form (http://gplsi.dlsi.ua.es/sepln15/en/contact)
    and we will resolve your doubts as soon as possible.
</string>
```

Ilustración 0.1. String Introducción

```
<!-- Strings seccion comite!-->
<string name="txtorganizacion">
    GPLSI (Group of Language Processing and Information Systems). University of Alicante\n
    \n
    <b>PRESIDENT</b>\n
    -Patricio Martínez-Barco\n
    \n
    <b>TECHNICAL SECRETARY</b>\n
    -Rafael Muñoz\n
    -Andrés Montoyo\n
    -Estela Saquete\n
    \n
    <b>SCHEDULE AND SOCIAL EVENTS</b>\n
    -Paloma Moreda\n
    -David Tomás\n
    -Lea Canales\n
    \n
    <b>SCIENTIFIC SECRETARY</b>\n
    -Borja Navarro\n
    -Sonia Vazquez\n
    -Maite Romá\n
    \n
    <b>MEDIA</b>\n
    -José Manuel Gómez\n
    -Armando Suárez\n
    -Javier Fernández\n
    \n
    <b>SPONSORS</b>\n
    -Yoan Gutiérrez\n
    -Fernando Peregrino\n
    \n
    <b>WORKSHOP SESSION</b>\n
    -David Tomás\n
    \n
    <b>DOCTORAL SYMPOSIUM SESSION</b>\n
    -Elena Lloret\n
    \n
    <b>DEMONSTRATION SESSION</b>\n
    -Yoan Gutiérrez\n
    \n
    <b>PROJECTS SESSION</b>\n
    -Paloma Moreda\n
```

Ilustración 0.2. String Organización

```
<!-- Strings seccion organizacion!-->
<string name="txtcomite">
    <b>PRESIDENT</b>\n
    -Patricio Martínez-Barco. <i>Universidad de Alicante (Spain)</i>\n
    \n
    <b>MEMBERS</b>\n
    -Manuel de Buenaga. <i>Universidad Europea de Madrid (Spain)</i>\n
    -Sylviane Cardey-Greenfield. <i>Centre de recherche en linguistique et traitement a
    -Irene Castellón. <i>Universidad de Barcelona (Spain)</i>\n
    -Arantza Díaz de Ilarraza. <i>Universidad del País Vasco (Spain)</i>\n
    -Antonio Ferrández. <i>Universidad de Alicante (Spain)</i>\n
    -Alexander Gelbukh. <i>Instituto Politécnico Nacional (Mexico)</i>\n
    -Koldo Gojenola. <i>Universidad del País Vasco (Spain)</i>\n
    -Xavier Gómez Guinovart. <i>Universidad de Vigo (Spain)</i>\n
    -José Miguel Goñi. <i>Universidad Politécnica de Madrid (Spain)</i>\n
    -Bernardo Magnini. <i>Fondazione Bruno Kessler (Italy)</i>\n
    -Nuno J. Mamede. <i>Computadores Investigação e Desenvolvimento em Lisboa (Portugal
    -M. Antonia Martí Antonín. <i>Universidad de Barcelona (Spain)</i>\n
    -M. Teresa Martín Valdivia. <i>Universidad de Jaén (Spain)</i>\n
    -Raquel Martínez. <i>Universidad Nacional de Educación a Distancia (Spain)</i>\n
    -Leonel Ruiz Miyares. <i>Centro de Linguistica Aplicada de Santiago de Cuba (Cuba)<
    -Ruslan Mitkov. <i>University of Wolverhampton (United Kingdom)</i>\n
    -Manuel Montes y Gómez. <i>nstituto Nacional de Astrofísica, Óptica y Electrónica (
    -Lidia Moreno. <i>Universidad Politécnica de Valencia (Spain)</i>\n
    -Lluís Padro. <i>Universidad Politécnica de Cataluña (Spain)</i>\n
    -Manuel Palomar. <i>Universidad de Alicante (Spain)</i>\n
    -Ferrán Pla. <i>Universidad Politécnica de Valencia (Spain)</i>\n
    -German Rigau. <i>Universidad del País Vasco (Spain)</i>\n
    -Horacio Rodriguez. <i>Universidad Politécnica de Cataluña (Spain)</i>\n
    -Kepa Sarasola. <i>Universidad del País Vasco (Spain)</i>\n
    -Emilio Sanchís. <i>Universidad Politécnica de Valencia (Spain)</i>\n
    -Thamar Solorio. <i>University of Houston (U.S.A.)</i>\n
    -Maite Taboada. <i>Simon Fraser University (Canada)</i>\n
    -Juan-Manuel Torres-Moreno. <i>Laboratoire Informatique d'Avignon / Université d'Avi
    -José Antonio Troyano Jiménez. <i>Universidad de Sevilla (Spain)</i>\n
    -L. Alfonso Ureña López. <i>Universidad de Jaén (Spain)</i>\n
    -Rafael Valencia. <i>Garcia Universidad de Murcia (Spain)</i>\n
    -Felisa Verdejo. <i>Universidad Nacional de Educación a Distancia (Spain)</i>\n
    -Manuel Vilares. <i>Universidad de la Coruña (Spain)</i>\n
    -Luis Villaseñor-Pineda. <i>Instituto Nacional de Astrofísica, Óptica y Electrónica
</string>
```

Ilustración 0.3. String Comité

3. Actualizar base de datos: Como sabemos, para que el sistema vuelva a crear la base de datos es necesario cambiar su nombre. El nombre de la base de datos también se encuentra en el fichero strings.xml en la ruta: \app\src\main\res\values. Su identificador es “DataBaseName”.

