_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

PROGRAMACIÓN DE APLICACIÓN ANDROID PARA GESTIÓN Y ACCESO DE CONTENIDOS SANITARIOS DE CALIDAD

TRABAJO FIN DE GRADO

Autor: Víctor Barberá Lledó

Tutor: Dr. Vicente Traver Salcedo

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2013-14

Valencia, 24 de julio de 2014

Resumen

El presente documento ha sido realizado por Víctor Barberá Lledó en el grupo de Tecnologías de la salud y Bienestar (en adelante TSB) del Instituto de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones Avanzadas (en adelante ITACA), pertenecientes a la Universidad Politécnica de Valencia, siendo supervisado por el Doctor Vicente Traver Salcedo y apoyado por el equipo de investigación del ITACA.

En esta memoria se recorrerá el proceso de realización del proyecto, donde se provee una introducción inicial al contexto del proyecto, los objetivos del mismo, la metodología y planificación de trabajo, el desarrollo y resultados y finalmente, las conclusiones y propuestas de mejora.

El trabajo realizado consiste en la programación de una aplicación para smartphones Android con la que acceder a contenidos sanitarios de calidad, nutriéndose de la plataforma Salupedia, desarrollada por el ITACA en el año 2011, y que se dedica a proveer dicho contenido a los usuarios dese la web.

Resum

El present document ha estat realitzat per Víctor Barberá Lledó en el grup de Tecnologies de la salut i Benestar (d'ara endavant TSB) de l'Institut d'Aplicacions de les Tecnologies de la Informació i de les Comunicacions Avançades (d'ara endavant ITACA), pertanyents a la Universitat Politècnica de València, sent supervisat pel Doctor Vicente Traver Salcedo i recolzat per l'equip de recerca de l'ITACA.

En aquesta memòria es recorrerà el procés de realització del projecte, on es proveeix una introducció inicial al context del projecte, els objectius del mateix, la metodologia i planificació de treball, el desenvolupament i resultats i finalment, les conclusions i propostes de millora.

El treball realitzat consisteix a la programació d'una aplicació per smartphones Android amb la qual accedir a continguts sanitaris de qualitat, nodrint-se de la plataforma Salupedia, desenvolupada per l'ÍTACA a l'any 2011, i que es dedica a proveir aquest contingut als usuaris mitjançant la tradicional web.

Abstract

Víctor Barberá Lledó has written this document. The project has been done inside TSB' group, Tecnologías de la salud y Bienestar, that belongs to ITACA, Instituto de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones. Doctor Vicente Traver Salcedo, professor of the Polytechnic University of Valencia, has supervised the project. ITACA's engineers supported the project and offered help when needed.

It is shown in this report the development process of the project that includes an introduction to the project context, the objectives, the methodology and planning, the development and results and finally the conclusion and improvements.

The main topic of the project is the software development of an Android application that can be used to manage and obtain quality health contents. The information provided is obtained by Salupedia, a web developed by ITACA in 2011.

Índice

Capítulo 1 – Introducción

1 – Introducción

1.1 - Antecedentes

En el contexto sanitario español actual, la crisis económica ha mostrado la necesidad de transformación del Sistema Nacional de Salud. Desde hace años son bien conocidas las dificultades financieras del sistema sanitario, cuyo gasto crece a un ritmo superior que la economía. La mayor utilización de los servicios de salud, junto con el gasto farmacéutico, la inflación de los precios y la poca eficiencia del sistema, explican el nuevo contexto. No obstante, el desarrollo y la difusión de las **nuevas tecnologías** son un gran activo que puede ayudar a mejorar la viabilidad del sistema.

La tecnología sanitaria ha experimentado una evolución extraordinaria en las últimas décadas, y ha contribuido de un modo determinante a la mejora de la salud de la población.

No obstante, también se considera a la misma como una de las principales causas del incremento del gasto sanitario, en buena medida debido a su uso inadecuado, cuando ligado a su desarrollo y difusión se produce un incremento en las indicaciones de procedimientos médicos y quirúrgicos **no adecuados**, prescripciones farmacéuticas de complacencia o una ampliación de la población que recibe tratamientos no necesarios.

También se ha detectado la introducción de nuevas tecnologías con limitada evidencia sobre su eficacia, a lo que debería añadirse un efecto bien conocido, y es que las nuevas tecnologías no siempre sustituyen a las anteriores, si no que se añaden a ellas.

Es por ello que se hace obvio que el contexto sanitario necesita de un soporte tecnológico útil, de confianza y que apoye las técnicas habituales y genere unos beneficios sociales a la población y un ahorro al sistema. El presente proyecto está basado en una de las soluciones que pretenden apoyar al sistema sanitario mediante el uso de **internet** y de las **aplicaciones móviles**, con especial cuidado en la verificación de los contenidos publicados.

Es un hecho que el mayor proveedor de contenidos actualmente es internet, ya que ha cambiando y facilitado la forma de entender e interactuar con la información de las personas. Su expansión a lo largo de los últimos años ha seguido un crecimiento exponencial, siendo usado por millones de usuarios para comunicarse, entretenerse y acceder a contenidos de muy variado carácter, entre ellos los **contenidos sanitarios**.

Acorde con el estudio realizado por la organización "Pfizer" [1], ocho de cada diez usuarios consultan en internet antes y/o después de una consulta médica, situándose como primera vía de obtención de información complementaria a la proporcionada en la consulta médica o farmacéutica. Por otra parte, también denota la ausencia de sitios web específicos donde encontrar la información rigurosa y adecuada, ya que el usuario acaba consultando portales web que recogen contenidos de diversas clases, no estrictamente especializados en contenidos sanitarios.

Junto con lo anteriormente expuesto, las propias facilidades que proporciona internet también generan desconfianza en dichos contenidos, ya que la información es fácilmente accesible y en algunos casos, modificable. Por ejemplo, en la popular página "Wikipedia" (cuyos contenidos pueden ser modificados por cualquier usuario de internet), el 90% de contenidos acerca de salud y enfermedades contiene errores [2].

Con estos datos, en el ámbito de la salud surgía la necesidad de encontrar un portal de confianza donde tener la seguridad de que la información que se consulta es correcta y que provee de unas fuentes fiables. Dicha información, debería ser revisada y aprobada para su publicación con el fin de la tranquilidad del usuario, que busca resolver sus inquietudes desde una vía de confianza.

Esta necesidad fue explotada por el equipo de ingenieros del Instituto de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones Avanzadas, perteneciente a la Universidad Politécnica de Valencia, mediante el proyecto **Salupedia**, desarrollado entre los años 2011 y 2014 y que está en funcionamiento actualmente. Su *URL* es "http://www.salupedia.org".

1.2 - Salupedia

Salupedia es una web que proporciona un acceso seguro a información sanitaria en internet. Está basada en la idea de crear una comunidad donde profesionales del ámbito de la sanidad recomiendan contenidos, ya existentes en la red, a pacientes, familiares y ciudadanos en general. Todos los artículos que aparecen en Salupedia son revisados por profesionales de la salud que

son especialistas en el carácter de cada artículo. De este modo, se evita la habitual "desconfianza" en los contenidos que se obtienen de internet. A su vez, Salupedia también ofrece servicios para el profesional de la salud, donde puede dirigir a sus pacientes cuando desea prescribir información relacionada con su patología. De este modo, se puede ahorrar mucho tiempo invertido en las salas de espera, y los profesionales de la salud se pueden centrar en los casos de mayor importancia, suponiendo esto un ahorro para el estado ya que se mejora con creces la eficiencia del sistema.

Como se puede apreciar en la figura 1, el portal web ofrece una serie de artículos contrastados con título, una breve descripción, su fecha de publicación y una pequeña imagen, todo dispuesto para que escojamos leer los artículos que nos interesen.

Figura 1 – Un artículo típico en Salupedia

La página presenta seis categorías principales, pese a que su base de datos maneja en su totalidad hasta 100 ítems. Sus seis categorías principales son:

- Asuntos sociales y familia.
- Enfermedades o problemas de salud.
- Pruebas, procedimientos y diagnósticos.
- Salud y tecnología.
- Tratamientos y consejos.
- Vida sana y prevención de la enfermedad.

Sin embargo, los nuevos estándares de consumo indican que el usuario tiende a acceder a los contenidos desde su *Smartphone*, cada vez con mayor frecuencia. Por ello, es interesante

tratar de dar un paso más hacia el usuario, acercando Salupedia a su día a día mediante el desarrollo de una aplicación móvil de fácil acceso, con la que facilitar el consumo y la gestión de contenidos que Salupedia proporciona a los usuarios de las nuevas tecnologías.

El proyecto que a continuación se expondrá y describirá se centra en el desarrollo y la programación de dicha aplicación, proporcionando al portal Salupedia ya existente, una interfaz móvil de sus contenidos.

Capítulo 2 - Objetivos

2 – Objetivos

2.1 – Objetivo principal

El objeto del presente proyecto es **adaptar los contenidos que Salupedia ofrece a los terminales móviles mediante una aplicación Android**. Por ello, el objetivo principal del proyecto consiste en programar dicha aplicación. Con el fin de que la aplicación pueda cubrir este objetivo, debe seguir las siguientes características:

- **Facilidad de uso**

Dado que la gama de usuarios que consultan información sanitaria es muy amplia, se recogen edades de carácter muy variable, y en consecuencia habilidades tecnológicas dispares. Es por ello que la aplicación ha de ser programada "desde el caso peor" y debe ser sencilla para que su uso no suponga una barrera para el usuario poco habituado a esta tecnología.

- **Rápidez**

La aplicación ha de responder interactivamente sin suponer un coste de tiempo adicional para el usuario. Se pretende facilitar el contenido en el menor tiempo posible para permitir lecturas agradables.

- **Robustez**

Una característica principal de cualquier aplicación debe ser la robustez. Se pretende conseguir la ausencia de fallos y evitar el cierre repentino de la aplicación. El usuario ha de obtener la sensación de seguridad usando la aplicación.

- **Utilidad**

El servicio que se provee ha de ser lo suficiente completo, permitiendo al usuario obtener una experiencia lo más similar posible a la que la web actualmente provee.

2.2 – Objetivos secundarios

Continuando en la línea tecnológica, la aplicación debe estar diseñada para su posible modificación y ampliación. Su código deberá ser claro, espaciado y deberá contener comentarios explicativos, preferentemente con javadoc, para que en el caso de ser requerida una ampliación, facilitar el trabajo al profesional que se enfrente a este código.

Por otra parte, como objetivos secundarios no relacionados con la tecnología, se pretende el aprendizaje del desarrollo de un proyecto cubriendo todas sus fases, desde la documentación, estudio de situación previa y necesidades hasta finalmente su programación y testeo.

Capítulo 3 – Metodología

3 – Metodología

En este apartado se procederá a exponer la metodología utilizada para cada uno de los apartados que componen este proyecto, así como las herramientas utilizadas para llevarlos a cabo. Se explicará la distribución temporal de las tareas, exponiendo cómo se ha gestionado el proyecto de principio a fin en cada una de sus fases.

3.1 – Herramientas utilizadas

3.1.1 – Plataforma escogida

La plataforma que se escogió para programar la aplicación móvil fue **Android**. Inicialmente fue la plataforma propuesta ya que Android dispone de la mayor cuota de mercado como sistema operativo móvil, en concreto el 75% [Figura 2]. De este modo se consigue que la aplicación pueda llegar al máximo número posible de usuarios. Además, dispone de un entorno de desarrollo gratuito y disponible para cualquier sistema operativo y cuenta con un gran volumen de documentación en internet.

