Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Diseño de APP Android para seguimiento de la web de Antiguos Alumnos de la ETSI

Autor: Matías Huéscar Núñez
Tutor: Rafael Estepa Alonso

Departamento de Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Diseño de APP Android para seguimiento de la web de Antiguos Alumnos de la ETSI

Autor:
Matías Huéscar Núñez

Tutor:
Rafael Estepa Alonso
Profesor titular

Dep. de Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Proyecto Fin de Carrera: Diseño de APP Android para seguimiento de la web de Antiguos Alumnos de la ETSI

Autor: Matías Huéscar Núñez

Tutor: Rafael Estepa Alonso

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

A mis padres, a mi familia y amigos, que me han dado cuanto tengo y soy. Al sistema público Universitario, para que siga dando tantas oportunidades como a mí me dio.

Matías Húescar Núñez

Sevilla, 2016

Resumen

En el presente proyecto se lleva a cabo el desarrollo de una aplicación móvil basada en Android para la web de la Asociación de Antiguos Alumnos de la ETSI. La intención de aplicación es la de suministrar contenido RSS de la web en la aplicación, permitir a los usuarios descargarse ofertas de empleo en formato PDF e inscribirse a través de un sencillo formulario a las distintas actividades organizadas por la Asociación.

Abstract

This project carries out the development of a mobile application based on Android for the Asociación de Antiguos Alumno's web of ETSI. The application's goal is to provide RSS content from the Asociación's web, allowing users to download job offers in PDF and register through a simple form to differents activities organized by the Association.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCION Y ESTUDIO DE SITUACIÓN

Como cualquier problema de ingeniería, este proyecto nace de la necesidad dar respuesta a una problemática cotidiana y sufrida personalmente, que aún siendo muy visible, nadie había intentado remediar anterioremente en esta asociación, hablamos del mailing y la información web.

1.1 La Asociación Antiguos Alumnos

La Asociación de Antiguos Alumnos (en adelante la AAA) de la Escuela Superior de Ingenieros de Sevilla ANTONIO DE ULLOA da sus primeros pasos a comienzos de los 90, cuando se celebra el vigésimo quinto aniversario de historia de la Escuela, y se constituye formalmente en diciembre de 1994. A día de hoy la Asociación sigue activa, organizando multitud de actividades para sus asociados, publicando noticias e incluso ofertas de empleo.

A partir de entonces, la asociación se ha ido comunicando con sus asociados de diversas formas acordes a los tiempos que ha ido viviendo. En estos momentos en los que el paradigma de la comunicación ha cambiado y en el que hasta una web puede quedarse obsoleta, es el momento de modernizar a la Asociación de Antiguos Alumnos para dotarla de herramientas acordes a sus tiempos a la hora de comunicarse con sus inscritos.

1.2 Motivación

La Asociación encuentra varios problemas a la hora del acceso a la información que publica. Dividiremos los distintos asuntos en los siguientes apartados.

1.2.1 El mailing

La Asociación envía a sus asociados una media de tres emails semanales, además, estos emails no se distribuyen a lo largo de los días de la semana sino que son enviados conjuntamente un mismo día. Estos caracen de formato de identidad visual y llegan a colapsar las bandejas de entrada de los asociados creando una sensación de SPAM perjudicial para la entidad.

No es posible usar diversas listas de suscripción para cada tipo de información, puesto que la gestión de las listas depende de Universidad de Sevilla. Por otro lado, el uso de listas de distribución también tiene el inconveniente de los filtros antiSPAM y la gestión de las listas de asociados, lo que colleva la tramitacón de sus altas, bajas, etc. Todo ello contribuye a empeorar la experiencia del usuario con la Asociación al carecer éste de medios modernos e inmediatos para solucionar los problemas con el mailing.

1.2.2 La web

La web de la AAA está basada en Drupal 7.0, una version de 2011 del famoso CMS. La web a día de hoy no se encuentra adaptada para la visualización en dispositivos móviles por lo que no se hace atractiva para la consulta de su información. En la web se publican distintos tipos de información (anuncios, ofertas, eventos, etc) donde el impacto de ésta en los usuarios es muy bajo, ya que el número de visitas recibidas en mínimo.

Un reciente estudio de 2015 de IAB Spain [1] revela que el 85% de las personas que accede a internet lo hacen a través de un smartphone. Este estudio deja en mala posición a la AAA y aporta motivos para entender el bajo impacto de la web entre sus asociados.

1.3 Estructura de la web

1.3.1 Secciones

La web se compone de ocho secciones: inicio, actividades, noticias, empleo, ingenio, galerías, vídeos y convocatorias.

Vemos como la ilustración 1 muestra que existen secciones como galerías y vídeos que podrían ir juntas, mientras que inicio simplemente muestra una previsualización de las secciones noticias, actividades, convocatorias y empleo, éstas dos últimas unidas en una misma tabla. Se entiende por tanto que las secciones más interesantes para los asociados son éstas y en la solución que se plantea más adelante, serán las seleccionadas para informar a los asociados.

La sección de ingenio contiene un índice de enlaces a distintos PDF para la consulta de la revista del mismo nombre, publicada anualmente por la AAA. Sería interesante que en la solución a plantear se pudiera crear una sección de consulta de los mismos.

1.3.2 Sección a usuarios

Se dispone de un acceso a usuarios con nombre de usuario y clave pero existe una problemática asociada a esta sección y es que la gran mayoría de los asociados no recuerdan sus contraseñas y por tanto esta sección raramente es usada.

El Vicepresidente de la AAA, D. Rafael Estepa Alonso, señalaba la importancia de crear una solución sencilla en la que se elimine la necesidad de hacer login para acceder a la información debido a la cantidad de problemas de pérdida de contraseñas que se producen en la Asociación, haciendo que no merezca la pena establecer un servicio de este tipo.

1.3.3 Formulario de inscripción

En una de las columnas de la web encontramos un enlace a un formulario de inscripción. Inicialmente sería una buena idea incluir un formulario de inscripción dentro de la solución a desarrollar pero el formulario en cuestión incluye autorizaciones de cargo a cuenta y firma del asociado, por lo que en un principio no se implementará.

1.4 Objeto del Proyecto

El proyecto tiene como objetivo la creación de una aplicación móvil que se adapte a las necesidades de la AAA siendo capaz de ofrecer una solución actual y sencilla a los asociados para que puedan realizar diferentes funciones de manera cómoda e inmediata, mejorando así su experiencia con la Asociación.

Por otro lado, se llevará a cabo el desarrollo de este proyecto de manera que se interfiera lo mínimo posible del lado del servidor, es decir, modificando lo mínimo la web de la Asociación alojada en servidores de la US.

La aplicación soportará las siguientes funcionalidades:

1. Lectura de noticias y otras publicaciones
2. División de diferentes apartados por temáticas
3. Descarga de documentos PDF publicados
4. Inscripción a eventos de la AAA
5. Visita a la web desde la aplicación
6. Contacto con la AAA
7. Información de la AAA.

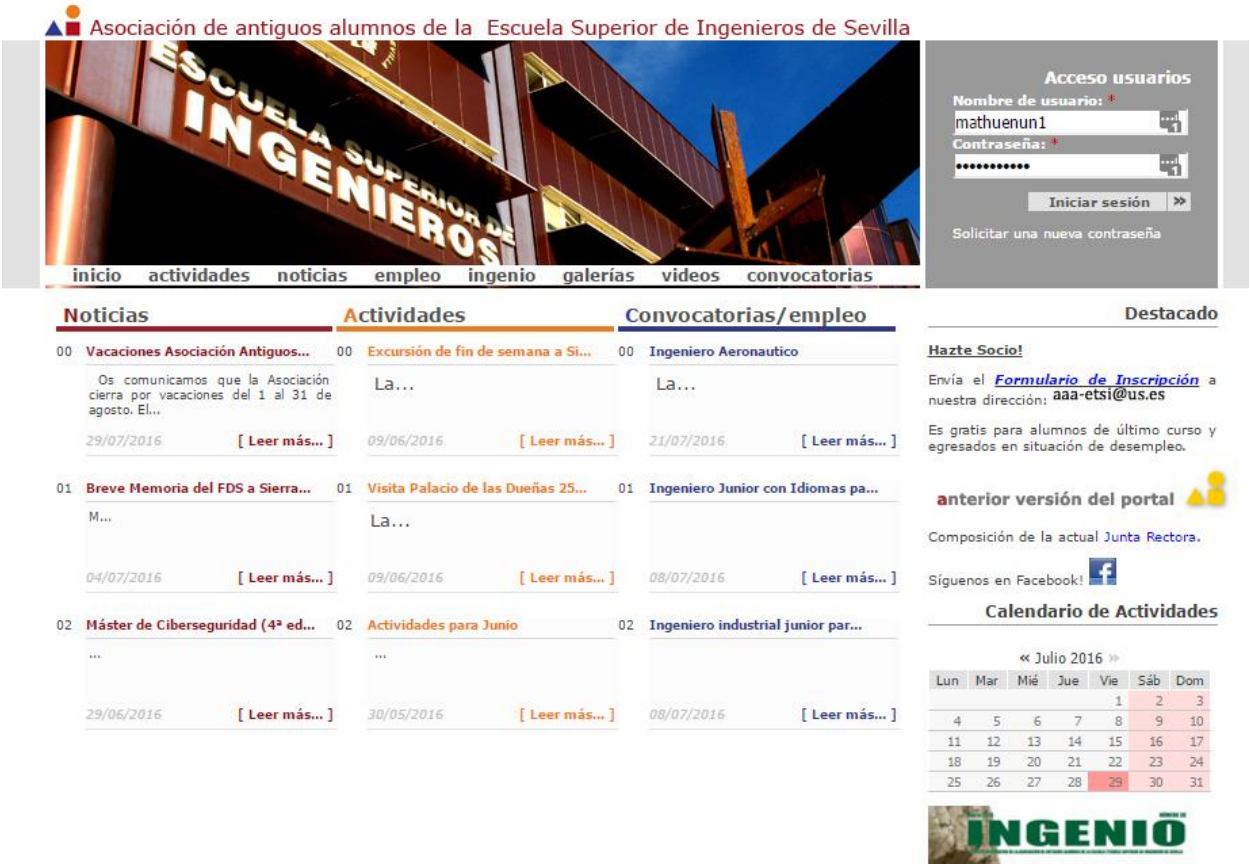

Ilustración 1 – Web AAA

2 ALTERNATIVAS PARA LA IMPLEMENTACIÓN

Para la realización de este proyecto, analizaremos las distintas soluciones posibles a llevar a cabo y se justificarán las opciones tomadas en cada caso. Tendrán prioridad aquellas soluciones que sean capaz de satisfacer más ampliamente las necesidades del proyecto interviniendo lo menos posible del lado del servidor.

2.1 Sistema operativo

En la actualidad nos encontramos con cierta diversidad de sistemas operativos móviles en el mercado, cada uno con niveles de implantación y madurez diferentes. Según el último estudio de Gartner [2]

Worldwide Smartphone Sales to End Users by Operating System in 1Q16 (Thousands of Units)

Operating System	1Q16 Units	1Q16 Market Share (%)	1Q15 Units	1Q15 Market Share (%)
Android	293,771.2	84.1	264,941.9	78.8
iOS	51,629.5	14.8	60,177.2	17.9
Windows	2,399.7	0.7	8,270.8	2.5
Blackberry	659.9	0.2	1,325.4	0.4
Others	791.1	0.2	1,582.5	0.5
Total	**349,251.4**	**100.0**	**336,297.8**	**100.0**

Tabla 1: Venta mundial de smatphones a usuarios finales por SO

El SO más implantado a nivel mundial es con diferencia Android con un 84.1% del mercado, además, según apunta wikipedia [3], en ciertos países como España las diferencias son más signfiicativas, donde Android tiene el 90,8% de la cuota de mercado.

Con intención de ser lo más eficientes posible, se plantea pues el desarrollo de la aplicación móvil en este SO, Android, que nos ofrece la capacidad de abarcar un mercado y un público más extenso programando en su plataforma.

2.1.1 Otras opciones de desarrollo

Se tiene constancia de la existencia de frameworks basados en tecnologías web que brindan la capacidad de portar el código a los principales sistemas operativos móviles actuales. El framework más conocido y estable a

día de hoy es IONIC [4], el cual se encuentra en una fase de desarrollo bastante avanzada y dispone de una gran comunidad de apoyo.

Si bien este framework ofrece la capacidad de abarcar más SO en un menor esfuerzo, por otro lado existen una serie de handicaps que hacen de este tipo de aplicaciones una experiencia pobre. Problemas como incompabilidades con según que webviews haya instalado en los sistemas, rendimiento muy bajo o no disiponibilidad de uso de todas las APIs nativas en el sistema, hacen que se descarte esta opción en pos de una experiencia usuario más fluida y con mayores capacidades.

2.2 Android

Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tablets o tabléfonos; y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, la compró. Android fue presentado en 2007 junto 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. [5]

Ilustración 2 – Android Logo

2.2.1 Entorno de programación Android

El Desarrollo de Programas para Android se hace habitualmente con el lenguaje de programación similar a Java y el conjunto de herramientas de desarrollo SDK. La plataforma integral de desarrollo (IDE) soportada oficialmente es Android Studio junto con el complemento ADT (Android Development Tools plugin). Además, los programadores pueden usar un editor de texto para escribir ficheros Java y XML y utilizar comandos en un terminal (se necesitan los paquetes JDK, Java Development Kit y Apache Ant) para crear y depurar aplicaciones, así como controlar dispositivos Android que estén conectados (es decir, reiniciarlos, instalar aplicaciones en remoto, etc.). Por último Android Studio ofrece funcionalidades de emulación de dispositivos para el testeo de aplicaciones. [6]

Ilustración 3 – Android Studio

2.2.2 Versiones de Android, el problema de la fragmentación y la compatibilidad

Las versiones de Android reciben, en inglés, el nombre de diferentes postres o dulces. En cada versión el postre o dulce elegido empieza por una letra distinta, conforme a un orden alfabético. Desde se lanzó "Apple Pie", la primera versión de éste SO, se han publicado ya hasta 14 versiones diferentes llegando a la más reciente publicada hace semanas: Android 7.0 Nougat.

Se ha elegido Android como plataforma de desarrollo por su posición dominante en el mercado, pero elegir el sistema implica elegir también en qué versiones de éste se va a programar y con cuántas versiones anteiores nuestra aplicación va a ser compatible.

Android comienza a madurar y a establecerse como SO potente a partir de la versión 4.1 (Jelly Bean) que se corresponde con la API 16 del sistema. Se ha elegido esta versión del 2012 como versión mínima compatible con nuestra aplicación para su correcto y total funcionamiento.

Al elegir la versión mínima compatible en la creación del proyecto en Android Studio, éste nos dice qué porcentaje de móviles con Android será compatible con nuestra aplicación, en nuestro caso hablamos del **95,2%.**

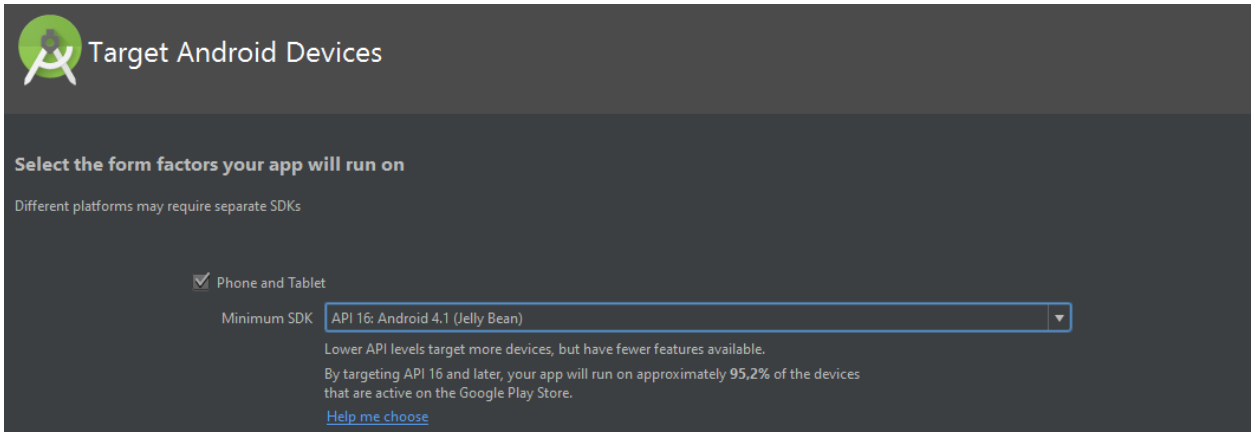

Ilustración 4 – Porcentaje de compatibilidad con dispositivos.

2.3 Obtención de la información para la aplicación

La principal funcionalidad de la aplicación móvil será la de dar acceso a las diferentes noticias, eventos y ofertas publicadas en al web de la AAA. En un primer momento se podría pensar en la necesidad de crear un backend donde la persona responsable de adminsitrar contenidos pudiera volcar la información. Esta solución no es práctica debido a que implica un trabajo por duplicado, cargando contenido en el backend de la aplicación y en la web de la Asociación.

Por otro lado, conocemos que todo CMS tiene su servicio de RSS por defecto, el cual publica el contenido de la web indicado en un archivo XML legible desde una URL asociada al webste. Por tanto, usaremos este método de fuente web (web feed) en formato RSS para tomar información en bruto de la web, o más bien su archivo XML, simplificando así la labor del responsable y ajustándonos a los objetivos de realizar los mínimos cambios posibles del lado del servidor.