```
<string name="seccion6">Organization</string>
<string name="seccion7">Scientific Committee</stri
<string name="salir">Exit</string>
<string name="DataBaseName">SEPLNdb7</string>

<!-- Strings seccion inicio!-->
<string name="txtbienvenida">
    The SEPLN conference, is the best conference i
    for the Spanish language. Here, national and i
```

Ilustración 0.4. Nombre de la base de datos

4. Modificar título de la aplicación: Este se encuentra también en el fichero strings.xml en la ruta: \app\src\main\res\values. Su identificador es “AppName”.

```
<!-- Strings generales!-->
<string name="app_name">SEPLN 2015</string>
<string name="seccion1">Introduction</string>
<string name="seccion2">Articles</string>
```

Ilustración 0.5. Nombre de la aplicación

5. Si se han realizado los cambios anteriores siguiendo la sintaxis correcta bastaría con reconstruir el proyecto y generar un nuevo .apk desde AndroidStudio.
6. Fin.

3. Posibles mejoras futuras de la aplicación

Como sección final de este anexo se van a realizar dos propuestas de mejoras de la aplicación para el futuro. He de decir que he tenido estas mejoras en mente durante todo el desarrollo de la aplicación. Sin embargo me ha sido imposible la investigación e implementación de las mismas, ya que disponía de un periodo de tiempo muy limitado para el desarrollo de la solución al problema que este TFG planteaba. Al fin y al cabo eso es la ingeniería: buscar la mejor solución a un problema con los medios de los que disponemos, y pienso que eso lo he conseguido

Estas mejoras dos propuestas son:

- **Carga de datos desde un repositorio web**: actualmente, como se explica en la memoria detalladamente, la aplicación consta de una base de datos con toda la información relativa al congreso, por lo que cualquier actualización de la misma (como es el otro objetivo ya descrito de este anexo) requiere la modificación manual de todas las inserciones en la base de datos.

  Con esta actualización se conseguirá automatizar todo el proceso y de un año para otro solo habría que cambiar la dirección del repositorio web.
- **Mostrar ruta en sección mapa**: actualmente, para ver la ruta desde la ubicación del dispositivo hasta un punto de interés se debe hacer uso de una aplicación externa. La mejora consistiría en cargar la ruta en el propio mapa que se muestra en la sección mapa de la aplicación.

Con esta actualización la aplicación mejoraría bastante esteticamente aunque quizás se cargaría más de procesamiento la aplicación y con ello, probablemente se entorpecería la fluidez de la interfaz.

Bibliografía

Tutorial de programación en Android: http://www.sgoliver.net/blog/curso-de-programacion-android/indice-de-contenidos/

Foro de resolución de problemas: http://stackoverflow.com/

Información teórica: https://es.wikipedia.org/