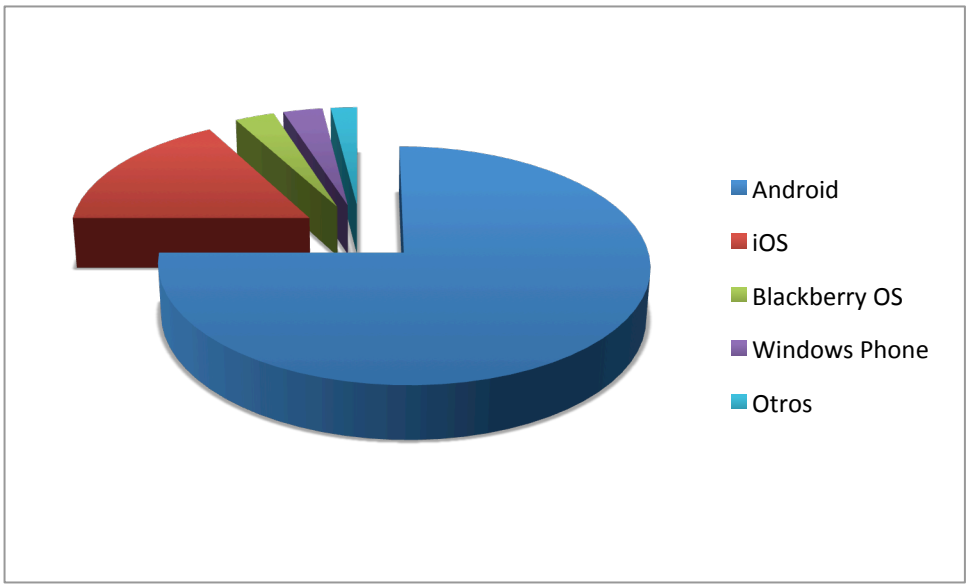

Figura 2 – Cuota mercado sistemas operativos móviles, Junio 2013

3.1.2 – Entorno de desarrollo

El entorno de desarrollo utilizado ha sido **Eclipse**. Se eligió Eclipse por que es el recomendado por Google, propietario de Android, plataforma para la que se programó la aplicación. Se puede obtener de forma gratuita desde el sitio web de desarrolladores para Android. Al entorno de desarrollo se le añade el plug-in **ADT** (Android Developer Tools), que se encarga de añadir funcionalidades propias de Android al IDE.

Se escogió este entorno de desarrollo por que aportaba más funcionalidades que otros más sencillos como, por ejemplo, App Inventor. Este entorno queda muy limitado ya que está enfocado a la programación visual y no a la escritura de código.

Otro entorno posible habría sido Android Studio, entorno de desarrollo en el cual Google está trabajando, pero que por su juventud se trató de evitar, ya que Eclipse aportaba mayor robustez.

3.1.3 – Entorno de pruebas

Para comprobar el funcionamiento de la aplicación, se usaron dos vías:

- **ADV** – Android Virtual Device es un emulador que permite reproducir el comportamiento de un terminal Android real. Fue escogido ya que permite escoger los parámetros del mismo, tales como el tamaño de la pantalla, la versión de sistema operativo Android o incluso escoger entre Smartphone o Tablet, y es una solución que no requiere conectar ningún terminal real.
- **Terminal real** – En este caso se utilizó un Motorola Moto g, con versión Android 4.4 KitKat y pantalla de 4,5”. Se optó por complementar al ADV ya que se obtiene una experiencia más cercana a la realidad.

3.1.4 – Lenguajes de programación

El lenguaje de programación principal utilizado para el desarrollo ha sido **Java**, ya que es el predeterminado por Android para definir el comportamiento y las acciones a realizar por la aplicación.

Para los campos gráficos, Android usa **XML**, lenguaje de etiquetas que se encarga de gestionar los apartados gráficos de la aplicación como los campos de texto, botones, listas de elementos, imágenes…

El lenguaje usado para el acceso y la gestión de la base de datos de Salupedia ha sido **PHP**

ya que fue el lenguaje usado por los creadores del portal web para gestionar su base de datos, además de disponer de métodos enfocados a la gestión de bases de datos y la ejecución de sentencias **SQL** (nótese que la base de datos de Salupedia es **mySQL**). Para la gestión de información devuelta por las consultas SQL, se usó el formato **JSON**, ya que es muy intuitivo y fácil de tratar debido a que trabaja con etiquetas y dispone de métodos predefinidos en Java para ser tratado.

3.1.5 – Entorno de gestión de la base de datos

Con el fin de comprender y analizar la estructura de la base de datos ya creada de Salupedia, se procedió a utilizar el software gratuito **Heidi SQL**. Este software muestra de forma sencilla y visual las tablas de cualquier base de datos (siempre disponiendo de los credenciales necesarios) y su contenido. Además, permite obtener el resultado de cualquier sentencia SQL ejecutada. Este software se hacía necesario ya que para obtener los datos deseados para el usuario se ha de conocer la estructura de la base de datos y los resultados de las sentencias.

3.1.6 – Distribución del proyecto y redacción de memoria

Con el fin de evaluar y obtener un gráfico visual del tiempo dedicado a cada proceso que compone el trabajo de fin de grado, he usado el software **Microsoft Project** ya que es el software referente para la gestión de proyectos utilizado globalmente, y provee unas herramientas intuitivas que facilitan la explicación a lo largo del tiempo de la gestión del proyecto. La memoria se escribió con **Microsoft Word** y la presentación se realizó con **Microsoft PowerPoint**.

3.2 – Distribución en tareas

El proyecto ha estado compuesto por cinco tareas principales en las que se ha distribuido su ejecución a lo largo del ciclo de trabajo. Se han incluido los apartados principales cuando se trabaja con un proyecto que recoge la programación software de una aplicación: la **documentación**, la **programación** y el **testeo** de la aplicación para comprobar los resultados. Complementando estos tres apartados, también se ha trabajado en la **recopilación** de la información para generar la memoria del trabajo de fin de grado y las **reuniones** con el tutor. Los porcentajes de tiempo dedicado a cada tarea se pueden observar en el apartado 3.3 diagrama

temporal, ya que se provee un documento de Microsoft Project donde se detalla el número de días dedicados a cada tarea. A continuación se detalla cada una de las tareas realizadas.

3.2.1 - Reuniones con el tutor

Conforme el proyecto ha ido avanzando, se han establecido reuniones con el tutor del trabajo que venían marcadas por el progreso en el proyecto. Cuando una tarea propuesta era finalizada se organizaba una reunión para discutir los avances, proponer mejoras y decidir la siguiente acción a realizar. Se dispuso de esta manera ya que el trabajo conlleva una amplia carga de documentación y programación, por lo que es conveniente reunirse con los resultados en mano para proponer mejoras y corregir lo realizado.

3.2.2 - Documentación

La documentación ha sido un apartado crucial en la realización de este proyecto. Dado que Android es un software libre, y los recursos necesarios para desarrollar aplicaciones en esta plataforma son gratuitos, existe en internet una cantidad ingente de información de la que poder nutrirse. No obstante, esta facilidad de obtención de información conlleva también una gran labor de selección de la información que se consulta, ya que muchas veces lo consultado no es estrictamente lo deseado. Por ello, la mayor parte de la documentación extraída ha provenido de los apuntes de la asignatura Aplicaciones Telemáticas [3] y sistemas Complejos Bioinspirados [4], cursadas ambas en el segundo cuatrimestre del presente curso universitario, ya que aportan contenidos generales, entendidos de ante mano y que son útiles para el diseño básico de cualquier aplicación.

3.2.3 – Programación

A priori la tarea principal del proyecto. Fruto de las reuniones periódicas con el tutor del proyecto, surgieron tres métodos para la obtención de información a programar. En primer lugar, se trabajó en conseguir mostrar un lector de las noticias que Salupedia mostraba de una forma visual para la pantalla de un Smartphone. Dado que Salupedia dispone de un lector de Feeds codificado mediante el lenguaje de etiquetas XML, me decanté por obtener las noticias directamente de este fichero disponible en internet. Mediante las funciones de XML Parser que

Android ofrece, recorrí el fichero XML y guardé la información en vectores de cadenas de texto, que luego mostraba.

Tras la correspondiente reunión con el tutor, se propusieron una serie de mejoras en la aplicación que llevaron al segundo gran bloque de programación. Se añadió un submenú que añadía dos nuevas funcionalidades. La primera, acceder a una descripción de Salupedia que indique al usuario información básica sobre la página. La segunda, un botón para abandonar y cerrar finalmente la aplicación. A parte de este menú, también se añadió un visor web mediante el cual, pulsando el botón indicado, la aplicación nos redirige a la página web oficial de Salupedia, para solventar posibles acciones que el usuario desee hacer que la aplicación no le permita, o simplemente saciar su curiosidad si es que desea acceder a la versión web.

Por último, se pretendió añadir la funcionalidad de filtrar por categorías, por lo que se requirió el acceso directo a la base de datos de Salupedia. Para ello, se programaron dos nuevas clases que gestionaron la conexión con la base de datos mediante peticiones http, y permitía tratar la información devuelta por el servidor en formato JSON. Para ello, se ubicó en el servidor de Salupedia un fichero PHP que gestionaba los credenciales de acceso a la base de datos, realizaba la consulta SQL deseada y devolvía la información en un fichero JSON, que luego el programa en Android gestionaba para sacar la información deseada. Al permitir utilizar sentencias SQL, se amplió en consideración las funcionalidades que la aplicación podía llevar a cabo.

Todo el proceso seguido en la programación se explicará en detalle en el apartado cuatro de la memoria.

3.2.4 – Testeo de la programación

Es necesario, en cualquier proyecto que incluya la programación de software, invertir una cantidad considerable de tiempo en la comprobación y el testeo del software que se programa. El software debe ser eficaz, útil y robusto para que la aplicación responda de la manera deseada. Además, el código ha de ser legible e inteligible para que en un futuro posibles colaboradores del proyecto puedan entender el trabajo realizado hasta ahora. Por ello, a cada paso que se programaba, se comprobaba que la aplicación respondía adecuadamente a lo requerido, era intuitiva y no se colgaba al descargar los ficheros.

Se ha programado de forma que cada vez que se caiga en una excepción (en los bloques try/catch) monitorice el texto que dicha excepción provee por la consola del programador, para

entender los posibles fallos en los que la aplicación entra. También se ha cuidado evitadar los bucles infinitos.

3.2.5 – Recopilación información

Por último, se hace necesario la recopilación de la información, tanto a nivel de código como a nivel propio de la memoria del trabajo de fin de grado. A nivel de código, se ha usado la regla de javadoc mientras se programaba, la cual genera información automática sobre las distintas clases y métodos del código y la presenta de una forma visual e intuitiva. El código también ha sido comentado para clarificar su funcionamiento a cualquier persona que pueda trabajar con él. La realización de la memoria y de la presentación se empezó cuando el código estaba acabado, sólo pendiente de mejoras y la aplicación ya tenía sus funcionalidades básicas.

3.3 – Diagrama temporal

Las tareas expuestas previamente se exponen a continuación en su distribución temporal aproximada. Como se puede apreciar en la figura 3, la mayor parte del tiempo de ejecución del trabajo de fin de grado se ha dedicado a la obtención de documentación, seguida por la programación, abarcando entre las dos un 60% del tiempo invertido en el proyecto. El testeo del código ha recogido también una cantidad importante del tiempo, con un 12%, aumentando el porcentaje de estos tres pilares al 72% de la realización del proyecto. El hecho de que estos tres apartados hayan dispuesto de tanto tiempo es debido a que el trabajo se ha focalizado en la obtención de la aplicación y su funcionamiento, por lo que se requiere una inversión muy importante en los tres pilares que componen los proyectos de software, como previamente se comentaba.

TAREA	PORCENTAJE
REUNIONES CON TUTOR	5,52%
DOCUMENTACIÓN	35,86%
PROGRAMACIÓN	25,52%
TEST CÓDIGO	12,41%
RECOPILACIÓN INFORMACIÓN	20,69%

Figura 3 – Tabla con porcentajes dedicados a cada tarea

En las figuras cuatro y cinco, se muestran los porcentajes anteriormente analizados en un gráfico circular, donde aparece en la primera las cinco tareas diferenciadas, y en la segunda los dos subgrupos de tareas, las relacionadas con la programación y las relacionadas con la recopilación y reuniones.