2.3.1 RSS

RSS son las siglas de Really Simple Syndication, un formato XML para sindicar o compartir contenido en la web. Se utiliza para difundir información actualizada frecuentemente a usuarios que se han suscrito a la fuente de contenidos. El formato permite distribuir contenidos sin necesidad de un navegador, utilizando un software diseñado para leer estos contenidos RSS. RSS es parte de la familia de los formatos XML, desarrollado específicamente para todo tipo de sitios que se actualicen con frecuencia y por medio del cual se puede compartir la información y usarla en otros sitios web o programas. A esto se le conoce como redifusión web o sindicación web. [7]

El estándar RSS 2.0 viene dado del a siguiente forma:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
 <title>Título del RSS</title>
 <description>Descripción del RSS</description>
 <link>http://www.sitiodelquesedeseapublicar.com/main.html</link>
 <lastBuildDate>Mon, 06 Jan 2013 00:01:00 +0000 </lastBuildDate>
 <pubDate>Mon, 06 Jan 2013 16:20:00 +0000 </pubDate>
 <ttl>1800</ttl>

 <item>
  <title>Entrada dentro del RSS</title>
  <description>Descripción de la entrada</description>
  <link>http://www.sitiodelquesedeseapublicar.com/enero-2013.html</link>
  <guid>clave única</guid>
  <pubDate>Mon, 06 Jan 2013 17:20:00 +0000 </pubDate>
 </item>

</channel>
</rss>
```

Tal y como vemos en el anterior código, el estándar tiene como etiqueta root la versión de XML que se está usando y el tipo de codificación, seguido de la etiqueta RSS que encierra el resto de la información. En primer lugar, nos encontramos con la etiqueta channel, la cual nos indica el canal principal de difusión del que vamos a tomar información, seguidamente vemos etiquetas hijas de channel como title, description, link, lastBuildDate,

pubDate y ttl que nos dan información a cerca del propio channel. A continuación, y de manera recurrente, vendrá la etiqueta ítem, esta etiqueta aparecerá tantas veces como entradas exista en el channel y contendrá información de cada entrada como título, descripción, enlace a la noticia, etc. Así, si leyésemos un feed de noticias sobre deportes, channel sería deporte y los ítems serían las diferentes noticias de deportes que hubiese dentro de ese canal.

Por tanto, encontramos a través de este sistema una manera sencilla de obtener una fuente de datos, diferenciarlos por etiquetas y así clasificarlos para la aplicación móvil.

2.4 Almacenamiento de las noticias

Una vez se realiza una petición al servidor para obtener el RSS de la web de la Asociación, es necesario almacenar las entradas recibidas en una pequeña base de datos de la que podamos leer mientras se ejecuta nuestra aplicación. De otro modo, no tendría sentido que la aplicación estuviese constantemente realizando peticiones de contenido pues haría que bajase el rendimiento de ésta y crearía un consumo de datos más alto.

Necesitamos para ello un motor de base de datos sencillo y ligero, que no haga nuestra aplicación pesada, ya que simplemente vamos a gestionar entradas de un archivo XML. Además, buscamos una base de datos que disponga de librerías en Android. Esa base de datos es SQLite.

2.4.1 SQLite

Es un ligero motor de bases de datos de código abierto, que se caracteriza por mantener el almacenamiento de información persistente de forma sencilla. A diferencia de otros Sistemas gestores de bases de datos como MySQL, SQL Server y Oracle DB, SQLite tiene las siguientes ventajas:

- No requiere el soporte de un servidor: SQLite no ejecuta un proceso para administrar la información, si no que implementa un conjunto de librerías encargadas de la gestión.
- No necesita configuración: Libera al programador de todo tipo de configuraciones de puertos, tamaños, ubicaciones, etc.
- Usa un archivo para el esquema: Crea un archivo para el esquema completo de una base de datos, lo que permite ahorrarse preocupaciones de seguridad, ya que los datos de las aplicaciones Android no pueden ser accedidos por contextos externos.
- Es de Código Abierto: Esta disponible al dominio público de los desarrolladores al igual que sus archivos de compilación e instrucciones de escalabilidad.

Es por eso que SQLite es una tecnología cómoda para los dispositivos móviles. Su simplicidad, rapidez y usabilidad permiten un desarrollo muy amigable. [8]

2.5 Inscripción a eventos

Se necesita implantar un sistema por el cual el usuario pueda inscribirse a un evento de una manera cómoda y sencilla a través de un simple formulario, sin la necesidad de registros o cuentas de usuario que tantos problemas han traído a la AAA. La inscripción, según se acuerda con el tutor de este proyecto, debería llegar en forma de email al correo de la Asociación para su gestión.

2.5.1 A través de un servidor

En un primer momento se piensa en la posibilidad de crear un servidor externo, al cual mediante una petición HTTP POST se envíe un archivo tipo JSON el cual trate y una vez procesado envíe a la AAA. El servidor sería un JSP por la sencillez del código que requiere y por los conocimientos previos del autor.

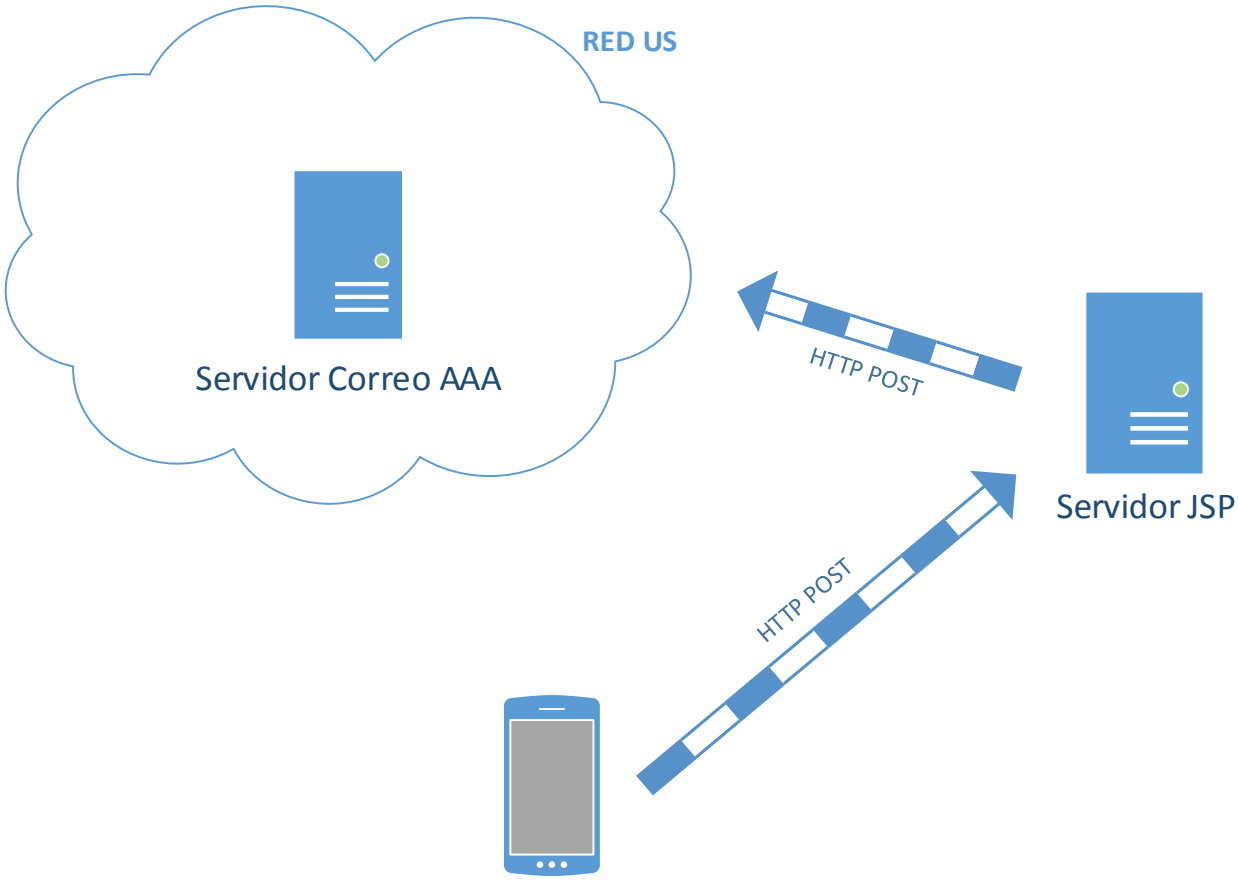

Ilustración 5 – Envío de datos a través de servidor JSP

Este método, si bien es el más transparente y cómodo para el usuario, implica una serie de problemas:

- La información que se envía para la inscripción contiene información sensible del usuario, como nombre y DNI, sería necesario por tanto encriptar la información para una transmisión segura.
- Se rompe con la regla de intereferir lo mínimo posible del lado del servidor. La localización del JSP sería un problema ya que la US debería dar permisos para la instalación de dicho servicio, con los consecuentes trámites burocráticos.
- El servidor debería ser lo suficientemente seguro para que nadie que descompile la aplicación y pudiera obtener información suficiente para hackearlo y usarlo a voluntad propia.

2.5.2 Usando la propia mensajería del usuario

Todo usuario Android que haya descargado la aplicación ha debido acceder previamente a la Google Play Store lo cual implica tener una cuenta de correo asociada al teléfono y activa. Con ello nos estamos asegurando de que cualquier usuario podrá ser capaz de enviar un email.

Para evitar los problemas anteriores, parece adecuado por tanto, gererarle cómodamente al usuario un email que incluya los datos que éste ha introducido previamente en el formulario, destinatario, asunto y descripción, teniendo solamente que presionar en enviar para completar la inscripción. Por la sencillez de la solución y la poca injerencia en los sistemas de la US, se eligirá este método como el más adecuado.

2.6 Interfaz usuario

La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar, aunque en el ámbito de la informática es preferible referirse a que suelen ser "amigables e intuitivos" porque es complejo y subjetivo decir "fácil". [9]

Según los estudios realizados por Jakob Nielsen [10], una interfaz de usuario que sea usable debe ser una interfaz que cumpla con los siguientes factores:

- Facilidad de aprendizaje - El uso de una interfaz debe ser intuitivo, una persona de entre las que conforman su público meta debe ser capaz de entender su funcionamiento sin la necesidad de un manual.
- Eficiencia - El flujo de acciones debe ser corto y veloz, no requiriendo muchas etapas o mucho tiempo por parte del usuario, mas sin embargo entregando los resultados esperados.
- Sin requisición de memoria - El usuario no debe necesitar de su capacidad de memorización. A pesar de haber dejado de utilizar la aplicación por un tiempo, es importante que logre llevar a cabo las tareas que requiere sin mayor problema o consulta adicional.
- Índice de error - La aplicación debe estar preparada para recuperarse en caso de errores, generar la menor cantidad posible de los mismos y ofrecer alternativas de corrección para el usuario en caso de que se presenten.
- Experiencia satisfactoria - El usuario debe sentirse victorioso tras haber llevado a cabo una tarea con la ayuda de la aplicación, es decir, debe sentir la satisfacción de haber logrado la meta que se proponía.

Además, es importante acompañar la aplicación de un apartado estético atractivo y actual, acompañado de transiciones amigables y que hagan la experiencia más fluida y natural. Google pensó en ello con Android y publicó unas reglas de diseño llamadas Material Design.

2.6.1 Material Design

Material design es una normativa de diseño enfocado en la visualización del sistema operativo Android, además en la web y en cualquier plataforma. Material Design tiene superficies físicas y bordes, las escenas y sombras proporcionan significado sobre lo que se puede tocar y cómo se va a mover" [11]. Todo está conceptualmente entendido como capas que se superponen, animadas y unidas entre sí para crear una sensación intuitiva y elegante de la interfaz.

La aplicación que se desarrollará seguirá las líneas de diseño de Material Design para aprovechar todo su potencial y cubrir con estas línas de diseño, las carencias de imágenes y falta de contenido que proporciona el RSS de la AAA.

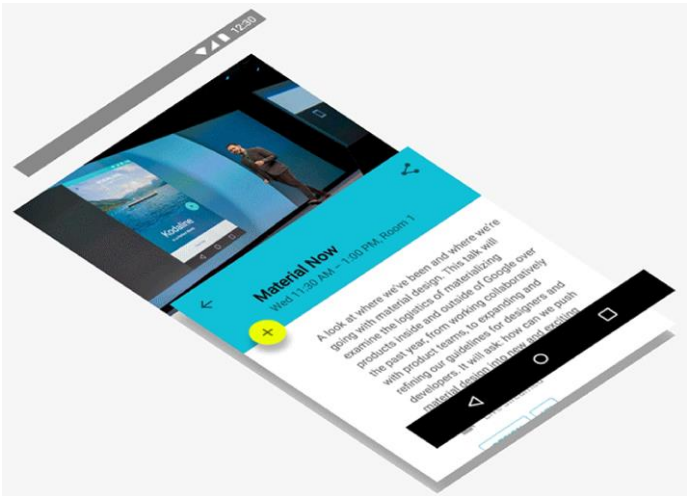

Ilustración 6 – Concepto Material Design

3 DESARROLLO DE LA APLICACIÓN

Este apartado se centrará principalmente en describir en detalle el funcionamiento de cada una de las partes de la aplicación, ver cómo se han llevado a cabo y por qué. Para un buen entendimiento de las razones y procesos, es necesario tener conocimientos básicos de programación Android, por lo que se comenzará la la sección con algunos conceptos clave de este SO.

3.1 Modelo de vistas de la aplicación

Antes de comenzar con el desarrollo y entrar en detalle, se definirá en este apartado cual va a ser el modelo de vistas a programar y cómo va a ser su desarrollo lógico.

Siguiendo las reglas de Nielsen para la usabildiad de interfaces, se plantea en la siguiente ilustración, un modelo de vistas sencillo, en el que para realizar las principales funciones se lleve a cabo el mínimo número de pulsaciones posibles y haciendo que los pasos entre vistas sean intuitivos.

Como podemos ver en la ilustración 7, partimos de una pantalla inicial de arranque que muestra el logo de la Asociación y de inmediato nos encontraremos con un paginador de tres vistas asociadas en los que mediante desplazamientos horizontales podremos movernos entre los distintos apartados: noticias, eventos y empleo. Cada una de estas vistas dispondrá de una vista "scrollable" en el que se mostrarán tarjetas cuyo contenido serán las noticias correspondientes a cada sección. Mediante el click en alguna de esas tarjetas, se accede a pantalla completa de la noticia seleccionada donde podremos leer más texto de la noticia y visualizar opciones. Una última vista cabe esperar si presionamos sobre el botón "visitar web", dentro de la noticia, que nos llevará a un Webview donde podremos visualizar la noticia original en el portal web de la AAA.

Como excepción, la página presentación presenta en la toolbar de la aplicación, unos pequeños tres puntos a modo de menú donde el usuario puede acceder al apartado "quiénes somos" donde puede consultar más información sobre la AAA.

La aplicación por tanto solo tiene avances y retrocesos entre vistas en un sentido y en cada avance que se produce, en la barra superior de la vista, dispondremos de una flecha de atrás que nos llevará a la vista anterior en la que estamos.

3.2 Tipos de archivos y clases principales de Android

Para entender el desarrollo del proyecto, se hace necesario hacer una breve presentación de la estructura de Android. No se profundizará demasiado pues no es objetivo de este documento el instruir en Android, pero sí que se expondrán los principales conceptos que se usarán en esta aplicación.

3.2.1 Activity

La activity (o actividad) en Android es el componente más básico. Se trata de una aplicación que provee una pantalla con la cual los usuarios pueden interactuar. A cada actividad se le da una ventana, que normalmente rellena ocupa toda la vista, en la que dibujar la interfaz de usuario.

Una aplicación normalmente consiste en multiples actividades que están débilmente unidas entre ellas. Típicamente, una actividad en la aplicación es espeficada como principal, la cual es la primera en presentarse al usuario al lanzar la aplicación.

Cada actividad puede lanzar otra actividad con el objetivo de ejecutar diferentes acciones. Cada vez que una actividad nueva comienza, la anterior se para, pero el sistema preserva la actividad en una pila. La fila tiene un funcionamiento "último en entrar, primero en salir" de modo que en el momento en el que usuario finaliza con la actividad, ésta se cierra y se vuelve a la anterior.

3.2.2 Fragments

Un fragment representa el comportamiento o una porción de la interfaz de usuario en una actividad. Mientras que la actividad solo puede ejecutar una vista, con fragments se pueden combinar multiples vistas en una sola actividad. Se puede pensar en los fragments como una sección modular de la actividad con su propio ciclo de vida, sus propios eventos de entrada y el cual se puede crear o eliminar mientras que la actividad está funcionando.

Una actividad debe siempre estar embebida en un fragment y el ciclo de vida del fragment se ve directamente afectado por el ciclo de vida de la actividad. Si una actividad es destruida, todos los fragments que contiene también lo serán, por otro lado, mientras una actividad está corriendo, se pueden manipular cada fragment independientemente creándolos o destruyéndolos.

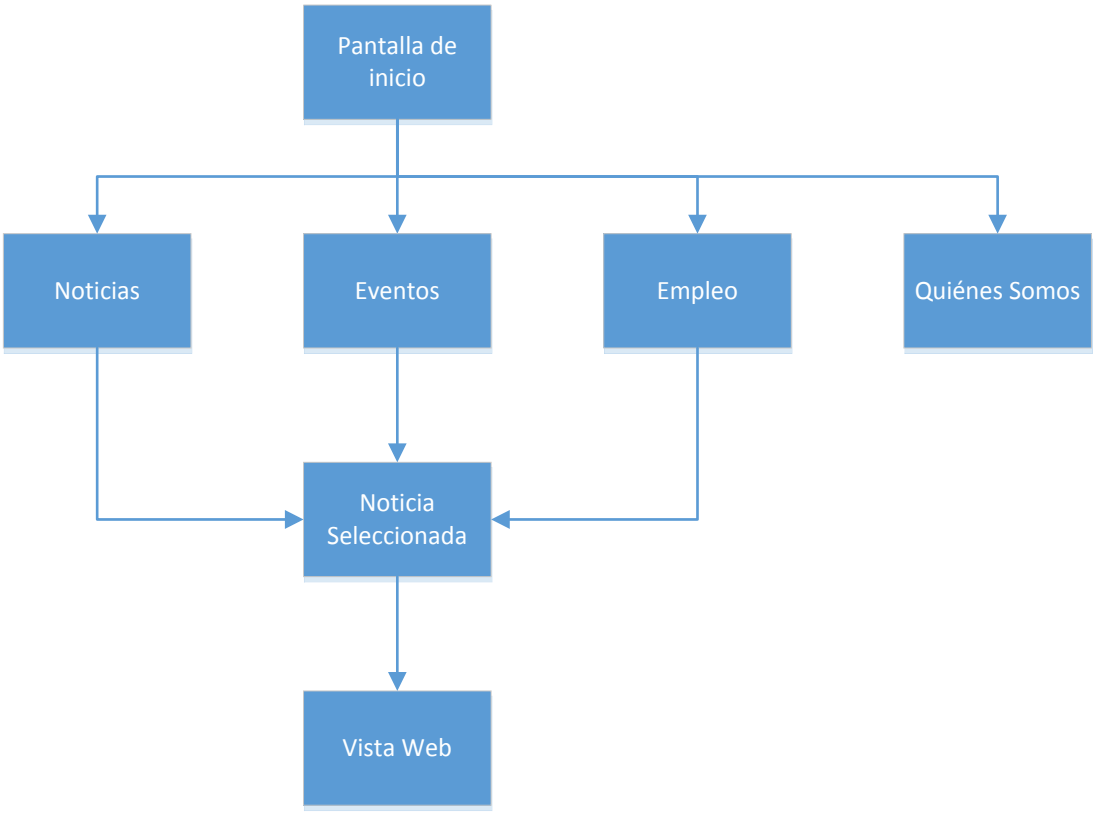

Ilustración 7 – Diagrama de vistas

3.2.3 Layout

El layout define la estructura visual para la interfaz de usuario, por ejemplo, para la de una actividad o un fragment. El layout viene escrito en XML y Android aporta una serie de etiquetas predefinidas con las cuales trabajar para la creación de los diferentes elementos de una interfaz. Una vez construido, el layout será llamado por las actividades o fragments para crear el interfaz.

3.2.4 Toolbar

Sustituye a la anterior Action Bar, pero se ha decidido en este proyecto usar la versión más moderna. Es la barra superior de la aplicación donde aparece el título, logo, puntos de menú o botones de atrás entre actividades. Se encuentra siempre visible y dentro de la jerarquía de layouts ocupa la posición supeior.

3.2.5 Viewpager

Es un layout maganer que permite al usuario navegar entre diferentes vistas a la izquierda y derecha. Es común usar el viewpager en conjunto con fragments. En este proyecto se usa para desplazarse entre noticias, eventos y empleo.

3.2.6 TabHost

Contenedor con etiquetas seleccionables que permiten seleccionar las vistas que se deseen. Se situa bajo la toolbar.

3.2.7 Adaptadores

Algunos tipos de layout precisan de adaptadores de contenido, de manera que dispongan de una clase que gestione como van a visualizarse los elementos dentro de él y de que tipo van a ser. Es el caso de los RecyclerViews o el ViewPager.

3.2.8 Intents

Un intent es una descripción abstracta de una operación a realizar. Puede ser usada para iniciar una actividad, lanzarla, iniciar un servicio, comunicarse con un servicio en background etc. Se usa principalmente para comunicación entre activities añadir otros datos a la petición.

3.2.9 Otros Elementos

A continuación, se describen brevemente otra serie de elementos usados en el proyecto. La implementación de algunos de ellos es compleja a pesar de encontrarse brevemente descritos en este apartado.

- Textview: Elemento de texto
- ImageView: Elemento imagen
- Buttons: Botón
- Dialog: Fragment que genera un cuadro de dialogo
- Coodinator Layout: Permite la gestión y animación de la toolbar
- RecyclerView: Vista que representa listados de manera más eficiente que la antigua ListView y da la capacidad de animar objetos e interactuar con ellos.
- Cardview: Elemento tarjeta que se crea tantas veces como ítems se tenga dentro del recyclerview.

Ilustración 8 – Tipos de elementos en la vista

3.3 Estructura de la aplicación a nivel de actividades y fragments

Una vez conocidos los distintos elementos que componen la aplicación. Se puede presentar un diagrama de flujo de la aplicación en la que se detallen todas las actividades, fragments y otros elementos de primer nivel que funcionan ligados a los fragments. Todo ello sin especificar qué funciones se ejecutan en cada instante ya que más adelante profundizaremos en detalle.

Fíjandonos en la ilustración 9, al iniciarse la aplicación se crea la actividad principal, ésta genera la pantalla inicial que se compone de toolbar, tabhost y viewpager. El viewpager a su vez crea tres fragments, por lo tanto, para la actividad principal tenemos tres vistas asociadas. Con la pulsación de alguno de los elementos de los de la vista y a través de los adaptadores de los recyclerviews contenidos en los fragments, se hace una llamada a la actividad “noticia completa” la cual mostrará una nueva pantalla con la información de la noticia seleccionada. Si por últmo pulsamos dentro de la noticia seleccionada la opción de visitar web, la actividad de noticia completa hará una llamada a la actividad webview.

Por otro lado, en la pantalla inicial de la actividad principal, el toolbar dispone de un botón de menú con la opción de consultar más información sobre la escuela, esta opción de menú llama a la creación de una nueva actividad con información de la AAA.

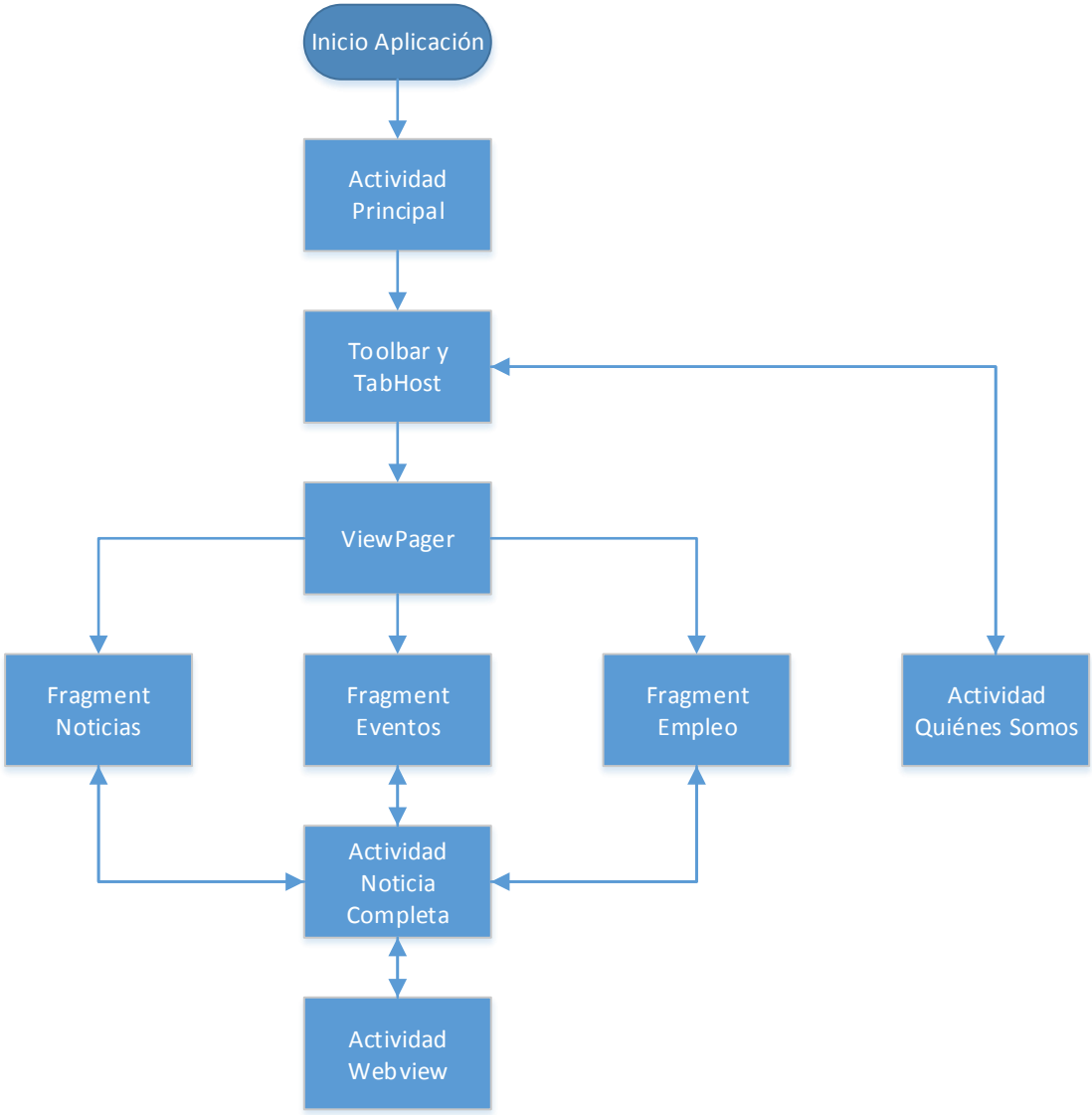

Ilustración 9 – Diagrama de flujo actividades y fragments

3.4 Implementación de la base de datos

Según la decisión tomada en el apartado dos, se implantará una base de datos SQLite ya que es ésta la que mejor se ajusta a las necesidades del proyecto.

En primer lugar, se escribirá un script que nos permita definir la estructura de la base de datos, los valores para las columnas de la tabla de la base de datos, los tipos de datos y la orden de creación de la tabla.