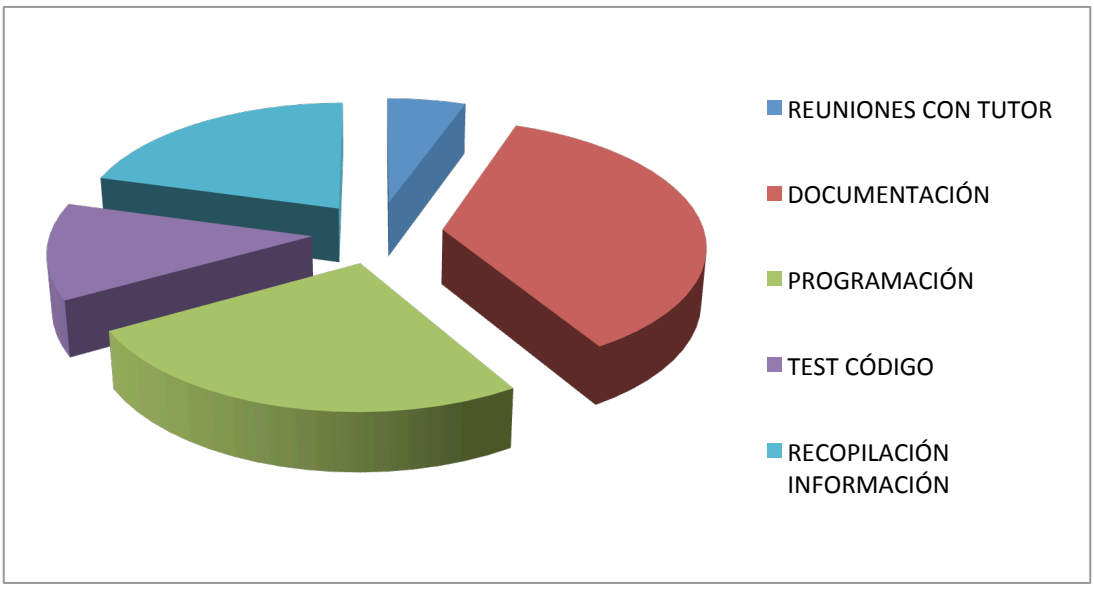

Figura 4 – Gráfico con distribución de tareas

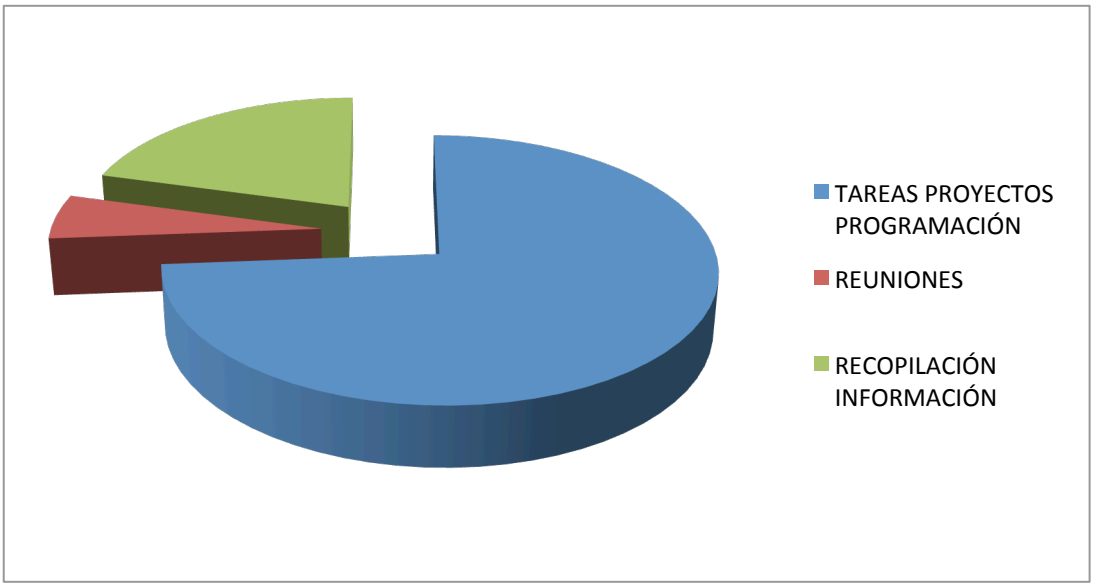

Figura 5 – Gráfico con distribución de tareas agrupadas

A continuación se muestra el diagrama de Grantt asociado al proyecto realizado:

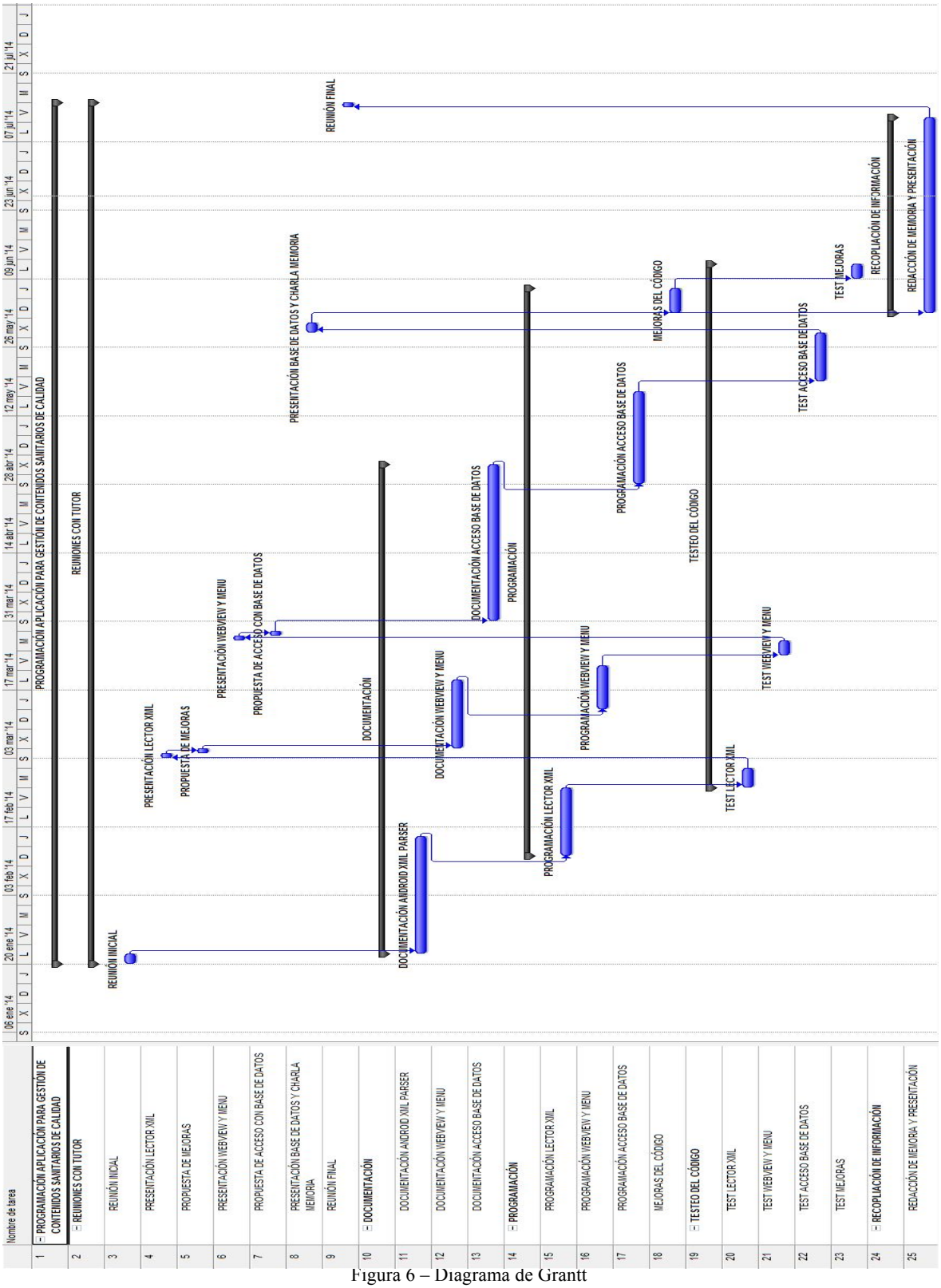

Figura 6 – Diagrama de Grantt

Capítulo 4 – Desarrollo y resultados del trabajo

4 – Desarrollo y resultados

En este capítulo se va a especificar el proceso completo de desarrollo del trabajo de fin de grado, cuyo autor ha sido un servidor, Víctor Barberá Lledó, junto con su correspondiente tutor Vicente Traver Salcedo. El desarrollo ha consistido en la **programación de la aplicación para gestión de contenidos sanitarios de calidad**. En este apartado se detalla para cada apartado de la aplicación cómo se ha afrontado la solución de la misma, y cómo se ha llevado a cabo. Además, se mostrarán los resultados obtenidos y el rendimiento de la aplicación general mediante diversos métodos como capturas de pantalla, diagramas explicativos y pequeños fragmentos del código desarrollado.

Los apartados en los que se subdivide la explicación de este capítulo, incluyendo el desarrollo y los resultados, son los siguientes:

- Diseño técnico.
- Características aplicación.
- Vista principal.
- Lector XML.
- Webview.
- Menú aplicación.
- Acceso base de datos.

4.1 – Diseño técnico

Previamente a la programación de cada apartado, se ha realizado un boceto previo a mano sobre papel, con el fin de conseguir una esquematización de las ideas y clarificar la orientación a la programación de la solución requerida. Se recogieron los campos necesarios en cada vista de la aplicación, así como la relación entre ellas.

A continuación se recoge una descripción de la totalidad de los elementos que se han utilizado con el fin de familiarizar con los términos que aparecerán en la explicación del desarrollo de los distintos apartados en los que se ha programado la aplicación:

- **Activity**: Una Activity es una vista cualquiera de la aplicación, recogiendo todos los campos que se muestran en la pantalla con la que interactuamos en un momento dado. Los ciclos de vida de una Activity son:
    - **onCreate**: La activity se crea, inicializa y carga los campos gráficos XML. Puede recibir datos de una activity anterior mediante una instancia de la clase Bundle.
    - **onStart**: La activity está preparada para ser mostrada al usuario.
    - **onResume**: Paso previo de la activity a ser ejecutada. Es utilizado para introducir música o animaciones en la activity.
    - **Running**: La activity está ejecutándose.
    - **onPause**: La activity va a ser lanzada a segundo plano, generalmente por que otra activity va a ejecutarse. Puede volver a onResume para volver a ejecutarse.
    - **onStop**: La activity deja de ser visible pero se mantiene en segundo plano, permitiendo volver a onStart mediante onRestart. Nótese que si el terminal tiene memoria limitada puede no entrar en onStop y destruirse directamente.
    - **onRestart**: Permite a la activity la transición de onStop a onStart.
    - **onDestroy**: La activity es “destruida” y deja de consumir recursos.

  En la figura 7 se observa la explicación de la transición entre ciclos de una activity de forma gráfica.

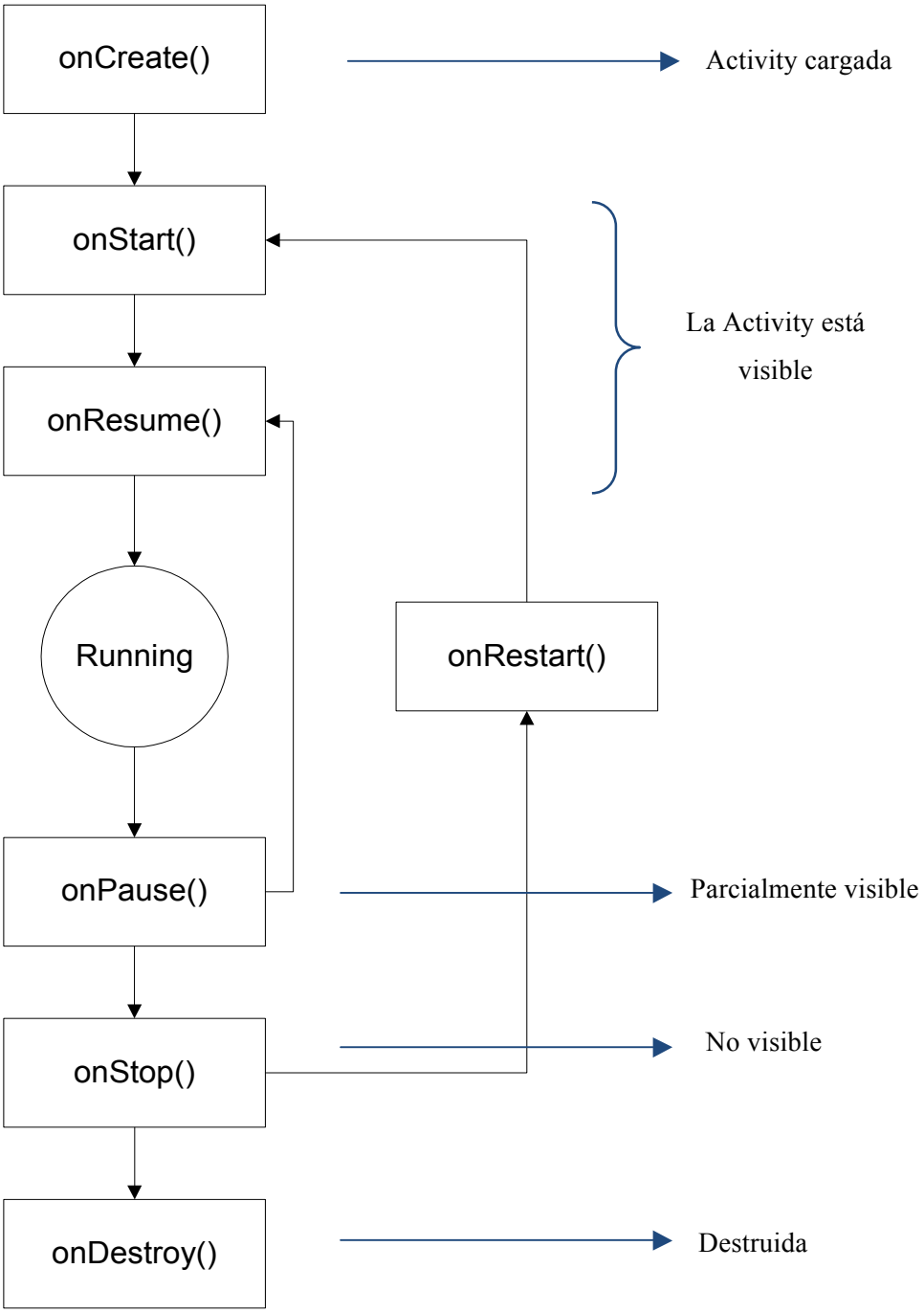

Figura 7 – Ciclo de vida de una activity

- **Intent**: Un intent es una acción ejecutada en el código de la aplicación que permite cambiar la actividad que se está mostrando en pantalla.
- **Bundle**: Un bundle es un "recipiente" a nivel de código en el que se añaden los contenidos que se desean enviar de una activity a otra. Un bundle va incluido en in intent.

En la figura 8 se observa un ejemplo de transición entre dos activities. La transición se genera mediante un intent, que llevará incluido un contenedor bundle con el nombre (en este caso particular Paco) a mostrar.

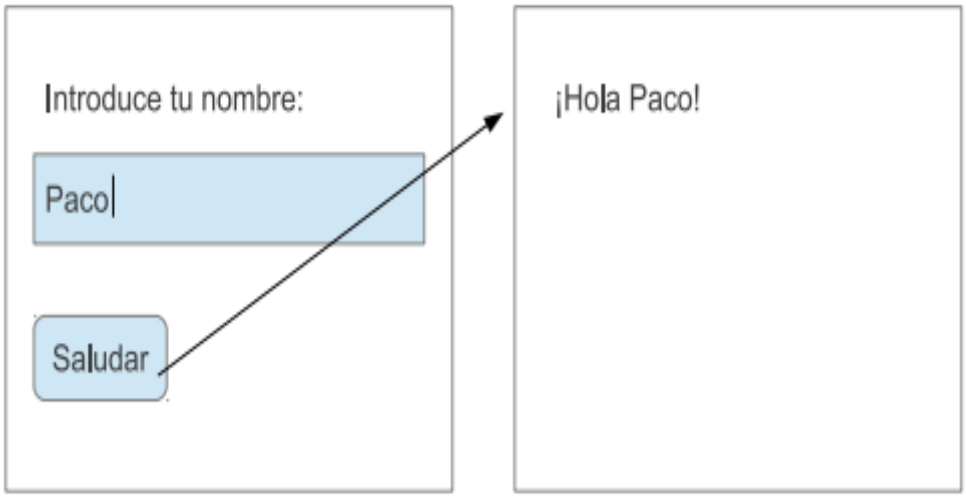

Figura 8 – Ejemplo de dos Activities

- **Layout**: Un layout es un método de distribución de los distintos componentes gráficos que componen una activity. A continuación se describen los principales tipos de layouts:
    - **Relative Layout**: Establece las posiciones de los distintos elementos en función de otros elementos ya existentes o de los márgenes de la pantalla. Ha sido el escogido ya que simplifica la distribución y facilita la equidistancia de los elementos.
    - **Absolute Layout**: Establece posiciones absolutas de los componentes.
    - **Linear Layout**: Establece secciones fijas en forma de línea para los componentes.
    - **Table Layout**: Establece los componentes en formato fila y columna.

- **TextView**: Un textview es un campo gráfico que muestra texto. Es utilizado para sacar la información de los artículos, como por ejemplo los títulos de las noticias.
- **ListView**: Un listview es una lista de información, organizada por ítems, de cadenas de texto. Permite hacer click en un ítem y programar la consecuencia de dicho click.

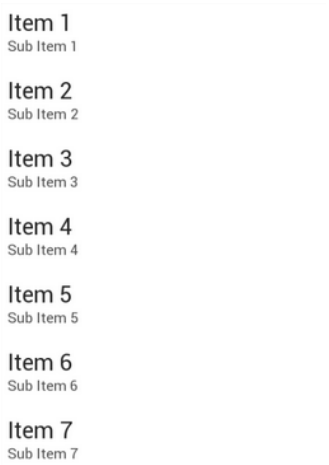

Figura 9 – Ejemplo de listview

- **Spinner**: Un spinner sigue el patrón de un listview pero agrupa sus elementos si no está seleccionado.
- **Button**: Un button es un botón clásico, que se usa como pasarela para realizar una acción determinada.
- **ImageView**: Un imageview muestra una imagen determinada en la pantalla de la aplicación.
- **WebView**: Un webview muestra en la activity una página web determinada, que se ha de indicar mediante código al objeto webview.

4.2 – Características de la aplicación

Las versiones de Android para las que la aplicación se ha programado son a partir de la 4.0, ya que las versiones anteriores están obsoletas en el mercado y limitan la aplicación a la hora de programar ciertos métodos que se incluyeron en las nuevas versiones.

La aplicación ha requerido únicamente que se le de permiso de acceso a internet, con el fin de poder acceder a los contenidos.

La aplicación ocupa 5.55 Mb, lo cual no la hace en exceso "pesada" para los terminales con poca memoria. Como se observa en la figura 10, es un peso considerablemente más bajo que la media.

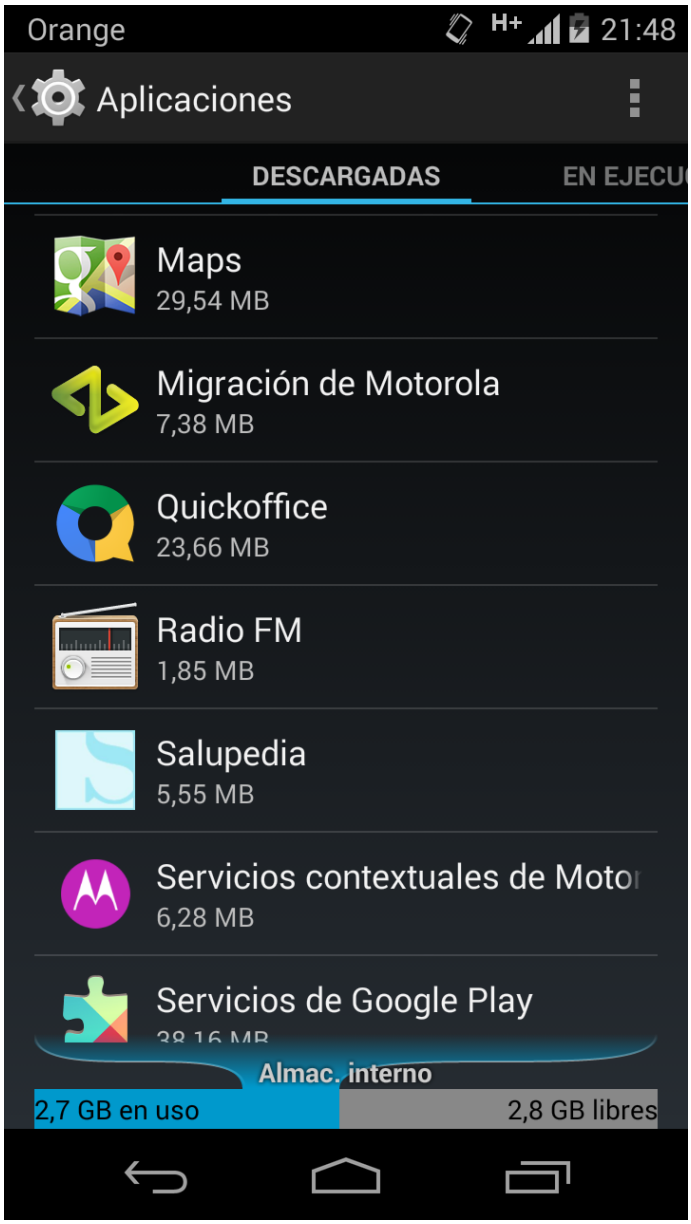

Figura 9 – Comparativo tamaño aplicaciones.

Además, la aplicación esta traducida a español e inglés, siendo el idioma principal del terminal móvil el que marca el idioma en el que se muestran las aplicaciones. Esto significa que si el usuario tiene su terminal en inglés, la app de Salupedia se mostrará en inglés. En las figuras 11 y 12, se ha configurado el terminal para que su idioma principal sea el inglés, y por ello al ejecutar la aplicación se observa, tanto la vista principal como el menú, en inglés.

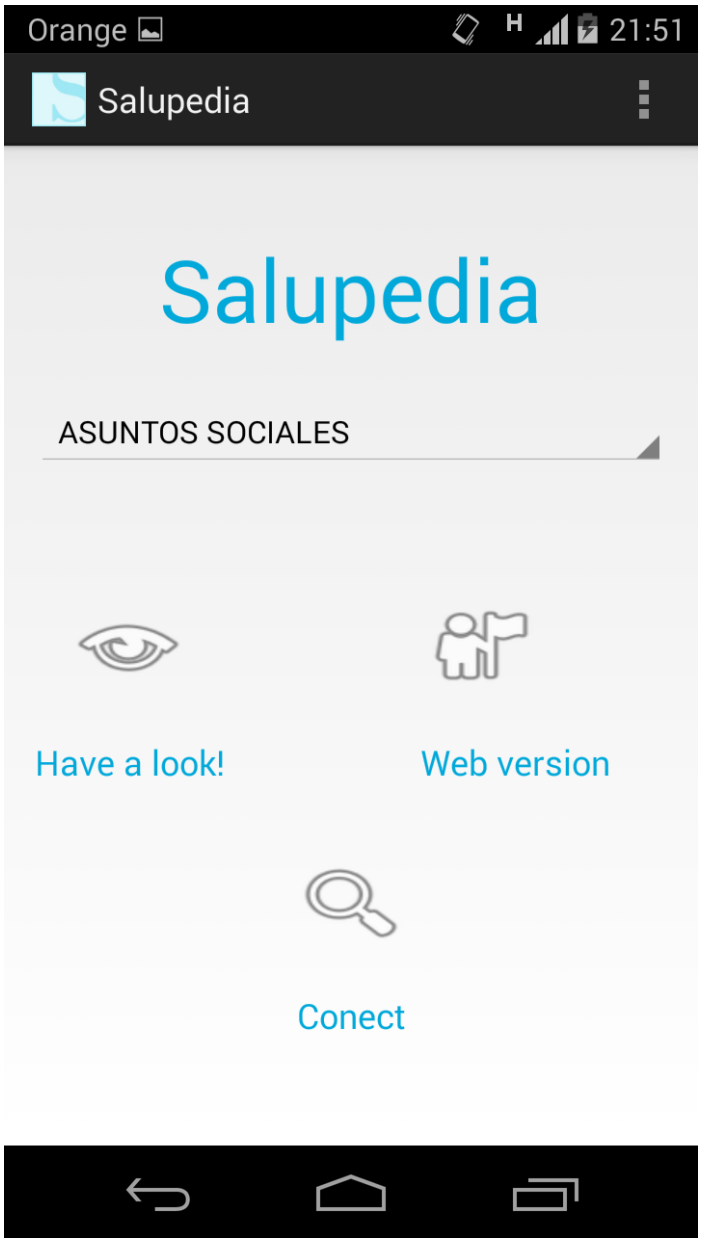

Figura 11 – Aplicación con Android configurado en inglés.

Por último, conforme se ha programado la aplicación, sea ido comentando cada clase y cada método para obtener la correspondiente documentación oficial **javadoc**, una facilidad que proporciona el lenguaje de programación para recoger y documentar toda la información acerca del código diseñado.