```
// Comando CREATE para la tabla ENTRADA
public static final String CREAR_ENTRADA =
        "CREATE TABLE " + ENTRADA_TABLE_NAME + "(" +
                ColumnEntradas.ID + " " + INT_TYPE + " primary key
autoincrement," +
                ColumnEntradas.TITULO + " " + STRING_TYPE + " not null," +
                ColumnEntradas.DESCRIPCION + " " + STRING_TYPE + "," +
                ColumnEntradas.URL + " " + STRING_TYPE + "," +
                ColumnEntradas.CATEGORIA + " " + STRING_TYPE +  "," +
                ColumnEntradas.FECHA + " " + STRING_TYPE +  "," +
                ColumnEntradas.CONTENIDO + " " + STRING_TYPE +  "," +
                ColumnEntradas.LEIDO + " " + INT_TYPE +  ")";
```

El trozo de código del archivo "ScriptDatabase" muestra el string "CREAR_TABLA" el cual recoge la sentencia preparada para la creación de la base de datos. Las variables como título o descripción han sido definidas anteriormente en el mismo script como strings que dan nombre a las columnas de las tablas. Por otro lado, "STRING_TYPE" y "INT_TYPE" también han sido definidos antiormente como strings que definen los tipos que SQLite dispone, "TEXT" e "INTEGER" correspondientemente.

Mediante la librería soporte de Android para SQLite llamada "SQLiteOpenHelper" se crea una clase que implementará diferentes funciones para la creación de la tabla y la agregación de entradas. A continuación, algunos trozos de código de las principales funciones.

```
@Override
public void onCreate(SQLiteDatabase db) {
    // Crear la tabla 'entrada'
    db.execSQL(ScriptDatabase.CREAR_ENTRADA);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // Añade los cambios que se realizarán en el esquema
    db.execSQL("DROP TABLE IF EXISTS " + ScriptDatabase.ENTRADA_TABLE_NAME);
    onCreate(db);
}
```

Además, se han creado distintas funciones para la obtención de entradas que más adelante llamaremos desde actividades o fragments.

```
/**
 * Obtiene todos los registros de la tabla entrada ORDER BY _ID ASC
 *
 * @return cursor con los registros
 */

public Cursor obtenerEntradas(String variable) {
    // Seleccionamos todas las filas de la tabla 'entrada' con categoría ?
    String val = variable;
    String query = "select * from " + ScriptDatabase.ENTRADA_TABLE_NAME + "
WHERE CATEGORIA = ? ORDER BY _ID DESC";
    return getWritableDatabase().rawQuery(query,new String[]{val}) ;
}

public Cursor obtenerUnaEntrada(String variable) {
    // Seleccionamos la fila con el ID determiando
    String val = variable;
    String query = "select * from " + ScriptDatabase.ENTRADA_TABLE_NAME + "
WHERE _ID = ? ";
    return getWritableDatabase().rawQuery(query,new String[]{val}) ;
}
```

Se han implementado también funciones de "insertar Entrada", "actualizar entradas" y "sincronizar entradas". Sincronizar entradas lleva a cabo quizás la más importante de las labores de la base de datos que consiste en comparar las entradas recién obtenidas del servidor con las almacenadas, estas son actualizadas si tienen campos diferentes y agregadas si no existían entradas con el mismo título. Se muestra un diagrama de flujo en la ilustración 10 para comprobar su funcionamiento.

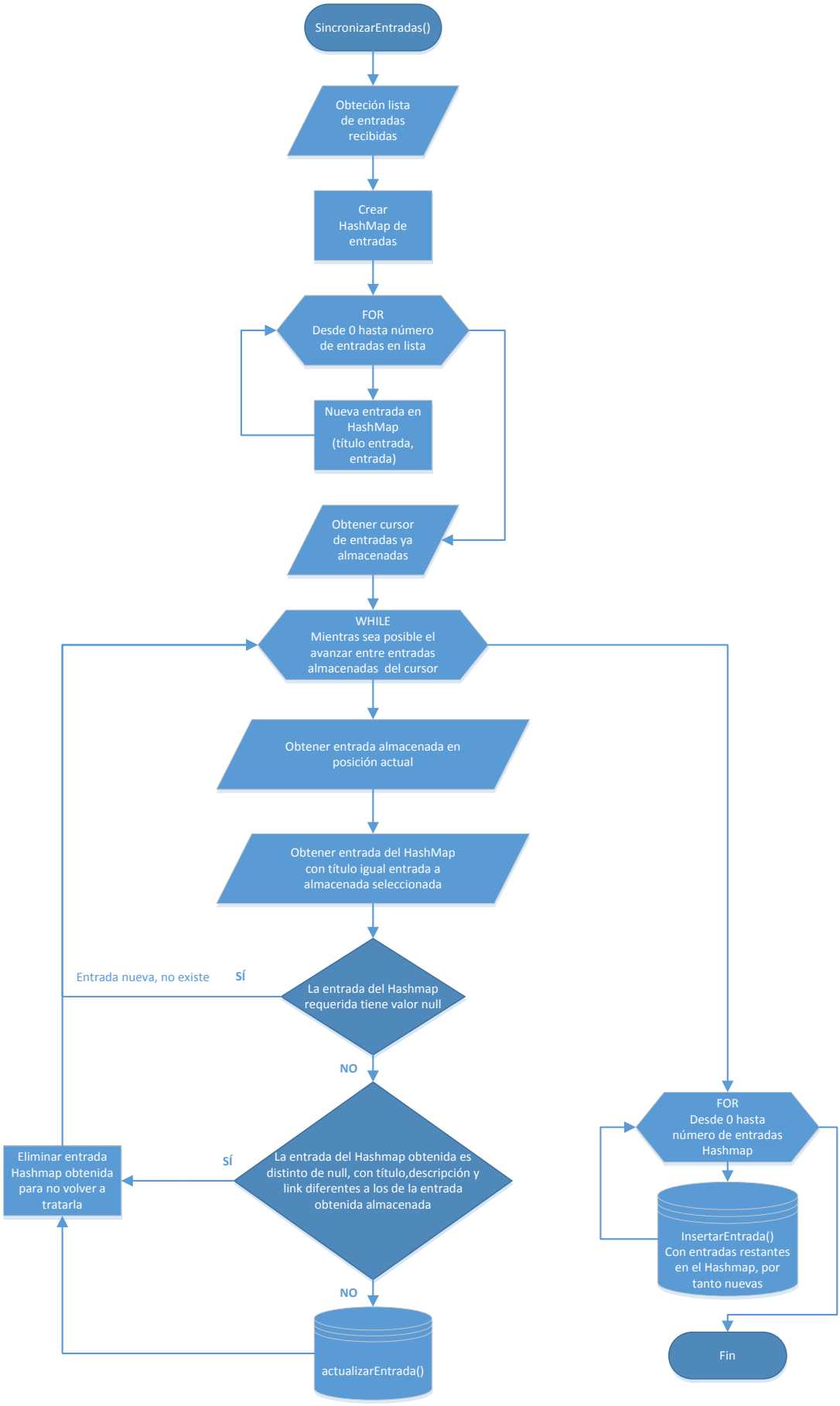

Ilustración 10 – Diagrama de flujo del funcionamiento de la función sincronizarEntradas

3.5 Conexión HTTP para obtención del RSS

Dentro de la actividad principal MainActivity, existe una función que genera una petición HTTP llamada onMethodRefreshCall. Esta función es accesible también a través de interfaces ligadas a los fragments para la actualización de contenido.

En la función se hace uso de la librería ConectivityManager el cual nos informa del estado de conexión del dispositivo, si se dipsone de ella o no. Si la conexión está disponible, se crea una instancia singleton a través de la librería Volley la cual trataremos en el siguiente punto. Dicha instancia al ejercutarse crea una nueva llamada XmlRquest al que se le pasa la URL del XML, la clase a la que tendrá el contenido devuelto y se establecem las funciones de respuesta y error. En la función de respesta onResponse pasamos todos los ítems parseados a la función sincronizarEntradas que tratamos en el apartado anterior, en el error mostramos un mensaje en consola.