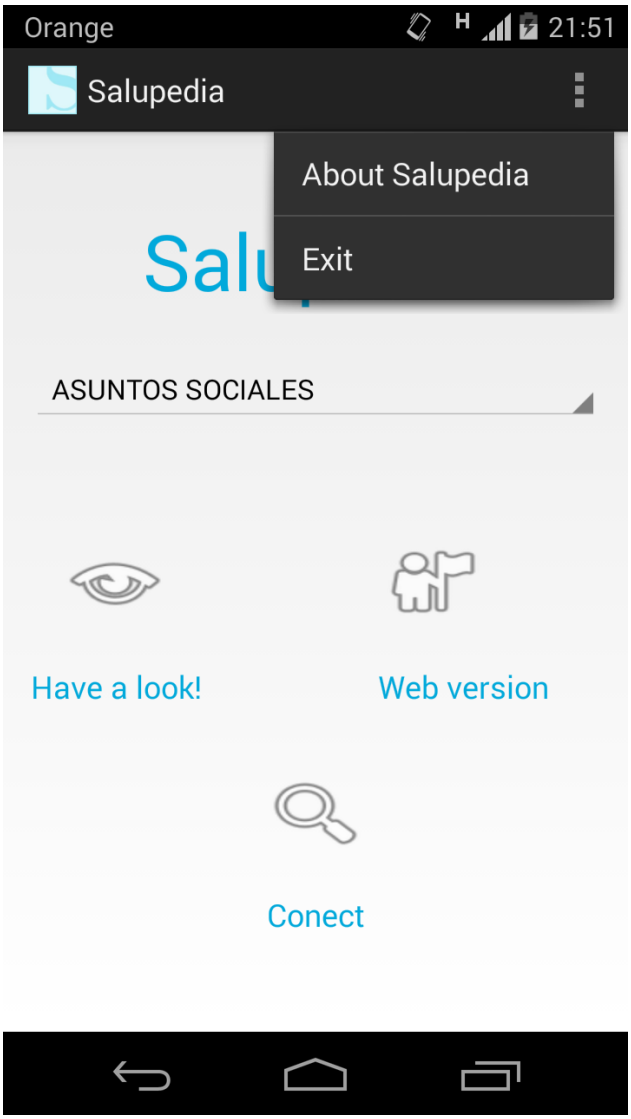

Figura 12 – Menú con Android en inglés.

Esto se ha conseguido ya que Android facilita la posibilidad de guardar todas las cadenas de texto en un fichero de **cadenas de texto**, que se recoge en la carpeta del proyecto "Values". Así pues, cuando en el código se ha hecho referencia a una cadena de texto para ser mostrada por la pantalla, por defecto se accede a la carpeta "Values", e incluida en ella al fichero de cadenas de texto "Strings". Sin embargo, esta carpeta se ha copiado y renombrado a "Values-en", incluyendo una copia del fichero de cadenas de texto traducidas a inglés. De esta forma, tenemos dos carpetas "Values", una con las cadenas de texto predeterminadas en español, y una con las cadenas de texto traducidas al inglés, introducidas en la carpeta "Values-en", en la cual el propio sistema operativo está preparado para acceder si detecta que el móvil está configurado en inglés (el código "Values-en" es el oficial establecido por los desarrolladores de Android).

En las figuras 13 y 14 se puede observar el árbol del proyecto, donde aparecen dentro de los recursos del mismo las dos carpetas previamente comentadas, con los ficheros de cadenas de texto en ambos idiomas. Así, el identificador de cada cadena de texto es común, para poder referenciarlo en el código. No obstante, el valor de la cadena de texto sí cambia, dependiendo del idioma configurado.

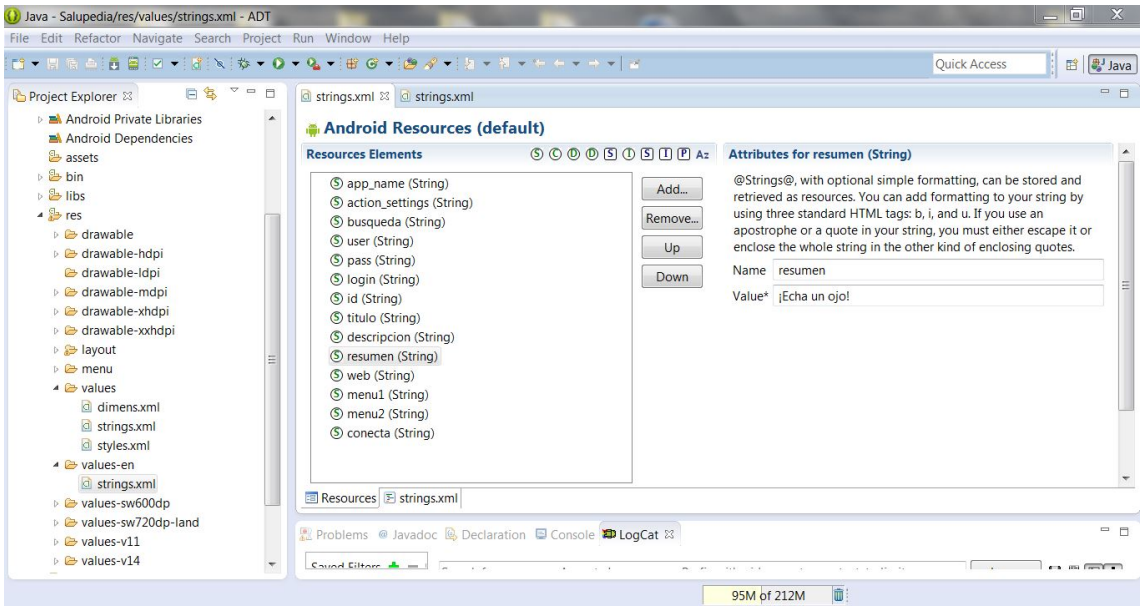

Figura 13 – Fichero de cadenas de texto en español.

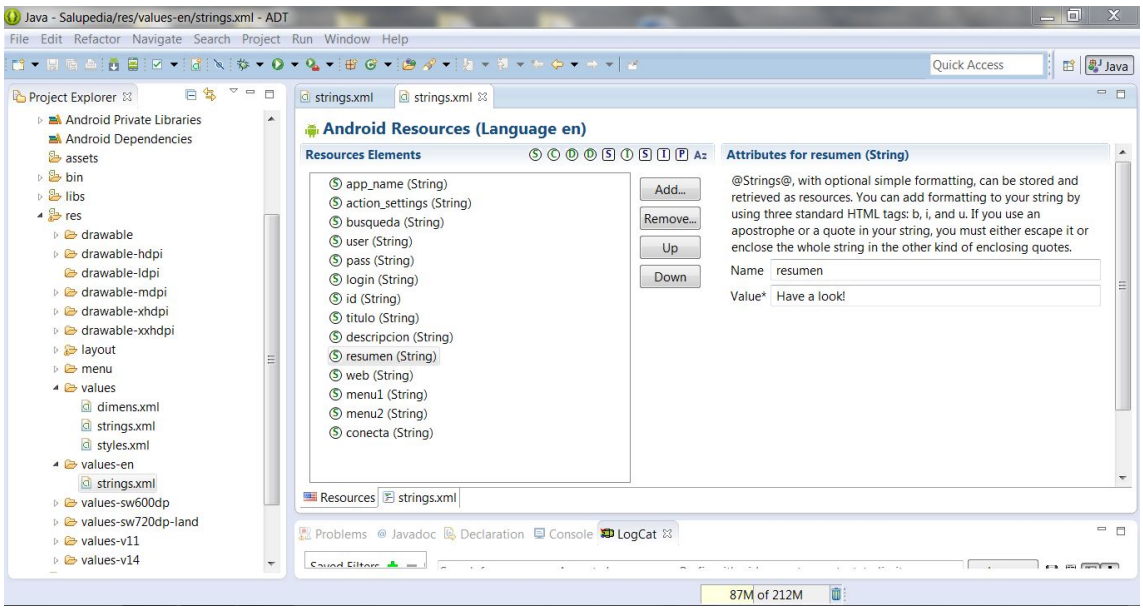

Figura 14 – Fichero de cadenas de texto en inglés.

4.3 – Vista principal

La vista principal se ha diseñado para ser amena, sencilla e intuitiva. Dado el amplio abanico de edades a las que esta aplicación está dirigida, no podía optarse por un diseño cargado, complejo o poco intuitivo ya que cierto sector de los usuarios no tiene unas grandes habilidades tecnológicas.

En la figura 14 se observa la vista principal de la aplicación cuando es ejecutada desde inicio:

Figura 14 – Vista principal de Salupedia App.

Esta vista corresponde con la *activity* principal, que está formada por dos archivos principales, la clase de Java "Main" y la vista XML "mainview". En la clase de Java se detalla el código de la aplicación, es decir, cómo van a reaccionar los distintos campos gráficos cuando sean pulsados. En la vista XML, simplemente se hace el diseño de los campos gráficos. En este caso se cuenta con cuatro Textviews (el título principal y el texto de cada botón), tres botones con imágenes y una lista desplegable Spinner.

Así pues, esta activity se puede considerar como "pasarela" entre las otras activities que llevan el peso de la aplicación y se encargan de mostrar los contendidos. También otorga la posibilidad de seleccionar la categoría acerca de la cual se quiere obtener información mediante la lista desplegable (spinner).

En relación al código, en la figura 15 se observa la documentación obtenida mediante **javadoc** para la clase Main, donde se aprecian los distintos métodos que recoge esta clase y que darán paso a las siguientes clases.

Method Summary

Methods

Modifier and Type	Method and Description
boolean	`onCreateOptionsMenu(Menu menu)` Fija el Menu personalizado que hemos creado
boolean	`onOptionsItemSelected(MenuItem item)` Especifica que método ejecutar en función del item seleccionado
void	`runAcercaDe(View v)` Llama a la clase AcercaDe
void	`runBase(View v)` Llama a clase Basedatos
void	`runExit(View view)`
void	`runUltimas(View v)` Llama a la clase UltimasNoticias
void	`runWeb(View v)` Llama a la clase Webview

Figura 15 – Métodos de la clase Main en javadoc.

El primero de los métodos que aparece, *onCreateOptionsMenu*, simplemente se encargará de que el menú que aparezca al pulsar el botón de menú en nuestro terminal Android sea el que hemos diseñado y que se explicará posteriormente en su apartado.

El segundo de los métodos, *onOptionsItemSelected*, especificará las medidas a tomar cuando uno de los ítems de la lista del menú es seleccionado.

Los métodos *runAcercaDe*, *runBase*, *runUltimas* y *runWeb* se encargan de lanzar un intent para cambiar de activity y empezar las clases correspondientes a cada uno de ellos. Cada uno de

estos métodos está asociado a un botón, de modo que cuando se pulse uno de estos botones se ejecuta el código asociado al método correspondiente.

El método *runAcercaDe* está asociado con la opción del menú "Acerca de", que provee una descripción de Salupedia para que el usuario móvil que desconocía el portal web tenga una introducción explicativa acerca del mismo (el menú es explicado en los siguientes apartados en profundidad).

El método *runBase* está asociado con el botón "Conecta" (con el icono de la lupa) y se encargará de ejecutar la clase que se conectará con la base de datos.

El método *runUltimas* hace referencia al botón "¡Echa un ojo!" (icono del ojo) y ejecutará la clase que mostrará el lector XML.

El método *runWeb* está relacionado con el botón "Versión web" (icono de la bandera) y ejecutará la versión web de Salupedia.

Por último, el método *runExit* está asociado a la opción "Salir" del menú que simplemente ejecutará el método "finsih()" (predeterminado en Andorid) y saldrá de la aplicación.

Por ejemplo, en la figura 16 se observa el código típico de estos métodos, que realizan una pasarela entre la propia clase y la siguiente clase que se desea ejecutar. Posteriormente, se ejecuta el intent descrito mediante el comando startActivity. El código empezará una nueva activity, con nuevos campos gráficos y nuevo código a ejecutar.

```
/**
 * Llama a clase Basedatos
 * @param v
 */
public void runBase(View v){   // muestra las noticias
    Intent i = new Intent(this, Basedatos.class);
    startActivity(i);
}
/**
 * Llama a la clase AcercaDe
 * @param v
 */
public void runAcercaDe(View v){   // muestra las noticias
    Intent i = new Intent(this, AcercaDe.class);
    startActivity(i);
}

public void runExit(View view){ //Cierra la app
    finish();
}
```

Figura 16 – Código de los métodos asociados a los botones.

En la figura 17 se observa el código XML asociado, en este caso al botón que redirige al lector XML, donde se detalla el campo “android:onClick=”runBase”. Esto quiere decir que cuando se haga click en dicho botón (el procedimiento es idéntico para el resto de botones) se buscará el método indicado (en este caso runBase) y se ejecutará el código asociado.

```
<Button
    android:id="@+id/resumen"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@+id/textView1"
    android:layout_centerVertical="true"
    android:layout_marginRight="18dp"
    android:background="@android:drawable/ic_menu_view"
    android:onClick="runUltimas" />
```

Figura 17 – Código XML de un botón.