```
@Override
public boolean onMethodRefreshCall() {
ConnectivityManager connMgr = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
if (networkInfo != null && networkInfo.isConnected()) {
    VolleySingleton.getInstance(MainActivity.this).addToRequestQueue(
            new XmlRequest<>(
                    URL_FEED,
                    Rss.class,
                    null,
                    new Response.Listener<Rss>() {
                        @Override
                        public void onResponse(Rss response) {
                            // Caching
                            notified[0] =
FeedDatabase.getInstance(MainActivity.this).

sincronizarEntradas(response.getChannel().getItems());
                            // Carga datos...
                        }
                    },
                    new Response.ErrorListener() {
                        @Override
                        public void onErrorResponse(VolleyError error) {
                            Log.d(TAG, "Error Volley: " +
error.getMessage());
                        }
                    }
            )
    );
} else {
    Log.i(TAG, "La conexión a internet no está disponible");
}
}
    return false;
}
```

3.5.1 Librería Volley

Volley es una librería desarrollada por Google para optimizar el envío de peticiones HTTP desde las aplicaciones Android hacia servidores externos. Este componente actúa como una interfaz de alto nivel, liberando al programador de la administración de hilos y procesos tediosos de parsing, para permitir publicar fácilmente resultados en el hilo principal. [12]

Volley posee varios componentes que optimizan la administración de las peticiones generadas desde las aplicaciones Android. La gestión comienza en una Cola de Peticiones que recibe cada una de las peticiones generadas, donde son previamente priorizadas para su realización.

Entre sus características mas potenciadoras podemos encontrar:

- Procesamiento concurrente de peticiones.
- Priorización de las peticiones, lo que permite definir la preponderancia de cada petición.
- Cancelación de peticiones, evitando la presentación de resultados no deseados en el hilo principal.
- Gestión automática de trabajos en segundo plano, dejando de lado la implementación manual de un framework de hilos.
- Implementación de caché en disco y memoria.
- Capacidad de personalización de las peticiones.
- Provee información detallada del estado y flujo de trabajo de las peticiones en la consola de depuración.

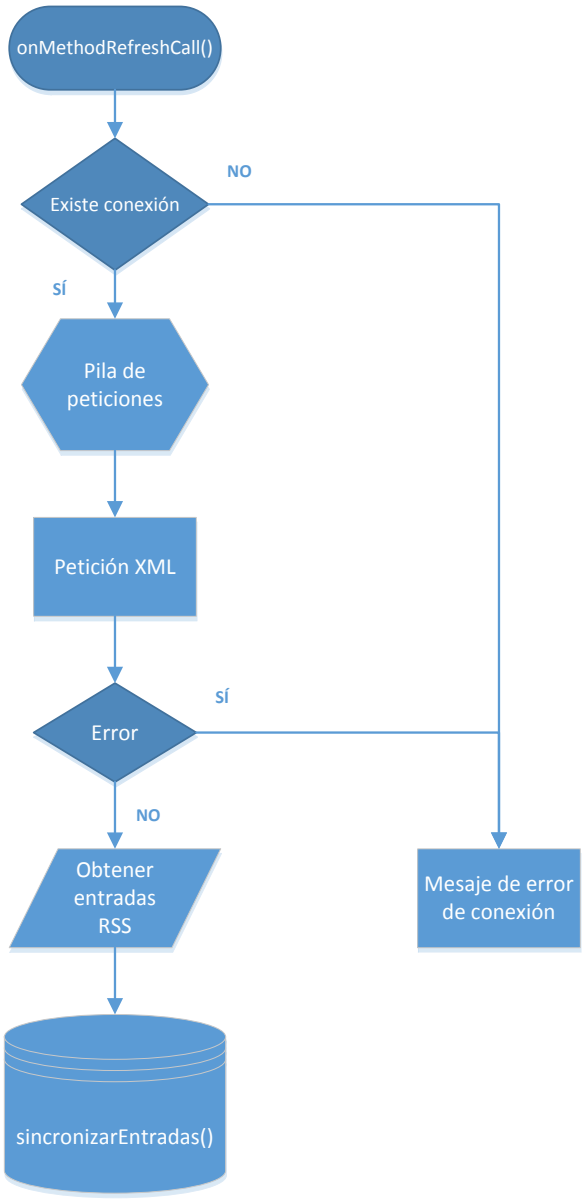

Ilustración 11 – Diagrama de flujo de la petición HTTP

3.6 Parsing del XML

Cuando el archivo XML es obtenido mediante la petición HTTP, es necesario el tratamiento del texto XML y sus etiquetas para sustraer la información necesaria que interesa tratar. Para ello Android dispone de ciertas librerías de bajo nivel, las cuales aportan una solución, pero resultan menos prácticas. Por el contrario, en el presente proyecto se hace uso de la librería Simple [13] para Java, la cual nos aporta una forma práctica y sencilla de extraer datos de nuestro RSS.

Conociendo la estructura del RSS, un ejemplo en código de cómo se haría el parsing en cada uno de los ítems (entradas) del documento XML obtenido sería de la siguiente forma:

```
@Root(name = "item", strict = false)
public class Item {

    @Element(name="title")
    private String title;

    @Element(name="description", required=false)
    private Description descripcion;

    @Element(name="link")
    private String link;

    @Element(name="category", required=false)
    private String categoria;

    @Element(name="pubDate", required = false)
    private Fecha fecha;

    @Element(name="enclosure", required = false)
    private Content contenido;
```

Creamos una clase a la que se le indica en la línea anterior que será el root. @Root establece la etiqueta principal de la que partimos a leer, seguidamente, @Element indica las etiquetas hijas de ese root, con etiqueta igual a "name". "Required", por otro lado, indica que si no se encuenra el campo, no es de obligatoria lectura para el parsing. Una vez obtenidos las diferentes variables, se han de establecer diferentes "getters" para las distintas variables.

Se han creado tres clases java diferentes para realizar el parsing del XML. En primer lugar la clase para RSS la cual establece como hijo a Channel. Por otro lado la clase Channel, la cual establece como hijos una lista de Items. Por último la clase Item, la cual mostramos en código, que obtiene el resto de datos de cada entrada.

Se puede apreciar en el trozo de código, que muchas de las variables obtenidas disponen de clase propia, esto es debido a la necesidad de hacer un pequeño tratamiento de los datos obtenidos antes de pasarlos para su tratamiento en la UI.

3.6.1 Description

Este campo es tratado para la eliminación del código CSS y HTML que viene incrustrado entre el texto de la etiqueta description. Una vez obtenido el campo, se le pasa un filtro a través del método Java regex, donde eliminamos todo el texto que cumpla lo establecido por el filtro, dejando así el texto descriptivo solamente.

Si el campo descripción es nulo, se establece la string descripción como "Sin descripción".

3.6.2 Fecha

La fecha obtenida por el RSS incluye un valor de "+0000" al final de la string. Eliminamos ese trozo para correcta visualización en la UI a través del mismo método regex.