Este procedimiento de relación entre los campos gráficos y el código es el que se ha seguido a lo largo de la realización del presente proyecto.

Figura 18 – Diagrama transición.

Se ha de destacar también la función de la lista desplegable Spinner que aparece en la vista principal. En ella se muestra al usuario cuando se hace click sobre la lista, como se detalla en la figura 19, las seis categorías principales en las que se dividen los artículos de Salupedia. Esta selección determinará posteriormente la temática de la información que la aplicación mostrará al usuario.

Figura 19 – Lista desplegada.

Esta lista desplegable asigna los parámetros que se enviarán mediante un bundle introducido en un intent (como se detalla en el apartado 4.1) a la activity que se encarga de mostrar la información vía web. Como se aprecia en la figura 20, dependiendo de que ítem esta seleccionado, la cadena de texto "url" obtendrá un valor distinto, relacionado con la temática correspondiente que la lista desplegable muestra. Si ningún valor esta seleccionado, al pulsar el botón por defecto redirigirá a la página principal de Salupedia para evitar posibles errores en la aplicación.

```
public void onItemSelected(AdapterView<?> padre, android.view.View v, int posicion, long id){
    if(datos[posicion]=="ASUNTOS SOCIALES"){
        url="http://www.salupedia.org/salud/categoria/05/asuntos-sociales-y-familia";
    }else if(datos[posicion]=="ENFERMEDADES"){
    url="http://www.salupedia.org/salud/categoria/01/enfermedades-o-problemas-de-salud";
    }else if(datos[posicion]=="PRUEBAS Y DIAGNÓSTICOS"){
        url="http://www.salupedia.org/salud/categoria/03/pruebas-procedimientos-diagnosticos";
    }else if(datos[posicion]=="SALUD Y TECNOLOGÍA"){
        url="http://www.salupedia.org/salud/categoria/06/salud-y-tecnologia";
    }else if(datos[posicion]=="TRATAMIENTOS Y CONSEJOS"){
        url="http://www.salupedia.org/salud/categoria/04/tratamientos-y-consejos";
    }else if(datos[posicion]=="VIDA SANA"){
        url="http://www.salupedia.org/salud/categoria/02/vida-sana-y-prevencion-de-la-enfermedad-";
    }else{
    url="http://www.salupedia.org";
    }

}

@Override
public void onNothingSelected(AdapterView<?> parent) {
    url="http://www.salupedia.org";
```

Figura 20 – Código relacionado con la selección de ítem.

Nótese que "datos" es un vector de cadenas de texto que tiene almacenadas todas las cadenas de texto que aparecen en la lista desplegable. Así pues, los condicionales pueden devolver verdadero si el texto coincide o falso en caso contrario.

El método *onNothingSelected* se encarga de cubrir a posibilidad que ningún elemento se haya seleccionado, evitando un cierre repentino de la aplicación si se requiriese el valor de "url" y no tuviese ninguno seleccionado. No obstante, al inicializar la cadena de texto "url" ha sido igualada a la página principal para cubrir esta posibilidad. De este modo, si se pulsa el botón de "Versión Web" sin haber seleccionado una categoría accederemos a la página principal, y si alguna está seleccionada, a la página correspondiente.

En los siguientes apartados incluidos en el capítulo 4 se detallará el funcionamiento de las distintas activities a las cuales provee acceso la actividad principal Main.

4.4 – Lector XML

El lector XML, al que se accede mediante el botón de la vista principal "¡Echa un ojo!", fue el primer método que se ha programado en le presente proyecto para obtener los contenidos sanitarios que Salupedia ofrece. Esta solución se pensó y posteriormente se desarrolló ya que se precisaba de un lector de noticias rápido e intuitivo y la información requerida se podía obtener tratando la información en formato XML de un lector RSS que Salupedia tiene en la web. De este modo se consigue leer las noticias que Salupedia va publicando adaptadas a la pantalla del *Smartphone* de una forma visual y cómoda para el lector.

Esta funcionalidad se hizo posible gracias la página web con los contenidos necesarios presentados en el lenguaje de etiquetas XML que Salupedia dispone. En la figura 21 se puede observar la visualización de dicha página, y en la figura 22 su código fuente, que es clave para entender cómo se ha programado el lector XML.

Figura 21 – Página web en XML.

Como se puede observar en la figura 21, se muestra el título de la noticia, la fecha de su publicación y la descripción del artículo. Estos campos son casi todos los necesarios. Sin embargo, no todos los campos están mostrados, la página esconde campos que obtendremos y mostraremos como el de la url asociada al artículo.

```
href="http://pubsubhubbub.appspot.com/" /><item>
                        <title>Consejos para una alimentación saludable.</title>
                        <author>salupedia@salupedia.org (Salupedia)</author>
                        <link>http://feedproxy.google.com/~r/Salupedia/~3/WVl-GrHTC58/consejos-para-
una-alimentacion-saludable-</link>
                        <description><![CDATA[<img class='imgEnlaceSnapshot'
src='http://www.salupedia.org/images/webThumbs/snap_9091ecdb877931fa4528eca5db050642.jpg' width='118'
alt='Captura de
http://www.semfyc.es/pfw_files/cma/Informacion/modulo/documentos/guia_alimentacion.pdf' /><p>Extensa
guía publicada por la<span style="color: #f88910;"><span style="color: #f88910;"> Sociedad Española de
Nutrición Comunitaria (SENC) y <span style="color: #000000;">la</span> Sociedad Española de Medicina
de Familia y Comunitaria (semFYC) <span style="color: #000000;">donde se aborda el tema de la
alimentacion y aspectos tales como el diseño de un menú equilibrado y sugerente, orientaciones para
realizar la compra, la preparación y guiso de los alimentos, su conservación y la seguridad en la
cocina, así como las necesidades nutricionales en grupos especiales (embarazo y lactancia, niños y
ancianos).</span></span></span></p>]]></description>
                        <guid isPermaLink="false">http://www.salupedia.org/salud/enlaces/279/consejos-
para-una-alimentacion-saludable-</guid>
                        <content:encoded><![CDATA[<p>Extensa guía publicada por la<span style="color:
#f88910;"><span style="color: #f88910;"> Sociedad Española de Nutrición Comunitaria (SENC) y <span
style="color: #000000;">la</span> Sociedad Española de Medicina de Familia y Comunitaria (semFYC)
<span style="color: #000000;">donde se aborda el tema de la alimentacion y aspectos tales como el
diseño de un menú equilibrado y sugerente, orientaciones para realizar la compra, la preparación y
guiso de los alimentos, su conservación y la seguridad en la cocina, así como las necesidades
nutricionales en grupos especiales (embarazo y lactancia, niños y ancianos).</span></span></span></p>
<img src="http://feeds.feedburner.com/~r/Salupedia/~4/WVl-GrHTC58" height="1" width="1"/>]]>
</content:encoded>
                        <dc:subject>Consejos para una alimentación saludable.</dc:subject>
                        <dc:date>2014-05-30T12:15:48+02:00</dc:date>
                <feedburner:origLink>http://www.salupedia.org/salud/enlaces/279/consejos-para-una-
alimentacion-saludable-</feedburner:origLink></item>
```

Figura 22 – Código fuente de un artículo en formato XML.

Analizando el código fuente que se muestra en la figura 22, se observa que cada artículo comienza con la etiqueta <item>. Por lo tanto, el código programado se encarga de recorrer este archivo, al cual accede mediante la url de la página web mostrada anteriormente, buscando las etiquetas de inicio y fin de item, es decir, <item> y </item>. Cuando el programa encuentre esta etiqueta de inicio, sabrá que ha encontrado un artículo. Dentro de cada artículo, se guardará cierta información en vectores de cadenas de texto, pertenecientes a un objeto de la clase llamada “Noticia”. En concreto se guardará información de los siguientes campos:

- Título de la noticia. Se encontrará entre las etiquetas de apertura y cierre <title> y </title>.
- Descripción de la noticia. Se encontrará entre las etiquetas de apertura y cierre <description> y </description>.
- Fecha de publicación de la noticia. Se encontrará entre las etiquetas de apertura y cierre <dc:date> y </dc:date>.
- Link de la noticia. Se encontrará entre las etiquetas de apertura y cierre <link> y </link>.

Estos cuatro campos serán almacenados en vectores de cadenas de texto independientes, definidos en la clase "Noticia" con el fin de ser mostrados posteriormente. En cuanto el programa encuentre la etiqueta de fin de item, buscará la siguiente etiqueta de inicio y guardará sus atributos.

El lector XML esta formado por 3 clases Java (Noticia, Ultimasnoticias y Noticiadetalle) y dos vistas de campos gráficos XML (noticias y detalladas).

La clase **Noticia** se ha diseñado con el objetivo de facilitar la recopilación de la información asociada a un artículo en concreto, recogiendo los campos previamente expuestos (título, descripción, fecha de publicación y link). Por lo tanto será una clase con cuatro cadenas de texto, una para cada campo, y dispone de los métodos necesarios para obtener cada uno de dichos campos (comúnmente conocidos como getters).

La clase **Ultimasnoticias** va acompañada de la vista XML **noticias**, que es conveniente explicar previamente para entender como se ha programado el código de la clase. Como se aprecia en a figura 23, la vista incluye una lista de ítems listview, a la cual se le permite deslizar para ver la totalidad de los elementos (scrollview).

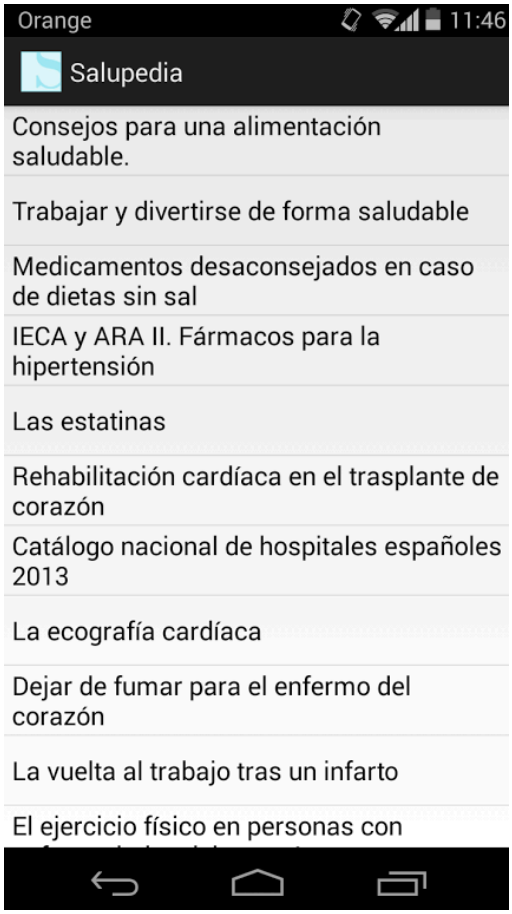

Figura 23 – Vista noticias.

Por lo tanto, en la clase de java **Ultimasnoticias** nos encargaremos de recoger toda la información de los títulos, descripciones, fechas de publicación y links de todas las noticias que aparecen en la web (vienen limitadas a veinte entradas, por lo que no se hace necesario limitarlas en el código).

No obstante, en esta primera vista no se desea mostrar todas las descripciones de todos los artículos, si no que se pretende mostrar únicamente los títulos para que el usuario no obtenga un exceso de información, y cuando encuentre uno de su interés, que haga click sobre él y aparezca una nueva activity con toda la información acerca del artículo seleccionado (título, descripción, fecha de publicación y link).

Por lo tanto, toda la información que se muestra mediante este método se ha almacenado en un vector de noticias, de modo que la primero noticia estará situada en la posición cero del vector de noticias, la segunda noticia en la posición uno, y así sucesivamente. En la figura 24 se muestra la estructura esquematizada.