```
public String getText()
{
    String regex = "\\+0000";
    String mod = FechaText.replaceAll(regex,"");
    return mod;
}
```

3.6.3 Content

Esta etiqueta, incluye además del nombre content, un atributo asociado dentro de la propia etiqueta. Este atributo contiene la URL de la noticia que estamos tratando. Para ello es necesario por tanto crear una clase que obtenga a través de la etiqueta @Attribute de la librería Simple, la URL que se busca de la noticia.

3.7 Asincronía

Cuando trabajamos con interfaces de usuario, es importante que ésta no quede bloqueada por los diversos procesos que la aplicación ha de ir ejecutando. Necesitamos por tanto, algo que nos permita llevar a cabo ejecuciones en segundo plano mientras la UI sigue corriendo. Para ello haremos uso de AsyncTask.

AsyncTask permite desarrollar operaciones en segundo plano y publicar los resultados en el hilo de ejecución del UI sin tener que manipular dichos hilos. Una tarea asíncrona está definida por tres tipos genéricos (Params, Progress y Result) y cuatro pasos: onPreExecute, doInBackground, OnProgressUpdate and OnPostExecute. En este proyecto solo ha sido necesario usar los pasos doInBackGround y onPostExecute.

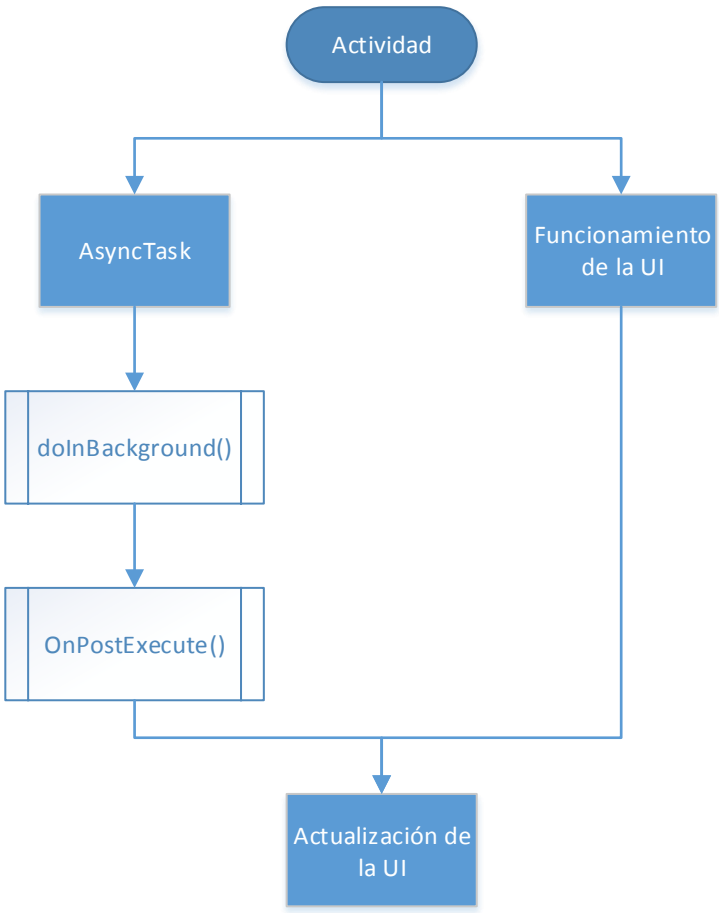

Ilustración 12 – Funcionamiento de AsyncTask

La actividad principal, MainActivity y los diferentes fragments asociados al Viewpager, hacen uso de AsyncTask a la hora de realizar la petición HTTP y cargar todo el contenido en la base de datos. Gracias a esto,

mientras se ejecuta todo este proceso, la UI no queda bloqueada y el usuario así disfruta de una experiencia fluida.

3.8 Fucionamiento de las distintas partes de la aplicación

Para un mejor entendimiento del funcionamiento de todas y cada una de las partes de este proyecto, se describirán a continuación los diferentes archivos y clases que se han usado y sus diagramas de flujo.

3.8.1 AndroidManifest

Este archivo en formato XML se declaran cuáles son las actividades que van a existir dentro de nuestra aplicación, se designa cuál es la actividad principal, como se relacionan entre ellas, el nombre de la aplicación y establece cuales son los permisos requeridos para la aplicación. En este caso los permisos requeridos son los de acceso a internet y consultar estado de la red en el teléfono.

3.8.2 MainActivity

La actividad principal es la primera en ejecutarse dentro de la aplicación. Esta actividad extiende la clase AppCompactActivty la cual permite crear una visualización estándar de activity junto con la toolbar. Nada más crearse, ejecuta un AsyncTask donde:

- **Background:** llama al método onMethodRefreshCall que ejecuta la petición HTTP y almacena las entradas en la base de datos.
- **OnPostExecute:**
    - Se establece el layout que generará la vista principal de la pantalla (“activity_main”)
    - Se declaran los objetos a los diferentes elementos del layout
    - Se configuran los tabs del TabHost donde se le indica el número de apartados y los títulos de estos.
    - Se configura el Viewpager al cual se le establece el número de páginas que mantendrá en memoria, tres y se le asigna el adaptador que generará las vistas.
    - Se enlazan Viewpager y Tabhost para que el funcionamiento de ambos esté ligado.

Fuera del Asyntask se configuran las opciones de menú, del toolbar, donde aparecerán tres puntos en la esquina superior derecha de la pantalla. En la configuración establecemos el número de opciones que aparecerán en dicho menú y en estableceremos las acciones a realizar ante la pulsación de las diferentes opciones.

MainActivity contiene el método onMethodRefreshCall que ejecuta la petición al servidor RSS y almacena las entradas en la BBDD. Este método es ejecutable desde los diferentes fragments, a la hora de actualizar contenido, a través de interfaces. Para el funcionamiento de estas interfaces, la clase MainActivity debe extender además las interfaces FragmentNoticias.RefreshCall, FragmentEventos.RefreshCall, FragmentEmpleo.RefreshCall.

3.8.3 ViewPagerAdapter

Extiende a la clase FragmentStatePagerAdapter y se encarga de la gestión de las posiciones del Viewpager y de determinar el número de páginas que tendrá el Viwepager. Esta clase además gestiona el guardado y recuperación de los estados del fragment. A través de una función “Case”, le indicamos qué Fragment ha de crear en cada posición (noticias, eventos o empleo).

3.8.4 Fragments noticias, eventos y empleo

Estos fragments son creados directamente por el ViewPager y generan el contenido que vemos bajo los TabHost. El diagrama de funcionamiento se muestra a continuación.

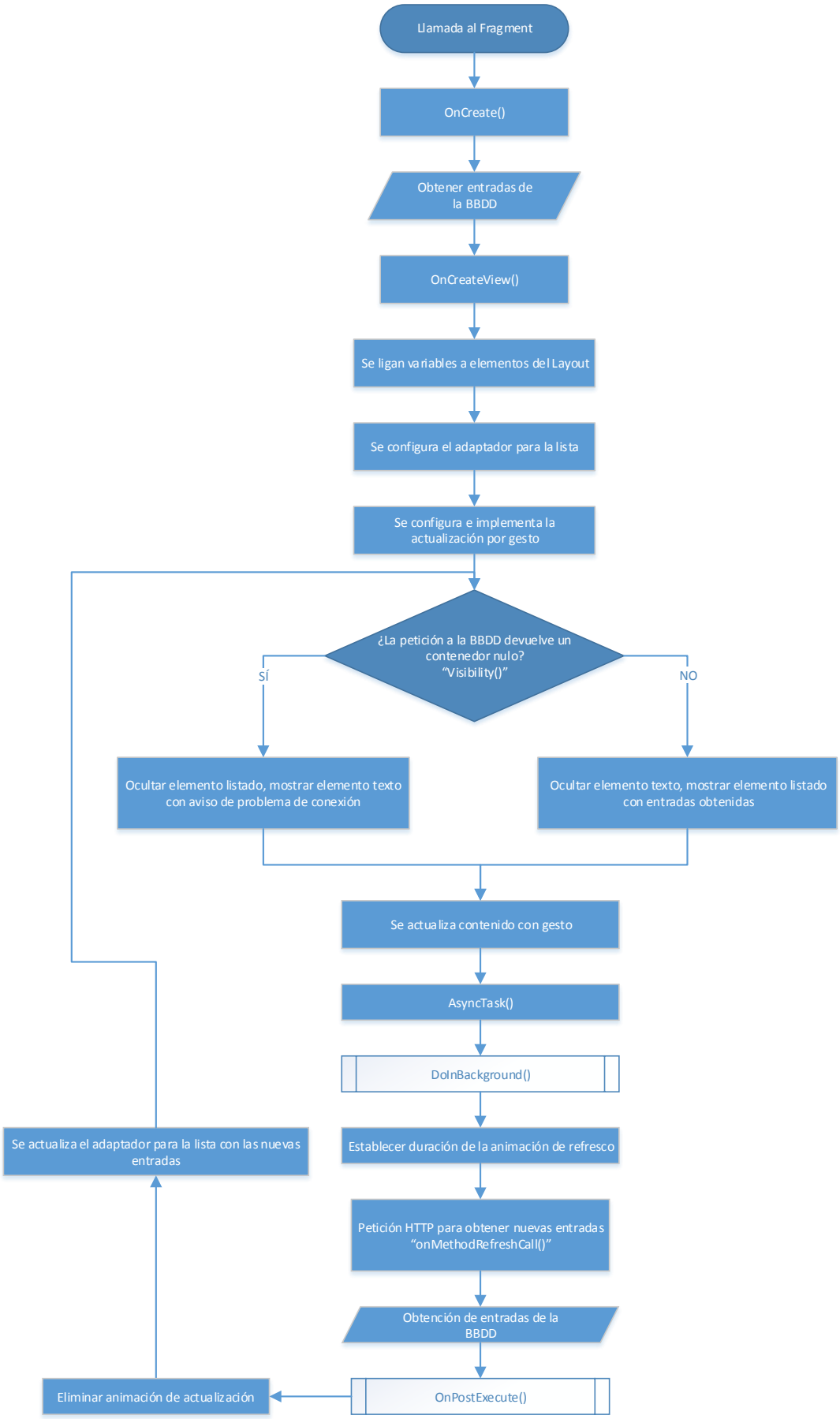

Ilustración 13 – Diagrama de flujo del Fragment

El fragment tiene un ciclo de vida en el que existe diferencia entre la creación del fragment, y la creación de la vista. Esto puede ser aprovechado para ejecutar peticiones antes de que la vista se genere, obteniendo a tiempo así los datos necesarios para la vista. En el diagrama de flujo, vemos que estos estados se corresponden con OnCreate() y OnCreateView() respectivamente.

El fragment está preparado para visualizar un mensaje o un listado en función si existe contenido o no en la base de datos. Jugando con la visibilidad en Android, podemos ocultar el listado de entradas para mostrar un mensaje de falta de conexión y viceversa.

Ilustración 14 – Animación de actualización

Una vez generada la vista, a través de un listener implementado por RefreshLayout (una librería de Android para la animación por gesto), se detecta el gesto de actualización y se llama a la función que la ejecuta. Para llevar a cabo un procedimiento de actualización, se usa un AsyncTask para no congelar la UI. Aprovechamos además, para establecer en ese AsyncTask el tiempo de animación de actualización y pararlo, realizar la llamada HTTP, cargar las nuevas entradas en la BBDD, obtenerlas y actualizar con ellas el adapdor de la lista.

3.8.5 RVAdapter

RVAdapter (RecyclerView Adapter) se encarga de adaptar o establecer toda la información que será mostrada por la lista scrollable de Recyclerview en cada uno de los fragments (noticias, eventos y empleo). El funcionamiento de este adaptador es sencillo y se representa claramente en los siguientes diagramas.

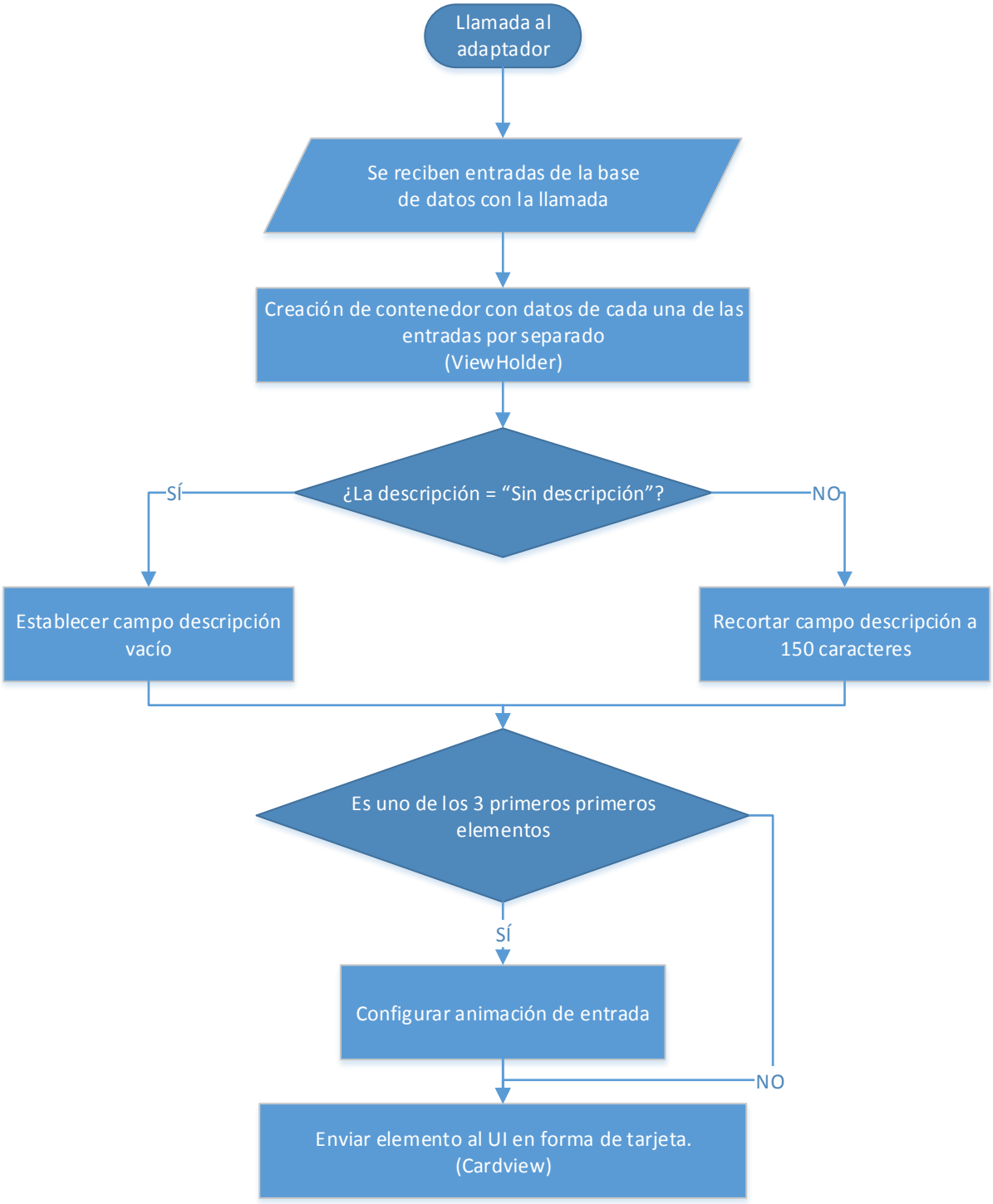

Ilustración 15 – Funcionamiento RVadapter

En primer lugar, tal y como se muestra en la ilustración 15, RVAdapter lleva a cabo la generación individual de cada uno de los elementos Cardview que se van a visualizar más tarde en el Recyclerview de los fragments asociados. Conociendo que los fragments son previsualizaciones de noticias, establecemos un máximo de texto a ver en las tarjetas (150 caracteres), a menos que el campo descripción venga dado como "sin descripción" en cuyo caso no mostraremos ningún texto asociado. Esto se debe a que el apartado empleo acostumbra a venir sin descripción y ver el mismo mensaje repetido en cada tarjeta resulta cargante. Además, en pos de cuidar el diseño de la aplicación, se llevará a cabo una animación de entrada de las 3 primeras tarjetas haciendo uso de un XML tipo anímate.

El adaptador es en todo momento el que tiene constancia de los elementos que son visualizados y pulsados

dentro de la UI, es por ello que también es el encargado de llevar a cabo acciones si interactuamos con los elementos que genera. Por tanto, dentro del adaptador va includo un listener el cual genera determinadas acciones en función de la noticia que pulsemos pero que en todo caso, nos llevará a la generación de la noticia a pantalla completa.