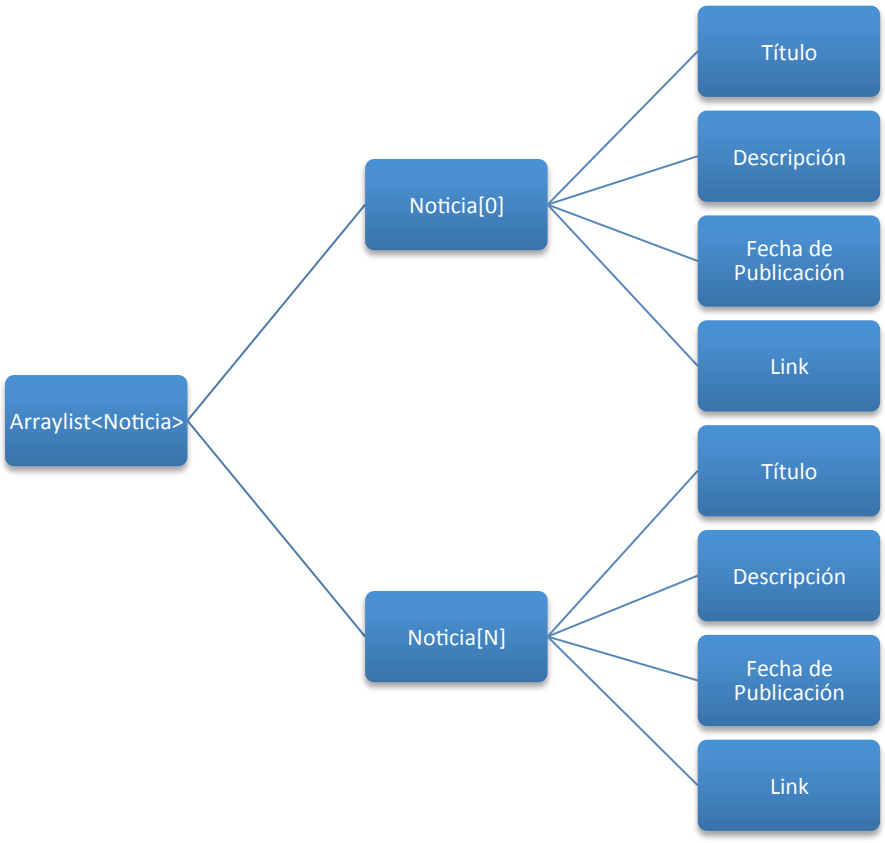

Figura 24 – Esquema distribución información en vectores.

El método de recogida de información ha sido llevado a cabo mediante las facilidades que Android proporciona para el tratamiento de ficheros XML con los objetos de tipo XML Parser y XML Factory.

```
while (evento != XmlPullParser.END_DOCUMENT){

    switch (evento) {

    case XmlPullParser.START_TAG:

        // Controla si es el inicio de una noticia
        if (xml.getName().equals("item")){

            esItem = true;
        }

        // Controla si es una noticia
        if (esItem){

            //Comprobación etiquetas

            if (xml.getName().equals("title")){

                leerTitutlo = true;
            }
```

Figura 25 – Código de flags.

Como se observa en la figura 25, se recorre el documento buscando las etiquetas que hacen referencia a los campos que se desea obtener, y se almacena en una variable tipo *boolean* para indicar que nos hemos encontrado con uno de los campos deseados y que efectivamente existe, y por lo tanto lo podremos almacenar. Este bucle que recorre el documento se realiza siempre que estemos dentro de el mismo, limitando su actividad con la condición del bucle *while*. En cuanto al objeto XML Parser, buscará las etiquetas en caso de que el evento actual corresponda con un inicio de etiqueta, como marca el método *switch case* del código. Pese a que no se observa en la figura superior, hay una variable tipo boolean para cada uno de los campos que queremos recoger, que la condición necesaria para guardarlas en los bloques de código posteriores será que estas variables tengan un valor de *true*, lo cual significa que hay información que recoger por que se ha encontrado la etiqueta correspondiente.

Una vez se tiene constancia de que se dispone de información a almacenar, en el siguiente caso del método *switch case* se almacena el texto correspondiente en los vectores de cadenas de texto.

```
case XmlPullParser.TEXT:
    // Almacenar el contenido
    if (leerTitutlo){
        titulos.add(xml.getText().toString());
    }
    if (leerLink){
        links.add(xml.getText().toString());
    }
    if (leerFecha){
        fechas.add(xml.getText().toString());
    }
    if (leerDescripcion){
        descripciones.add(xml.getText().toString());
```

Figura 26 – Código de almacenamiento en vectores.

En la figura 26, cuando el XML Parser encuentra texto, se fija en las variables tipo boolean (leerTitulo, leerLink, leerFecha y leerDescripcion). Si esta condición se cumple, significa que previamente, en la fase de principio de etiquetas, había información que guardar, por lo que ahora añade al vector de cadenas de texto correspondiente (títulos para los títulos, links para los enlaces, fechas para las fechas de publicación y descripciones para el texto descriptivo) la información obtenida.

Por último, las variables de tipo boolean se han de poner a *false* de nuevo ya que de otro modo seguiría guardándose texto en posiciones del vector de cadenas de texto que no corresponden. Por ello, como se muestra en la figura 27, los valores de las variables se modifican cuando el XML Parser encuentra las etiquetas de fin.

```
case XmlPullParser.END_TAG:

    if (xml.getName().equals("title")){
        leerTitutlo = false;
    }
    if (xml.getName().equals("link")){
        leerLink = false;
    }
    if (xml.getName().equals("dc:date")){
        leerFecha = false;
    }
    if (xml.getName().equals("description")){
        leerDescripcion = false;
    }
```

Figura 27 – Código de flags a false.

De este modo se recorrerá todo el documento, buscando etiquetas, encontrándolas y almacenando en los vectores de cadenas de texto, una posición del mismo para cada artículo, hasta que el documento se acabe (cuando el XML Parser encuentre el evento "End document").

Con el código visto hasta ahora, conseguimos almacenar la información en vectores de cadenas de texto, los cuales se han de cargar en el vector de noticias. Además, se ha implementado un método para obtener los títulos de las noticias que será lo que se cargue en la lista *listview* de la vista del lector. En la figura 28 se puede observar el desarrollo de ambos métodos.

```
public void almacenarNoticias(ArrayList<String> titulos, ArrayList<String> descripciones,
        ArrayList<String> fechas, ArrayList<String> links){

    for (int i = 0; i < titulos.size(); i++){

        noticias.add(new Noticia(titulos.get(i), descripciones.get(i), fechas.get(i), links.get(i)));
    }
}

/**
 * Obtiene y almacena los titulos de las noticias
 */
public void obtenerTitulos(){

    tituloNoticias = new String[noticias.size()];

    for (int i = 0; i < noticias.size(); i++){
        tituloNoticias[i] = noticias.get(i).getTitulo();
    }
}
```

Figura 28 – Código obtención y carga de información.

Se ha elegido limitar el bucle por el tamaño del vector de títulos ya que es un campo que siempre va a estar presente en cualquier artículo, ya que puede darse el caso de que, por ejemplo, una noticia no tenga referencia de su fecha de publicación y por ello no se cargue.

Los títulos se obtienen una vez se ha cargado la información en el vector de noticias, y luego se cargan en la lista mediante un adaptador, siguiendo el procedimiento estándar de Android.

Por último, para finalizar con el lector XML, se ha de permitir ampliar la información acerca del título que hemos seleccionado en la vista de los títulos, para poder acceder a su descripción, fecha y link. Esto se realiza en el método *onItemClick,* que actúa cuando se hace click sobre uno de los títulos, accionando un *intent* con un contendor de información *bundle* que contiene la información necesaria acerca de la noticia seleccionada, que se enviará a la nueva *activity* la cual dispondrá de campos de texto textview para mostrar el título, la fecha, el link y la descripción correspondientes.

En la figura 29 se muestra el código de cambio de activity a la segunda vista XML de este lector, la vista **detalladas**. El código que actuará sobre ella será la clase **Noticiadetalle**, que simplemente se encargará de recibir la información obtenida por la clase anterior y asignarla a cada campo gráfico de visor de texto textview.

```
        /**
         * Cada vez que pinchemos en un item reaccionará mostrando la noticia detallada
         */
        lstvNoticias.setOnItemClickListener(new OnItemClickListener() {

            /**
             * Muestra detalladamente la noticia seleccionada
             */
            public void onItemClick(AdapterView<?> adaptador, View vista, int posicion,
                    long arg3) {

                // Pasa la noticia seleccionada a la vista en detalle
                Intent intent = new Intent(Ultimasnoticias.this, Noticiadetalle.class);
                Bundle bundle = new Bundle();
                bundle.putSerializable("NOTICIA", noticias.get(posicion));
                intent.putExtras(bundle);
                startActivity(intent);
            }
        });

}
```

Figura 29 – Código cambio de vista.

Así pues, la vista de una noticia clicada, en detalle queda reflejada en la figura 30, donde se aprecia el título en la parte superior, la fecha y el link justo debajo y en el grueso de la pantalla la descripción de la noticia.

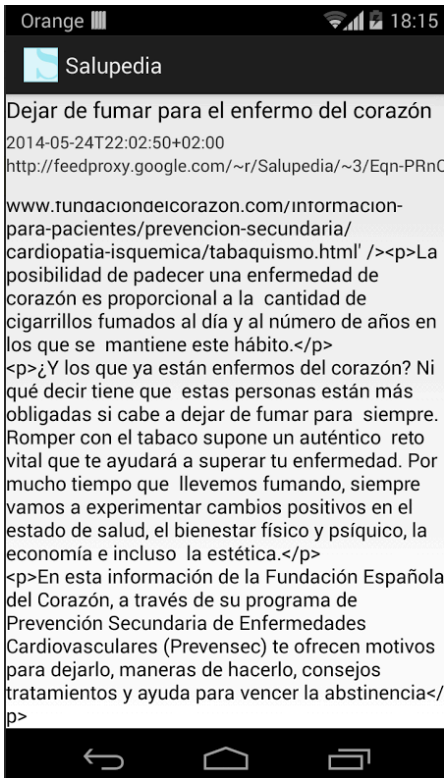

Figura 30 – Vista noticia detallada.

4.5 – Webview

Este apartado de la aplicación se ha pensado para proveer al usuario la posibilidad de acceder a la página web oficial de Salupedia sin salir de la aplicación, en el caso de que desee acceder a la vista que ofrece la web, o las funcionalidades extra que son características de la página completa. Además, se ha desarrollado permitiendo acceder a la web principal o por el contrario, acceder directamente a una de las categorías principales que Salupedia ofrece, para evitar ese paso intermedio, mediante la lista desplegable spinner que se detalló en el apartado 4.3 vista principal.

Recordemos que dependiendo del ítem de la lista seleccionado se generaba una pasarela a la activity que muestra el webview con la url particular de cada temática introducida (nótese que si ningún ítem fue seleccionado se enviará por defecto la dirección de la página principal). Así al pulsar el botón de "Versión web", se llama a la clase **Webview**. La clase Webview se encarga de recibir la url desde la activity principal, y cargarla en su vista gráfica. La vista gráfica asociada a este apartado se llama **weburl** y simplemente consiste en un visor web, con el que se interactúa desde el código de la clase Webview y se le carga la url deseada, mostrando en la pantalla de nuestro teléfono la página web como si de un navegador normal se tratara.

En la figura 31 se observa el método de la actividad principal que llama a la clase Webview y le envía la url deseada.

```
public void runWeb(View v){
    Intent i = new Intent(this, Webview.class);
    Bundle b=new Bundle();
    b.putString("Link", url);
    i.putExtras(b);
    startActivity(i);
}
```

Figura 31 – Método pasarela a clase Webview.

Por lo tanto, al recibir la cadena de texto identificada como "Link", la clase Webview únicamente la recibirá y la cargará al campo gráfico webView (visor web) dicha url. En la figura 32 se recoge el código íntegro de la clase Webview.

```java
public class Webview extends Activity {
    private URL url;
    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.weburl);
        Bundle b = this.getIntent().getExtras();
        String link = b.getString("Link");
        WebView web;
        web = (WebView) findViewById(R.id.webView);
        web.loadUrl(link);
    }

}
```

Figura 32 – Código clase Webview.

Finalmente, los resultados de este apartado de la aplicación de recogen en la figura 33, donde se aprecia una ventana abierta cuando la categoría seleccionada en este caso ha sido "Tratamientos y consejos".

Figura 33 – Vista de la página web desde la aplicación Salupedia.

4.6 – Menú aplicación

Con el desarrollo del menú se ha pretendido dotar a la aplicación de unas características básicas que deben estar presentes en todo proyecto de software pero que por cuestiones estéticas se ha decidido ocultar bajo el botón de menú el cual todos los terminales Android disponen. Estas funcionalidades son:

- **Acerca de**.
- **Salir**.