Según vemos en la ilustración 16, el listener escucha qué noticia ha sido pulsada, comprueba a través del ViewHolder (contenedor de información de la noticia seleccionada) cuál es la categoría de la noticia y su ID, y en función de eso crea un Intent para llamar a la actividad NoticiaCompleta a la vez que se envía el tipo de categoría a representarse y el ID de la noticia.

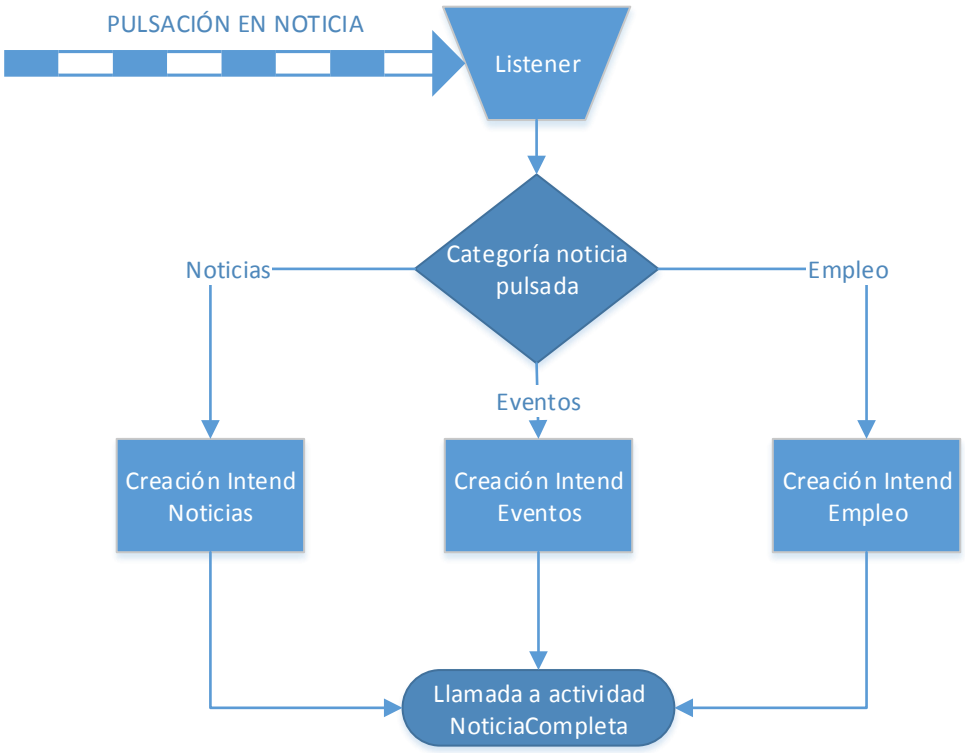

Ilustración 16 – Funcionamiento Listener RVAdapter

3.8.6 NoticiaCompletaActiviy

Esta actividad se genera a través de un intent producido por RVAdapter tras una pulsación en una noticia. Asociado al intent se encuentra almacenado el ID de la notica y el tipo de categoría de ésta. En función de esos datos, la actividad de noticia completa mostrará una serie de elementos u otros, cambiado su aspecto y opciones disponibles.

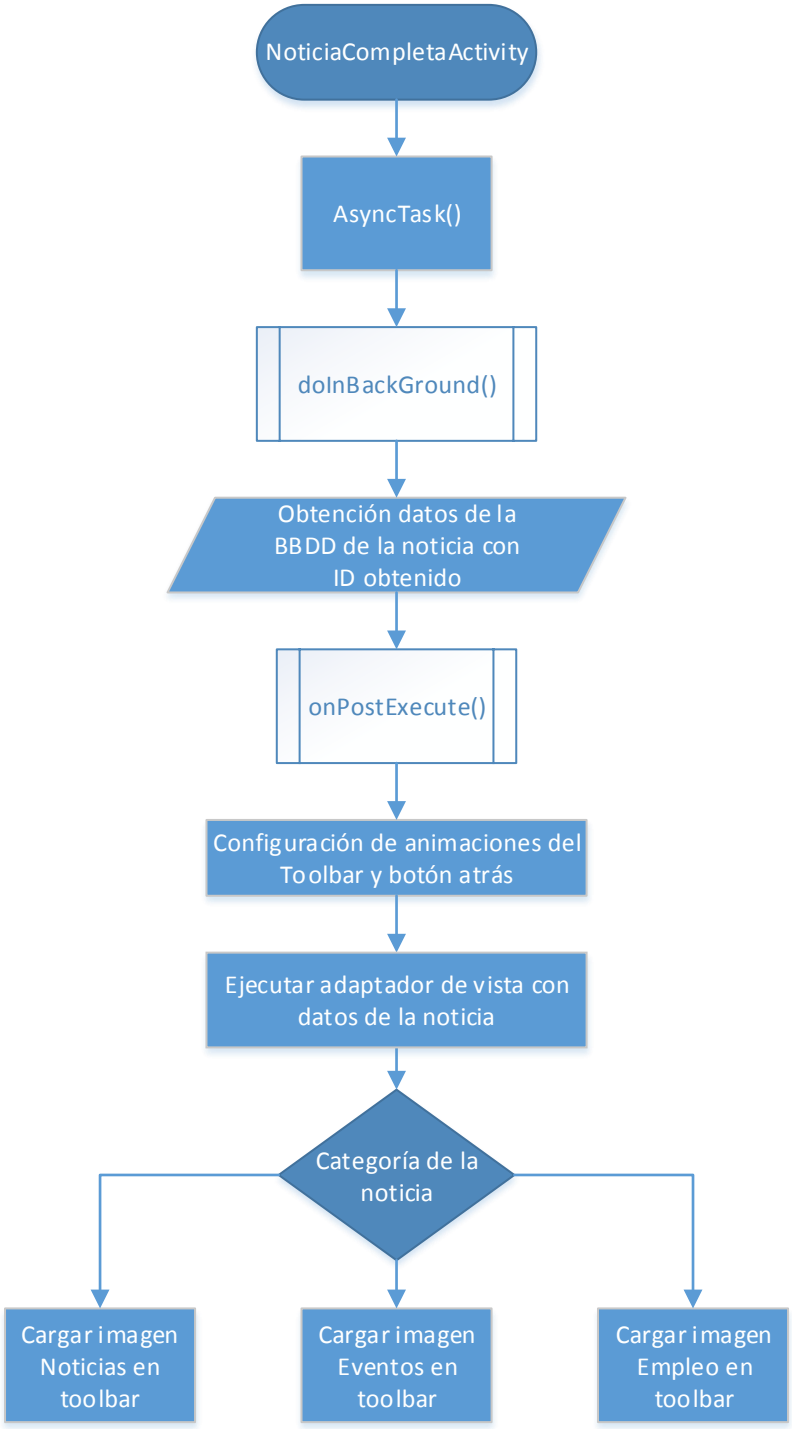

Ilustración 17 – Diagrama flujo ActividadCompleta

La noticia se mostrará dentro de un CardView, el cual será generado por RVAdapterCompleta e incluirá botones en función de las características de la noticia seleccionada.

3.8.7 RVAdapterCompleta

Este adaptador, en lo básico, realiza las mismas acciones que RVAdapter, crear objetos relacionados con el layout, asignar valores a estos elementos del layout y devolver una vista Cardview de la noticia.

Dentro del Cardview nos encontramos con tres botones diferentes que aparecerán en la esquina inferior derecha de la noticia. Los botones son “IR A LA WEB”, “DESCARGAR PDF” e “INSCRÍBETE”, estos dos últimos

botones aparecerán en la noticia en función de las características de ésta. Si la noticia incluye un campo "enclosure" aparecerá el botón "DESCARGAR PDF", si en cambio la noticia es del tipo "convocatoria" aparecerá el botón "INSCRÍBETE" el cual genera un cuadro de dialogo para la inscripción.

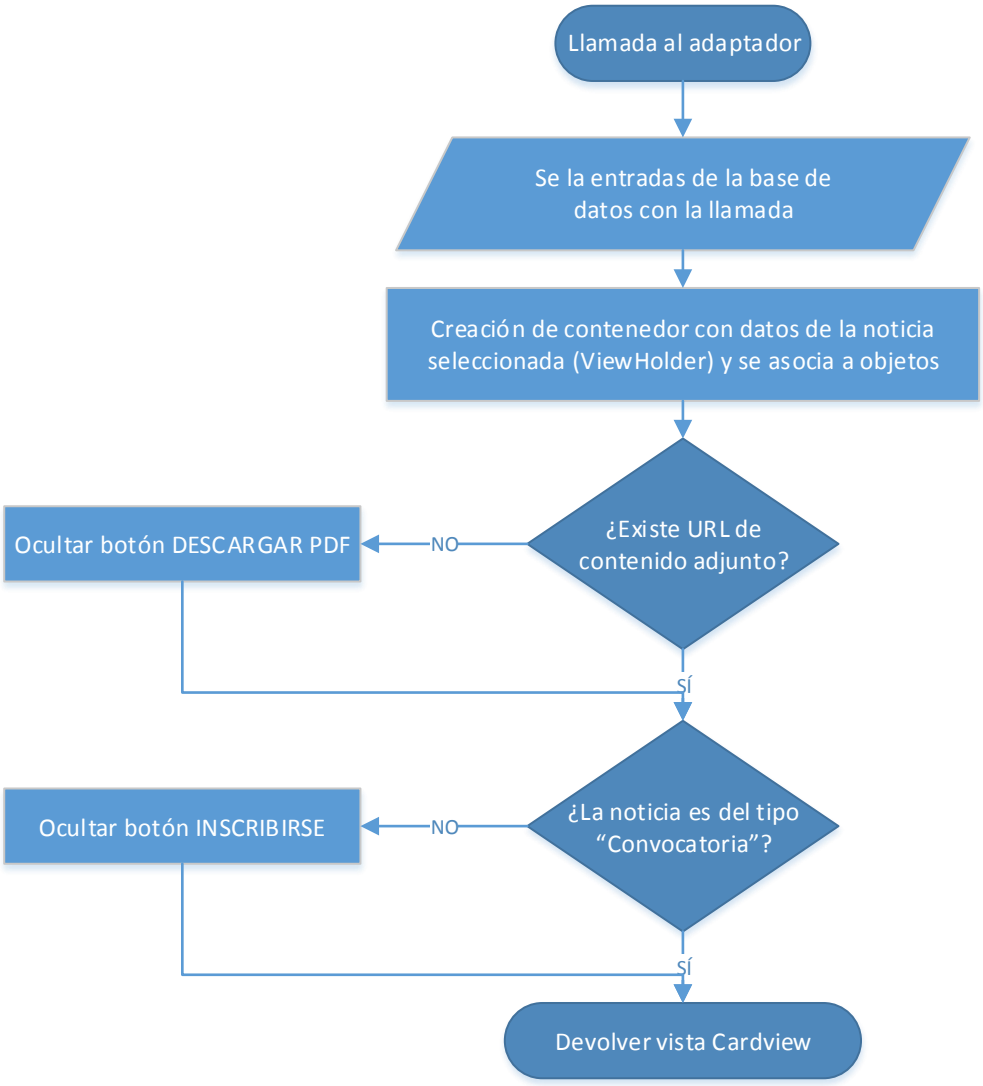

Ilustración 18 – Diagrama de flujo RVAdapterCompleta

Al igual que en RVAdapter, este adaptador también dispone de diferentes listeners para dar funcionalidad a los tres botones posibles de pulsar. En teoría, los listeners son funciones independientes pero en el siguiente diagrama de flujo serán mostrados como uno solo que ejecuta diferentes acciones para una más sencilla compresión de las acciones que se llevan a cabo tras pulsar los botones.

Dos de los tres botones generan un intent, uno es un intent directo y otro indirecto. El primero, WebView, se trata de un intent directo, el cual ordena específicamente qué actividad ha de iniciarse y a través de la propia orden, se envía la URL destino. El segundo, abrir pdf, es un intent indirecto, ya que manda la orden al sistema Android y éste hace aparecer un cuadro de diálogo con los lectores disponibles, en el caso de que existan. Por último el tercer botón llama a una función de NoticiaCompletaActivity en la que a través del FragmentManager, se crea un pequeño fragment a modo de ventana de dialogo donde aparecen dos campos formulario para la inscripción.

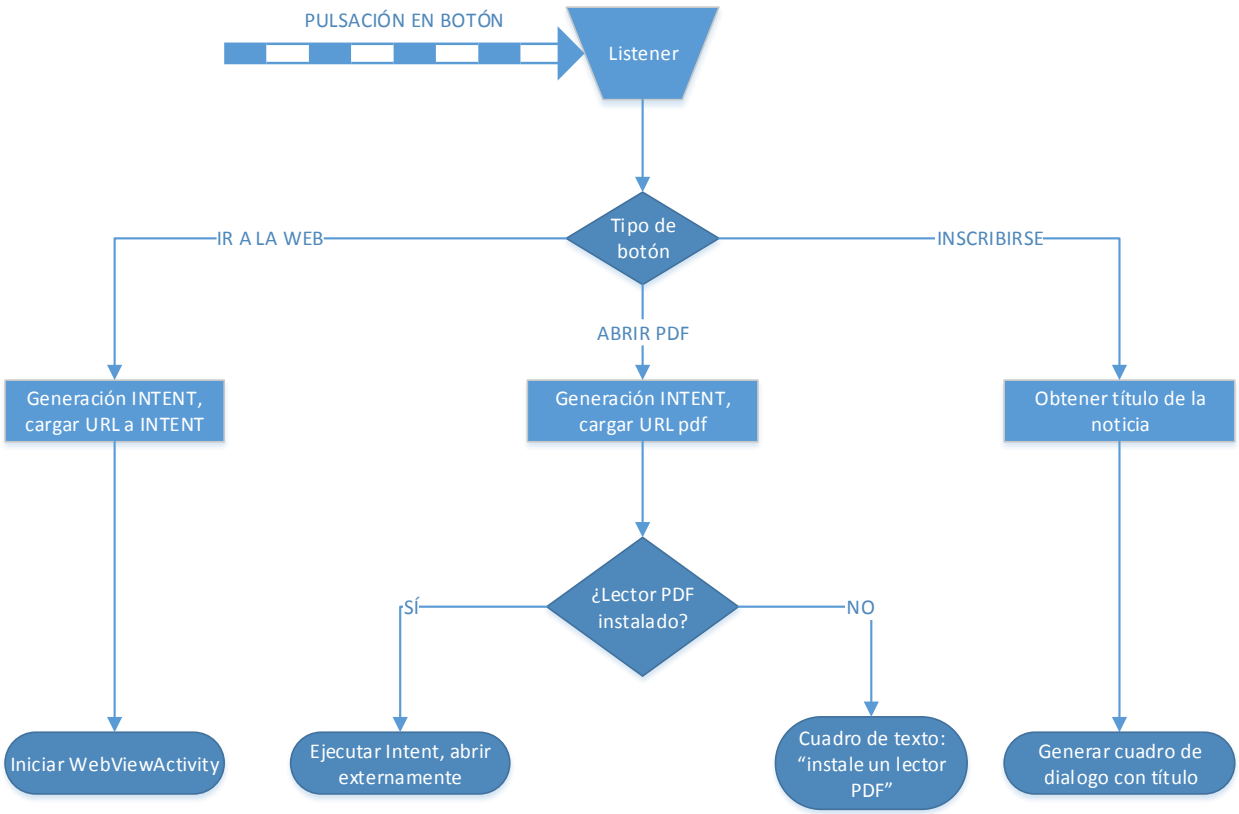

Ilustración 19 – Listener RVAdapterCompleta

3.8.8 FragmentDialog

Esta clase extiende otra llamada DialogFragment la cual ya genera unas características de layout predeterminado para ventanas de dialogo. Aun así, es necesario que carguemos un layout propio para añadir funcionalidades no estándares como son los cuadros de entrada de texto. Se requieren dos tipos de datos, nombre y apellidos del asociado y DNI. Así, más tarde el responsable de la AAA podrá comprobar si dicha persona que se pretende inscribir se encuentra inscrito en la Asociación.

El funcionamiento del cuadro de diálogo es sencillo, se pide un mínimo de caracteres para que no se introduzca ningún nombre al azar, se pide un número de DNI y ser verifica si es válido. Si no se cumple alguno de estos requisitos se muestra un error, en caso de cumplirse, se generará un email con todos los datos como se ilustra a continuación.

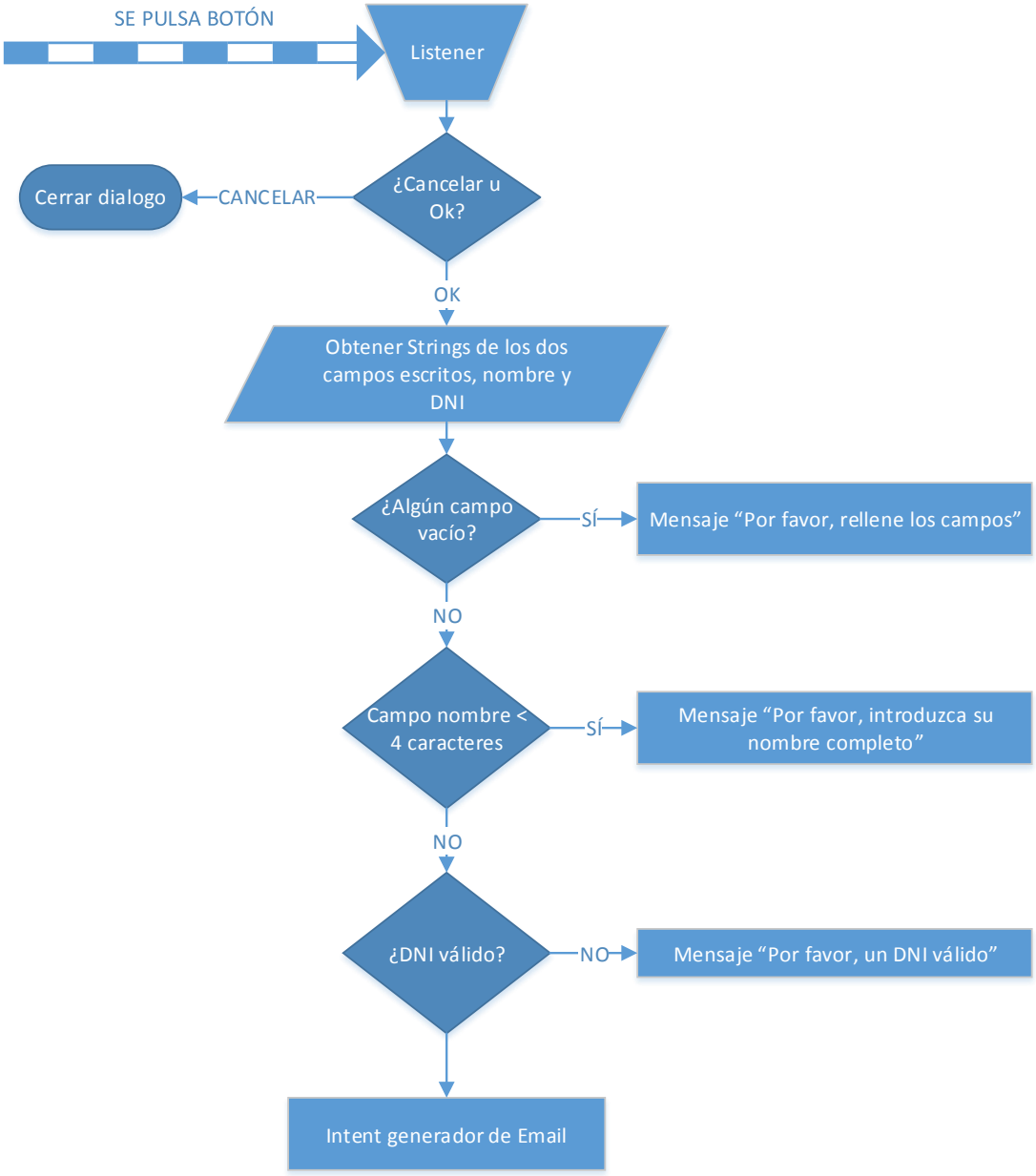

Ilustración 20 – Diagrama de flujo del Listener de FragmentDialog

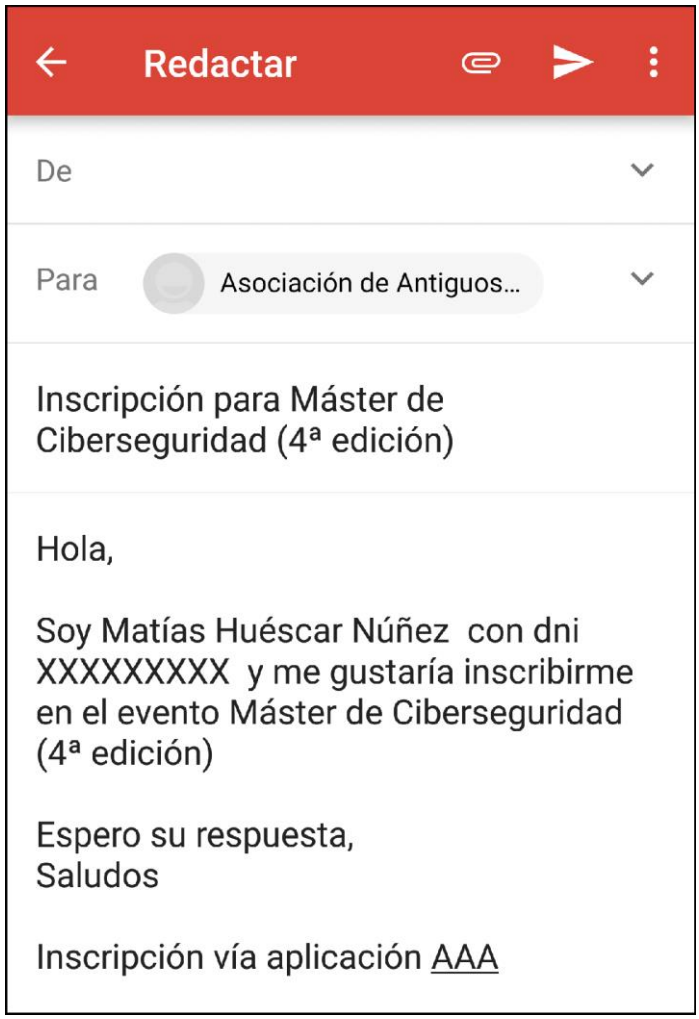

Ilustración 21 – Email generado por FragmentDialog

3.8.9 WebActivity

Esta actividad carga un Webview, su funcionamiento es sencillo. A través de la petición Intent de FragmentDialog se recibe la URL, una vez obtenida ésta se establece como URL a cargar. A continuación, se hacen configuraciones para que aparezca la Toolbar en la zona superior de la pantalla, hacer que funcione el botón atrás, establecer que se pueda hacer zoom o que cargue elementos JavaScript.

Por último, se configura la animación de carga para que aparezca visible al comenzar la página y desaparezca una vez esté cargada la página o se produzca un error.

3.8.10 SomosActivity y RVAdapterSomos

Esta actividad, a la cual se accede desde el menú de tres puntos de la Toolbar, genera una vista sencilla, parecida a la de una noticia, donde se da una introducción a cerca de lo que es la Asociación. Tenemos por tanto una actividad con un Reciclerview asociado a éste último su adaptador correspondiente, el funcionamiento es el mismo que los anteriores.

Ilustración 22 – Menú desplegable

3.9 Implementación de Material Desing y diseño

Gran parte del tiempo invertido en la realización de esta aplicación ha sido para la creación de efectos y animaciones o para el aprendizaje del uso de estos a partir de librerías propias de Android. Se entiende como parte fundamental de una aplicación el diseño que pueda tener ésta y todo aquello que hace una interfaz amigable y versátil.

3.9.1 Colores

Se han implementado principalmente los colores del logo de la ETSI que son los siguientes:

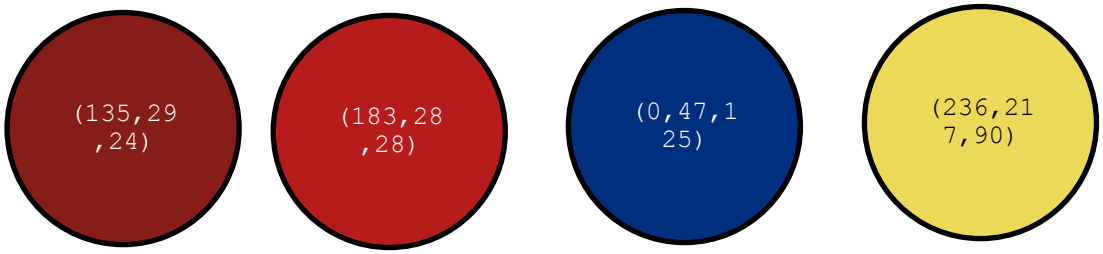

Ilustración 23 – Colores de la aplicación

Leídos de izquierda a derecha, el primer color representa el que corresponde a la barra de notificaciones, un tono más oscuro del color principal de la aplicación. A continuación, segundo tono de rojo es usado para toda la interfaz, es el color predominante. El tercer color, el azul, es usado para los títulos de las noticias en los diferentes fragments y el amarillo, siendo un color más difícil de añadir a la interface, es usado en el logo de animación de actualización y en aparece en los logotipos de la AAA.

Además de estos colores, se usa el blanco, el negro, un tono azul para botones predeterminado del sistema.y un gris de fondo. Es muy importante el uso de éste último para no cansar la vista y para ganar contrastes dentro de la interface. El tono de gris (238, 238, 238) se usa de fondo en las vistas principales creando un contraste con el blanco de la Cardview.

3.9.2 Toolbar inteligente

Esta parte del layout nos presenta el título de la aplicación, el logo y un pequeño menú desplegable a la derecha. Se encuentra en parte superior y tiene una altura de unos 30 dp.

La programación de una Toolbar inteligente consiste en, gracias a la implementación de "coordinatorLayout", realizar configuraciones de tal modo que cuando se produzca un scroll hacia arriba del contenido de la lista de entradas, la Toolbar se oculte progresivamente en proporción al scroll, creando un bonito efecto de ocultación a la vez que el usuario gana más cantidad de pantalla para leer contenido. En sentido contrario, al hacer scroll hacia abajo la Toolbar volvería a aparecer con todas sus funcionalidades.

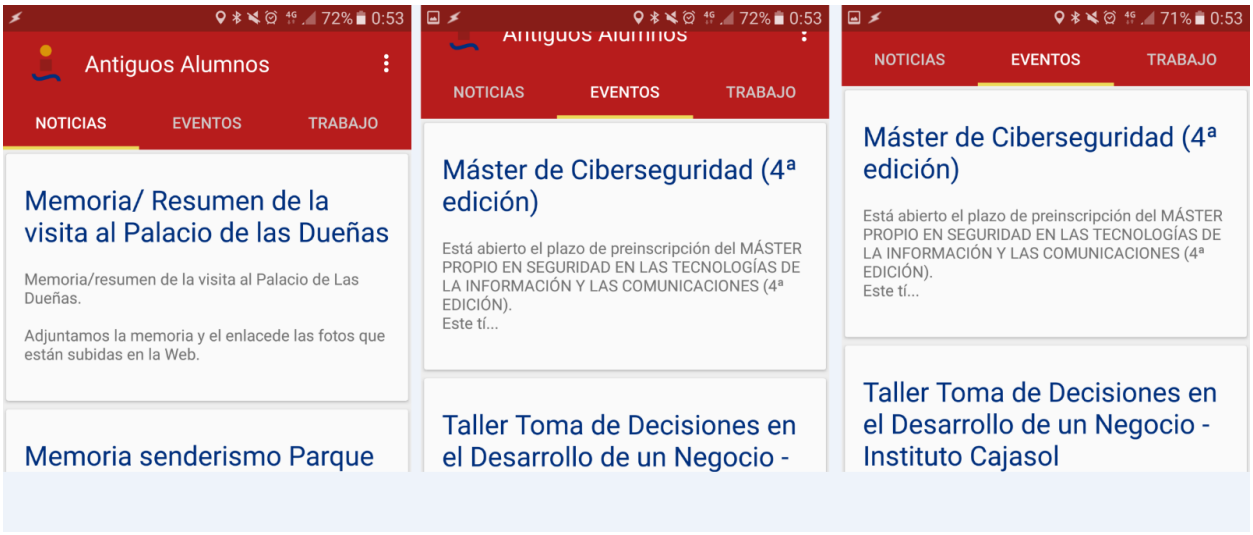

Ilustración 24 – Toolbar inteligente

Para su implementación se necesita que el layout esté construido en la siguiente jerarquía. Nótese que los distintos elementos no van acompañados de sus configuraciones pertinentes ya que se pretende mostrar simplemente el "esqueleto" del layout. Todo el código detallado se encuentra en los anexos.

```
<android.support.design.widget.CoordinatorLayout>
 <android.support.design.widget.AppBarLayout>
  <android.support.v7.widget.Toolbar>
   <RelativeLayout>
    <TextView/>
    <ImageView/>
   </RelativeLayout>
  </android.support.v7.widget.Toolbar>
```

```
 <android.support.design.widget.TabLayout/>
</android.support.design.widget.AppBarLayout>
 <android.support.v4.view.ViewPager/>
</android.support.design.widget.CoordinatorLayout>
```

3.9.3 Tarjetas Cardview

Para la implementación de las tarjetas, hemos empleado las siguientes líneas de diseño.

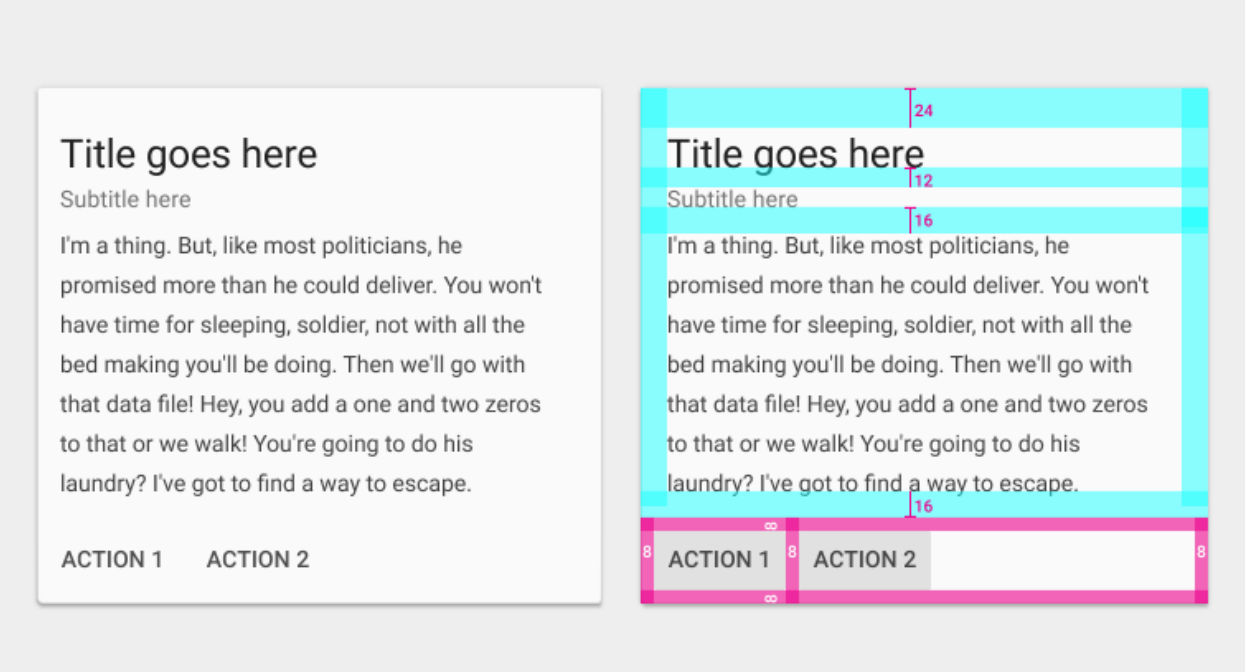

Ilustración 25 – Diseño Cardview

Se han establecido los tamaños de letra de título y descripción, 24 y 14sp correspondientemente y se han aplicado los paddings a los diferentes bordes que rodean a los botones, título y descripción tal y como ilustra la imagen.

3.9.4 Toolbar convertible

Una de las animaciones más vistosas de la aplicación es la encontrada dentro de las noticias, la toolbar convertible. Al abrir la noticia, encontramos una vista completa de foto más título más cuerpo de la noticia. Si hacemos scroll hacia arriba, la foto se va convirtiendo sutilmente en la toolbar, sustituyendo la imagen por un color y cambiando la posición y el tamaño del título. Ocurre lo contrario al volver a hacer scroll en dirección opuesta.

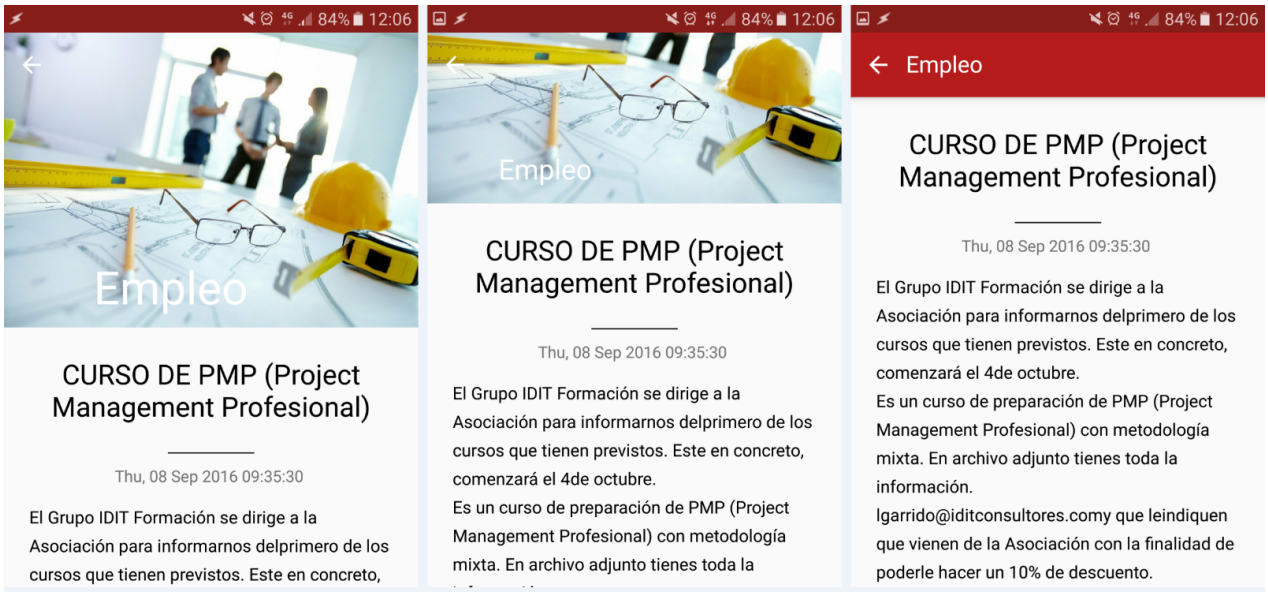

Ilustración 26 – Toolbar Convertible

Para su implementación se necesita que el layout esté construido en la siguiente jerarquía. Nótese que los distintos elementos no van acompañados de sus configuraciones pertinentes ya que se pretende mostrar simplemente el "esqueleto" del layout. Todo el código detallado se encuentra en los anexos.

```
<android.support.design.widget.CoordinatorLayout>
 <android.support.design.widget.AppBarLayout>
  <android.support.design.widget.CollapsingToolbarLayout>
   <ImageView/>
   <android.support.v7.widget.Toolbar/>
  </android.support.design.widget.CollapsingToolbarLayout>

 <android.support.design.widget.TabLayout/>
</android.support.design.widget.AppBarLayout>
 <android.support.v7.widget.RecyclerView/>
</android.support.design.widget.CoordinatorLayout>
```

3.9.5 Fuente

Roboto es la fuente elegida y usada para esta aplicación. Descrita por sus creadores como cercana y moderna, Roboto es una fuente de la familia sans-serif desarrollada por Google. Esta fuente ha sido creada y rediseñada para una óptima visualización en pantallas móviles y es la fuente principal de Google+, Google Play, Youtube o Google Maps.

Ilustración 27 – Fuente Roboto

3.9.6 Animación de las tarjetas de entrada

Para evitar una interfaz aburrida en la que todo parezca estático se han añadido animaciones de entrada para las primeras noticias que aparezcan en el layout. La animación de entrada se establece dentro de los adaptadores Recyclerview.

3.9.7 Pantalla de arranque

Normalmente para la creación de una pantalla de arranque se crea una actividad que una vez mostrada durante unos segundos, deriva a la actividad principal donde comienza el funcionamiento normal de la aplicación. En este caso se ha optado por una solución mucho más elegante, basada en el cambio del archivo estilo de la aplicación al inicio de esta.

AndroidManifest es el primer archivo que se inicia al abrirse la aplicación, este establece el estilo de la aplicación nada más comenzar y es ahí cuando introduciremos un estilo al que se le establece un background el cual será la de la imagen de inicio. Una vez se lanza MainActiviy y es creada, se vuelve a cambiar el archivo de estilos de la aplicación, dando de nuevo el estilo correcto de visualización de la aplicación.

La ventaja de esta solución es que es práctica, rápida, no retiene la aplicación tiempos ficticios pues la imagen dura el tiempo que la aplicación tarde en lanzar MainActivity y no necesita de crear otra activity solo para ello. Por otro lado, no se pueden implementar elementos de carga como iconos animados o barras de loading.

Ilustración 28 – Imagen de arranque de la aplicación

4 IMPLEMENTACIÓN, PRUEBAS Y RESULTADOS

Durante todo el proceso de desarrollo van apareciendo problemas de implantación, pivotajes de los planteamientos y simulaciones. En este apartado hablaremos de estos apartados. Además, se inclurá un apartado de Play Store donde se conocerá su funcionamiento y uso.

4.1 Librerías de soporte

La librería de soporte Android ofrece un número de características que no se encuentran en el framework. Estas librerías ofrecen retrocompatibilidad de versiones para las nuevas características, proveyendo útiles elementos de UI que no están incluidos en el framework y haciendo compatible una cantidad de utilidades que se pueden aprovechar. Para la mejor implementación en las distintas versiones de Android y complIr con nuestro objetivo de hacer compatible la aplicación hasta la API 16, es muy necesario trabajar con las librerías de soporte.

Siempre que se ha implementado un elemento o importado clases, se han hecho siempre de la librería de soporte, concretamente de la v4 y v7. Gracias al IDE de Android Studio, se hace más llevadero el control de funciones implementables, obsoletas o no interpretables por otras versiones. Tablayout, Viewpager, Toolbar y Fragments son ejemplos de elementos usado con librerías de soporte en este proyecto.

```
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
```

4.2 Soporte para Cursor en adaptadores de Recyclerview

Antes de que se extendiese el uso de Recyclerview como nuevo estándar, se disponía de los ListView. Estos, aunque menos eficientes, disponían de clases, las cuales se podían extender para dar copatibilidad al ListView con el tratamiento de Cursores para BBDD.

Por desgracia, no ocurre lo mismo con RecyclerView. El adaptador de esta lista no tiene clases propias en el framework para la implementación de Cursores ya que, al ser una nueva clase más versátil, la solución la puede quien lo necesite.

Durante los primeros meses de programación, la conversción del adapter para hacerlo capaz de tratar este tipo de datos fue bastante tedioso. Finalmente gracias a la comunidad online se lograron encontrar algunas clases capaces de mejorar el trabajo que se había realizado hasta el momento. Se trata de una clase abstrata llamada CursorRecyclerViewAdapter.

4.3 Etiquetas RSS y la clasificación de contenido

Inicialmente, la aplicación debía contar, dentro del apartado Empleo, con un selector en el que el usuario pudiese clasificar las ofertas en función de la titulación seleccionada. Hacer un selector de estas características es sencillo y simplemente se cambiaría la petición a la BBDD en función de lo seleccionado, recargando la vista.

El problema surge del lado del servidor ya que las etiquetas usadas en el apartado, categoría, no dan suficiente información, por ejemplo, para discernir una oferta de empleo de telecomunicaciones, de una de industriales. Todas las ofertas de empleo llevan la categoría “empleo”, por tanto, desde la aplicación no se dispone de mayor información para su filtración.

Una solución podría ser el parseo de títulos de las entradas de empleo para encontrar palabras clave que identificasen la oferta como de una titulación u otra, pero la mayoría de las entradas no incluyen información de ese tipo en el título y el 90% no disponen de descripción.

Es necesario por tanto acordar etiquetas con la persona responsable de la administración de contenido en la AAA para que estos sean más descriptivos, por ejemplo usando etiquetas como “empelo teleco”, “empelo industrial”, etc.

4.4 El campo description

El RSS de antiguos alumnos, a diferencia de la mayoría de RSS actuales, añaden dentro del campo description una cantidad notable de código HTML y CSS para dar formato, se entiende, al contenido que se incluye dentro de description. Esto no es realmente útil, pues los lectores de RSS tienen ya normalmente un formato dado para la visualización del contenido y por otra parte, crea un problema de parsing de tediosa solución.

En el apartado 3.6.1 se detalla la solución dada y como se procesa el texto de la etiqueta descripción pero a día de hoy este proceso se sigue mejorando hasta que se consiga un resultado perfecto o bien, se cambie el sistema RSS ya obsoleto. Entre las opciones del sistema RSS de la versión de Drupal que se está usando no existe ninguna opción que permita evitar este problema.

4.5 Pruebas en dispositivos reales y emulados

Gracias al emulador del que dispone Android Studio se han podido emular varios dispositivos con diferentes versiones de Android para la prueba de la aplicación. Los dispositivos emulados son los siguientes:

- Nexus 5 – Con API 21 y 22
- Nexus 6P – Con API 23 y 24
- Nexus 4 – Con API 16,17,18, 19 y 20

En dispositivos reales, se tiene constancia de que se han instalado en modelos tan dispares como

- Samsung Galaxy S6 – API 23
- Meizu M2 Note – API 22
- Meizu M3 Note – API 23
- Samsung Galay A5 – API 22

Tanto en dispositivos reales como en los emulados, la aplicación ha funcionado correctamente y se han corregido pequeños errores producidos ante la falta de visualizadores PDF instalados. Los contenidos se muestran en los tamaños correctos en todos los dispositivos. En cuantro al rendimiento de la aplicación es muy similar en la gran mayoría de dispositivos. Algunos dispositivos más antiguos pueden mostrar un pequeño retardo de alguna de las animaciones pero sin que esto llegue a mermar la experiencia del usuario con la aplicación.

Por tanto, hemos abarcado la compatibilidad con todas las versiones entre Android 4.1 Jelly Bean y la actual Android 7.0 Nougat. Esto quiere decir que la aplicación es compatible con versiones del sistema del 2012 hasta la actualidad.

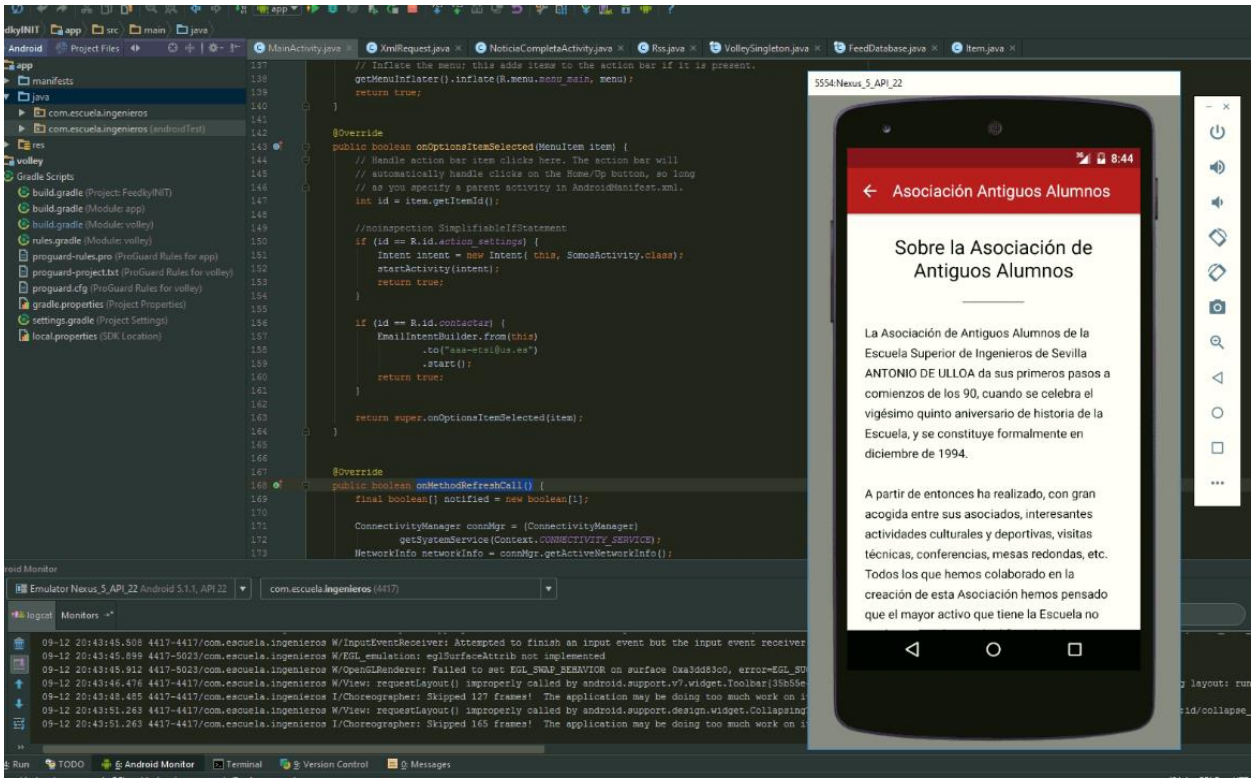

Ilustración 29 – Simulación en Android Studio

4.6 Google Play

La aplicación ya se encuentra disponible para todo el mundo en la tienda de aplicaciones de Android: Google Play a través del siguiente link https://play.google.com/store/apps/details?id=com.escuela.ingenieros . Para subir la aplicación ha sido necesario darse dalta como desarrollador, con un coste de 30€ y cumplir con una serie de requisitos a la hora de creación del contenido de la ficha para Google Play y en el código de la aplicación.

4.6.1 Generación de la aplicación firmada

Para la subida de una aplicación a la tienda aplicaciones primero es necesario que ésta sea compilada y firmada, para ello, Android Studio es genera archivo “key” (.jks) a raíz de los datos introducidos para su firma. Con la key y habiendose podido corregir los warnings y errores que este exigente compilado detecta, obtendremos nuestra apk lista para la subida.

Es importante realizar copias de seguridad de la key entregada pues no se puede volver a generar y si no se dispone en el futuro de ella, no se podrá volver a actualizar la aplicación en la tienda de aplicaciones, teniéndose que subir otra versión diferente duplicando así la aplicación en la Play Store y con el consecuente problema de actualización a los usuarios.

4.6.2 Google Developer Console

Google Developer Console es un portal donde los desarrolladores suben sus aplicaciones a la tienda Play Store, crean las fichas de información de dicha aplicación y consultan todo tipo de estadísticas relevantes para el desarrollo su desarrollo y seguimiento.

4.6.2.1 Clasificación de contenido

La aplicación está clasificada en la categoría "Noticias y Revistas" y se le han aplicado las siguientes clasificaciones de consumo. Se comprueba en la siguiente ilustración como se nos han otorgado las clasificaciones más sencillas según lo esperado para una aplicación de estas características.

Ilustración 30 – Clasificación de contenido

4.6.2.2 Compabilidad, instalaciones y actualizaciones

Según los datos del apartado APK y estadísticas de la Developer Console, la aplicación es compatible con 10801 modelos de dispositivos diferentes y ya se han llevado a cabo tres actualizaciones de ésta. Además se presentan los siguientes gráficos de instalaciones por versiones.

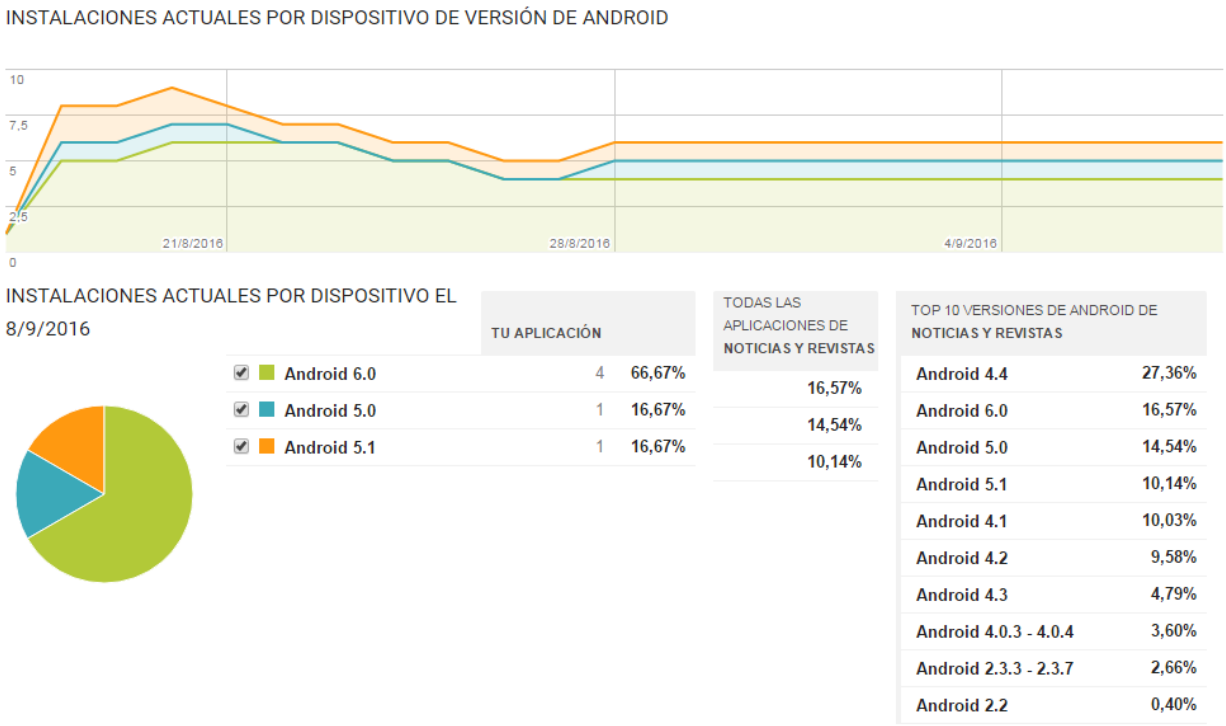

Ilustración 31 – Estadísticas de instalaciones por dispositivos

En el apartado de reporte de errores de usuarios no se ha detectado ninguno por lo que la aplicación muy probablemente esté corriendo sin problemas en los dispositivos de los usuarios.

4.6.2.3 Ficha técnica

La ficha de la aplicación requiere de una serie de imágenes para su promoción, se ha decidido dar un acabado más más profesional para las distintas ilustraciones, se han diseñado con Adobe Illustrator. Estos son algunos ejemplos.

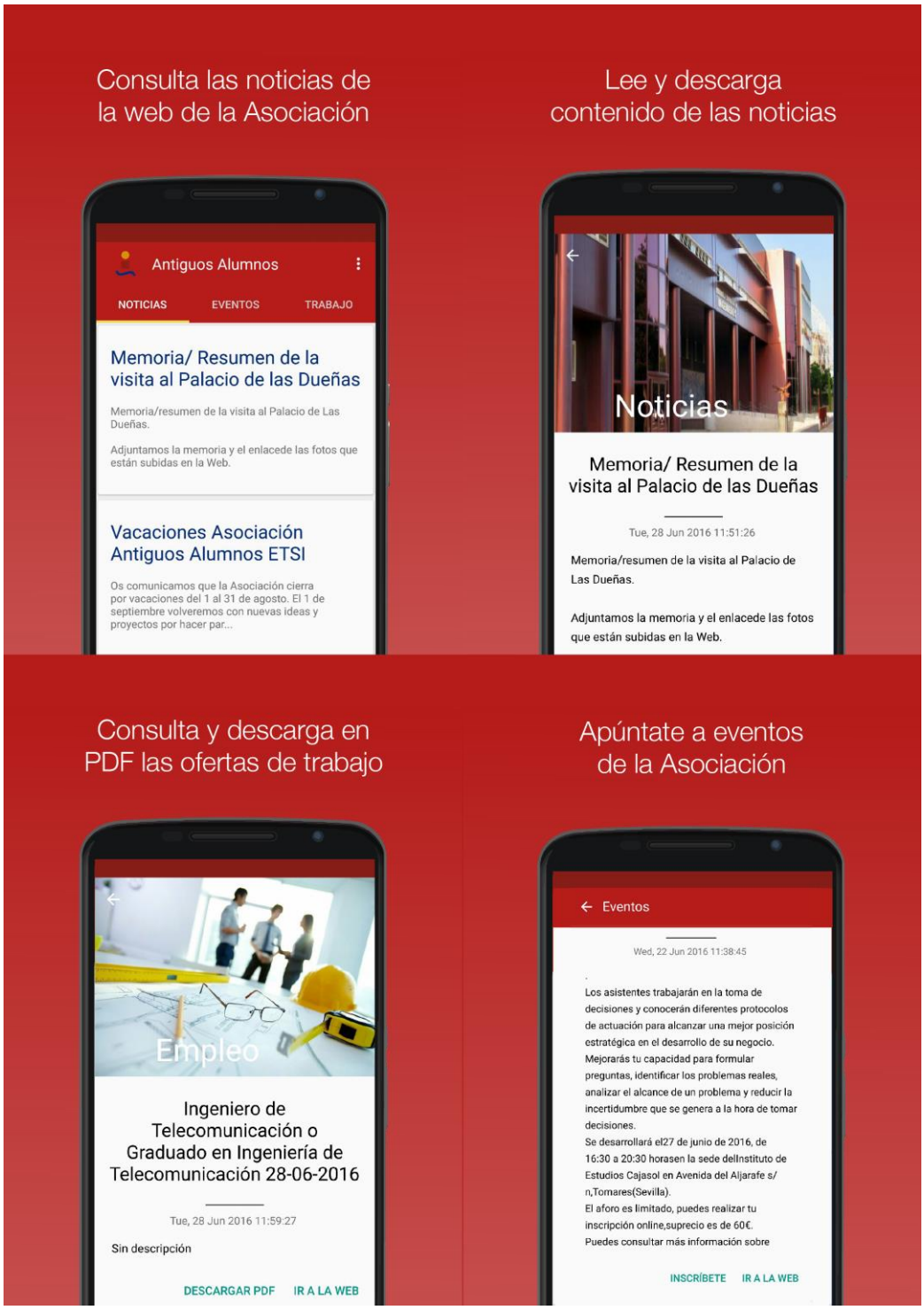

Ilustración 32 – Ilustraciones ficha técnica Play Store

Ilustración 33 – Ficha en Play Store

5 CONCLUSIONES Y LÍNEAS DE AVANCE

Existen líneas de trabajo que podrían cursarse en el futuro, algunas de ellas recaen sobre los administradores de la web de AAA mientras que algunas otras funcionalidades podrían llegar a ser programadas por el autor. A continuación se exponen las principales ideas planteadas.

5.1 La necesidad de un CMS y sistema RSS nuevo

Una de las principales condiciones para un mejor desarrollo y actualización de la aplicación en un futuro, pasa por la inevitable actualización de la web de la AAA, concretamiente de su CMS. Actualmente, las opciones de personalización del RSS dentro de la versión de Drupal actual (7.0) son muy pobres y éste se encuentra muy desorganizado.

El RSS no se encuentra dividido por canales sino que todos pertenecen al mismo, haciendo que sea más tediosa la gestión de contenido. Existen muchas secciones en la web de la AAA que podrían ser incluidas en la aplicación si estas tuvieran su propio canal RSS, como la revista Ingenio o la galería de fotos.

Podría crearse una más eficiente gestión de categorías y subcategorías gracias a un mejor uso de etiquetas en el RSS, dando así posibilidades a la aplicación de crecer y hacerse más eficiente. De esta forma la aplicación consumiría menos datos ya que las actualizaciones de contenido sería de secciones y no de todo el RSS, además, tardaría menos en realizar sus procesos de refresco.

5.2 Notificaciones push

Usando las notificaciones push, se pueden mandar recordatorios de vez en cuando, mejorando las posibilidades de que tu aplicación permanezca instalada en sus dispositivos. Podrían así los usuarios obtener información de las últimas ofertas de trabajo publicadas o eventos a organizarse.

5.2.1 Algunas ideas para su implementación

Google Cloud Messaging, abreviado GCM, es un servicio gratuito que puedes usar para enviar notificaciones a tus usuarios. En la mayoria de las comunicaciones cliente-servidor, el cliente inicia las peticiones para recibir datos del servidor. En otras palabras, el cliente es quien pide datos al servidor. En el caso de las notificaciones push, en cambio, es el servidor el que inicia la tranferencia de datos.

Esto generalmente se logra manteniendo una conexión persistente de tipo TCP/IP — una conexión que permanece abierta indefinidamente — entre el servidor y el cliente. Puede parecer estupendo, pero si tienes una app popular, mantener miles de conexiones persistentes entre tu servidor y los dispositivos de tus usuarios puede resultar muy caro.

Google Cloud Messaging es un servicio que resuelve este problema actuando como un intermediario entre el servidor y los dispositivos de tus usuarios. Con GCM, el Cloud Connection Server de Google, abreviado CCS, gestiona las conexiones persistentes por ti. También se asegura de que tus notificaciones push sean enviadas de forma segura. [14]

5.2.2 Funcionamiento

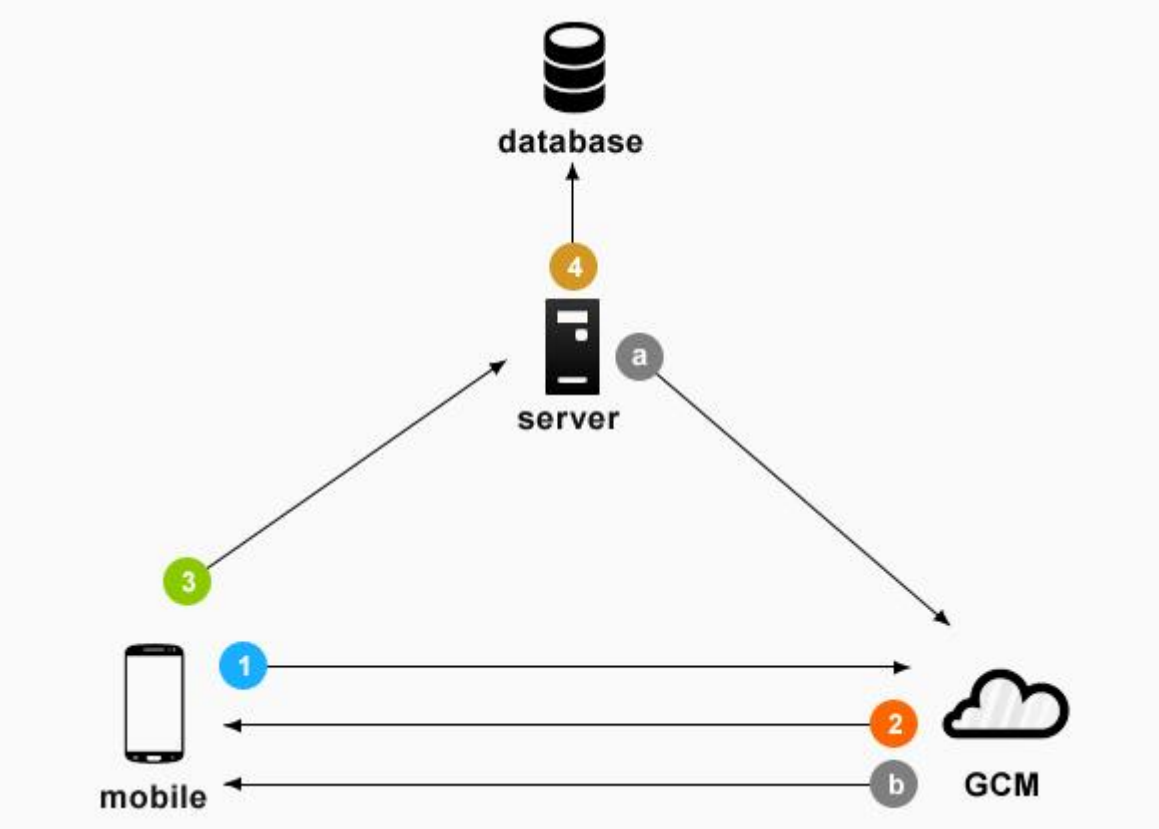

Ilustración 34 – Funcionamiento GCM

1. El dispositivo envía su ID y la ID de aplicación al sevidor GCM para su registro.
2. Si el registro se ha hecho correctamente, GCM devuelve la ID de registro al teléfono Android.
3. Despues de recibir la ID de registro, el teléfono envía dicho ID al servidor para su registro
4. El servidor almacena la ID de registro en la base de datos para su posterior uso
    a. Siempre que se necesitar realizar una notificación push, el server envía un mensaje al GCM server con la ID de registro.
    b. El servidor GCM manda la notificación push a los respectivos dispositivos móviles con la misma ID de registro recibida del servidor.

5.3 Actualizaciones y mantenimiento

El desarrollo de la aplicación no acaba con este proyecto pues su autor se compromete a darle un mantenimiento en función de las futuras necesidades que puedan ir apareciendo como:

- Nuevo RSS
- Adaptaciones a nuevas versiones de Android
- Reporte de problemas
- Nuevas secciones

5.4 Conclusiones

La aplicación cumple los requisitos para los que ha sido diseñada, proporciona una experiencia sencilla para el usuario y se puede hacer uso de todas sus funcionalidades sin la necesidad de ningún registro o login previo. Se han aprovechado todas las capacidades de Material Design para dotarla de un aspecto atractivo, limpio y moderno, usando animaciones e interfaces inteligentes para un mejor uso de la aplicación.

A falta de una promoción por parte de las administraciones pertinentes, la aplicación se encuentra en la Play Store, con valoración muy positiva y sin problemas de funcionamiento. Se espera que, siguiendo con un ritmo de actualizaciones, la aplicación pueda a extenderse y ser herramienta de referencia para la consulta de contenido de la AAA.

REFERENCIAS

[1] I. SPAIN, «Estudio Anual de Mobile Marketing 2015,» 2015. [En línea]. Available: http://www.iabspain.net/wp-content/uploads/downloads/2015/09/Estudio-Mobile-2015.pdf.

[2] Gartner, «Gartner,» 1Q 2016. [En línea]. Available: http://www.gartner.com/technology/topics/data-analytics.jsp. [Último acceso: 2016].

[3] Wikipedia, «Wikipedia - Sistema operativo móvil,» [En línea]. Available: https://es.wikipedia.org/wiki/Sistema_operativo_m%C3%B3vil#cite_note-2.

[4] IONIC, «IONIC,» [En línea]. Available: http://ionicframework.com/.

[5] Wikipedia, «Wikipedia - Android,» [En línea]. Available: https://es.wikipedia.org/wiki/Android.

[6] Wikipedia, «Wikipedia - Desarrolo de programas para Android,» [En línea]. Available: https://es.wikipedia.org/wiki/Desarrollo_de_programas_para_Android.

[7] Wikipedia, «Wikipedia - RSS,» [En línea]. Available: https://es.wikipedia.org/wiki/RSS.

[8] J. Revelo, «Hermosa programación - Bases de datos SQLite en Android,» [En línea]. Available: http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/.

[9] Wikipedia, «Wikipedia - Interfaz de usuario,» [En línea]. Available: https://es.wikipedia.org/wiki/Interfaz_de_usuario.

[10] sg.com.mx, «sg.com - La importancia de la interfaz usuario,» [En línea]. Available: http://sg.com.mx/revista/31/la-importancia-la-claridad-y-sencillez-una-interfaz-usuario#.V8MPaZiLTIU.

[11] Wikipedia, «Wikipedia - Material Design,» [En línea]. Available: https://es.wikipedia.org/wiki/Material_design.

[12] J. Revelo, «HermosaProgramación,» [En línea]. Available: http://www.hermosaprogramacion.com/2015/02/android-volley-peticiones-http/.

[13] Simple, «Simple XML,» [En línea]. Available: http://simple.sourceforge.net/home.php.

[14] A. Hathibelagal, «tutsplus,» [En línea]. Available: http://code.tutsplus.com/es/tutorials/how-to-get-started-with-push-notifications-on-android--cms-25870.

ÍNDICE DE CONCEPTOS