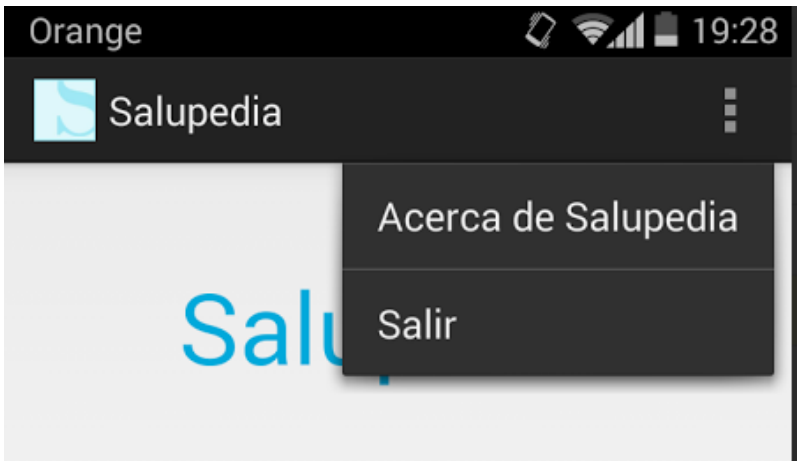

Figura 34 – Menú aplicación.

La pestaña de "Salir" simplemente ejecutará un método llamado *finish()* propio de Android que se encargará de cerrar la aplicación, dado que es más rápido y seguro que tratar de cerrar la aplicación pulsando el botón de ir hacia atrás.

La pestaña "Acerca de" se encarga de mostrar una activity pasiva, con la que no se interactuar, simplemente está formada por dos visores de texto (uno para el nombre de la aplicación y otro para la descripción de Salupedia) y un visor de imagen que muestra el icono de la aplicación. Con esto se dota al usuario de una breve descripción de en qué consiste Salupedia, orientado a clientes que se han descargado la aplicación pero que no conocían la plataforma web previamente.

En la figura 35 se observa una captura de pantalla de la vista previamente expuesta de "Acerca de".

Figura 35 – Vista "Acerca de".

4.7 – Acceso base de datos

Como funcionalidad final, se pensó dotar a la aplicación de la posibilidad de interactuar directamente con la base de datos que Salupedia dispone, dado que este recurso permite, mediante las consultas SQL, realizar cualquier funcionalidad que la propia interfaz web ya proporciona.

Por lo tanto, se procedió a realizar una interfaz que gestiona una conexión entre la aplicación y el servidor que contiene la base de datos SQL. En conversaciones con el equipo de ingenieros del ITACA, se entendió que la conexión se debía gestionar mediante un fichero lenguaje PHP que se ha alojado en el servidor de datos, al cual se accede desde el código de la aplicación.

El fichero PHP, aparte de gestionar la conexión se encargará de realizar las consultas SQL deseadas. Esto amplía en gran cantidad el número de funcionalidades que la aplicación puede alcanzar, ya que las consultas con la base de datos pueden desde simplemente obtener información hasta añadir nuevos artículos a la base de datos hasta puntuar noticias o crear usuarios. Por lo tanto, esta interfaz deja la puerta abierta a **múltiples posibilidades** que pueden ser explotadas. Además, también se acordó que el objeto devuelto por el fichero PHP tuviese el formato JSON, dado que es el estándar actual en internet y dispone de numerosos métodos de programación para ser tratado.

Para poder programar el fichero PHP, se ha de conocer la estructura de las tablas de la base de datos y los credenciales necesarios para acceder al servidor. Los credenciales fueron facilitados por el equipo del ITACA sin ningún problema, y para obtener la estructura de las tablas de la base de datos se usó la herramienta explicada en el apartado 3 HeidiSQL, con la cual introduciendo los credenciales necesarios de servidor, usuario y contraseña, muestra la totalidad de las tablas y sus datos.

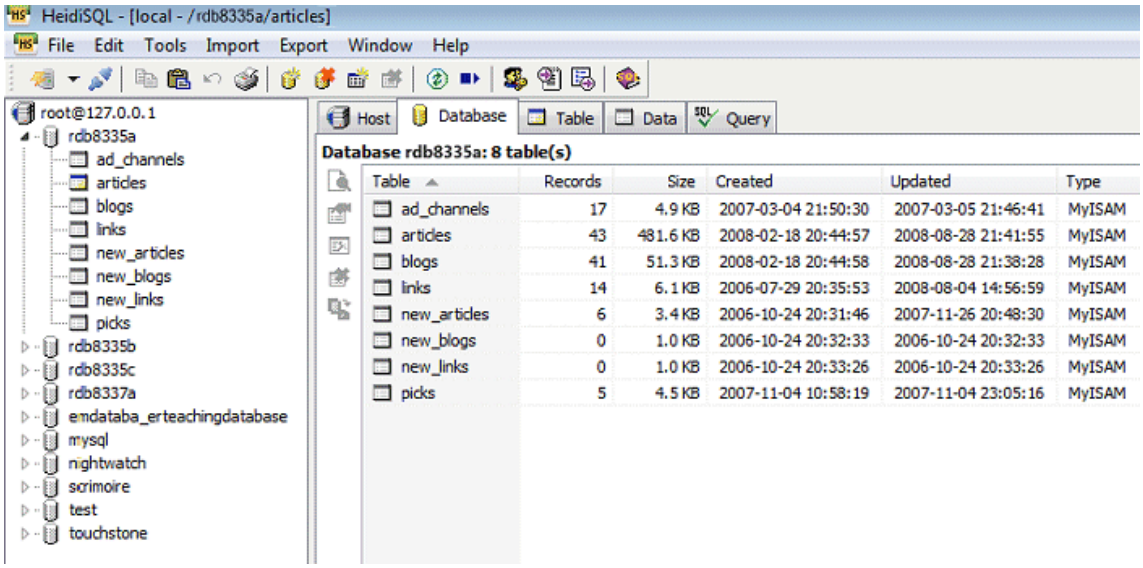

Figura 36 – Ejemplo interfaz HeidiSQL.

Así pues, con el conocimiento de la estructura de la base de datos, los credenciales necesarios y el formato del objeto que se devolverá, se programó un fichero PHP simple y se alojó en el servidor de datos de Salupedia. De este modo, se puede acceder al archivo desde internet simplemente escribiendo la url asociada a dicho archivo, y se observará que efectivamente devuelve un objeto en formato JSON.

En la figura 37 se observa el código del fichero PHP, mientras que en la figura 39 aparece el resultado de escribir la url asociada al fichero PHP, y efectivamente sigue la estructura de un objeto JSON.

```php
<?php

$con = mysql_connect("w2003php","movilSalupedia","******");
if (!$con)
{
die('Could not connect: ' . mysql_error());
}
mysql_select_db("salupedia_movil_db", $con);

$sql="SELECT links.ID_LINKS, links.TITLE, links.DESCRIPTION, links.DATE_REG, links.LAST_UPDATE, links.LINK_URL,
links_stats.NUM_VISITS as total_visitas, NUM_COMMENTS AS total_comentarios,
VOTES_USER AS prom_Usuarios, NUM_VOTES_USER,
VOTES_PROF AS prom_Profesional, NUM_VOTES_PROF
FROM  links
LEFT OUTER JOIN links_stats ON (links.ID_LINKS = links_stats.ID_LINKS)
LEFT OUTER JOIN categories_links ON (links.ID_LINKS = categories_links.ID_LINKS)
WHERE categories_links.ID_LINKS=65
GROUP BY links.ID_LINKS
ORDER BY links.LAST_UPDATE DESC, links.ID_LINKS DESC
LIMIT 10
";


$result = mysql_query($sql);

while($row = mysql_fetch_assoc($result))
{
$output[]=$row;
}

print(json_encode($output));

mysql_close($con);


?>
```

Figura 37 – Fichero PHP.

El fichero, como se comentaba, gestiona la conexión, y en este caso particular realiza la consulta de diez artículos con datos varios como el título, fecha link o número de visitas filtrados por categoría, en este caso la del identificador referencia 65, problemas infantiles. El nombre del fichero es “mobile_con2.php”, por lo que la ruta que se debe seguir para acceder a dicho fichero es “http://w2003php.itaca.upv.es/salupedia/mobileSalupedia/mobile_con2.php”.

Si decidimos escribir esta ruta en cualquier navegador, obtendremos un objeto con formato JSON. Es una buena medida para comprobar que nuestro fichero PHP está realizando lo que se esperaba de él. La estructura de un objeto JSON es simple, sigue el patrón de “atributo”:”valor”.

En la figura 38 se ha detallado un ejemplo de estructura de objeto JSON aplicado a un campo simple, una base de datos de personas.

```
[   {
        “Nombre” : ”Víctor” ,
        “Apellido” : “Barberá” ,
        “Categoría” : “Alumno”
    }
        ,
    {
        “Nombre” : ”Vicente” ,
        “Apellido” : “Traver” ,
        “Categoría” : “Profesor”
    }
]
```

Figura 38 – Formato JSON ejemplo.

Por lo tanto, los resultados obtenidos al acceder a este fichero PHP se muestran en la figura 39, observándose que el formato coincide con el esperado. Nótese que se ha usado un servicio web [9], que aportando la url del objeto JSON, lo muestra espaciado y le dota de un formato para que su análisis sea más ameno.

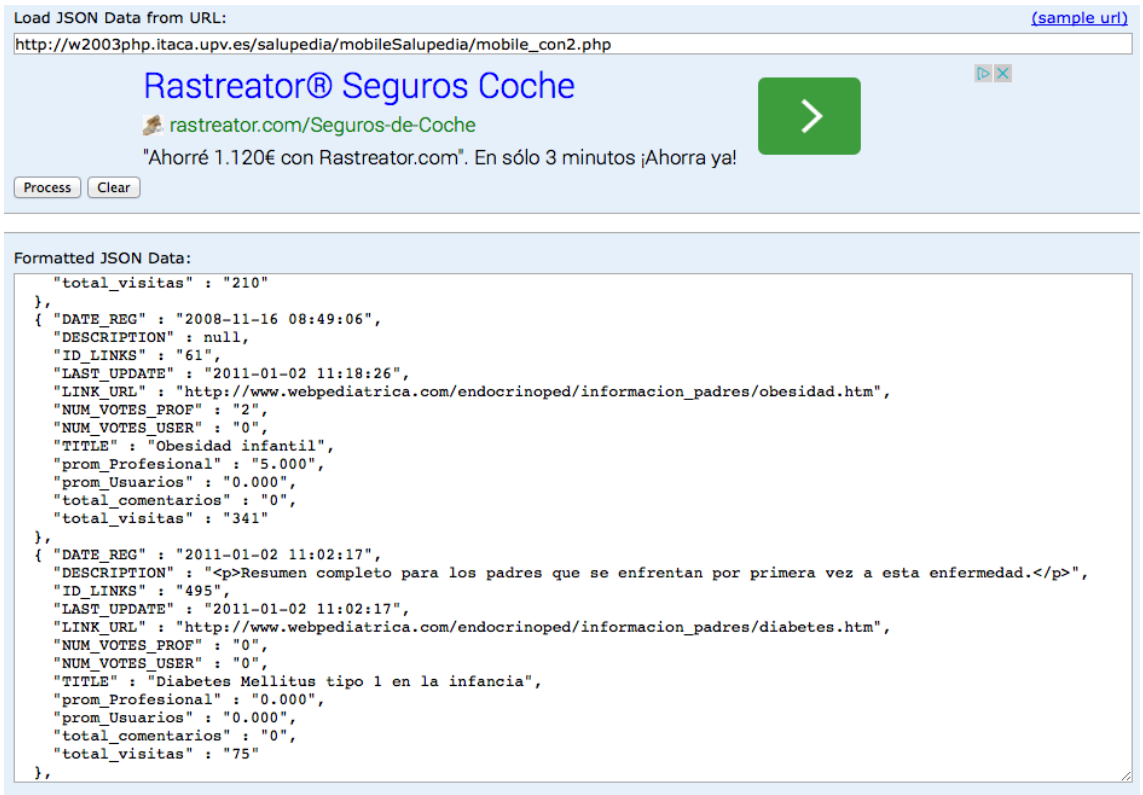

Figura 39 – Resultado JSON obtenido.

Así pues, basándonos en el método utilizado en el lector XML donde teníamos una clase llamada Noticia que nos facilitaba trabajar con la información, en este apartado se ha programado una clase llamada **Apiconnection**, que gestionará la conexión con el fichero PHP y que dispone de un método que devuelve un objeto JSON (ya que el acceso al fichero devuelve un JSON). Este objeto JSON será usado por la segunda clase de este apartado, la clase **Basedatos**, que se encargará de obtener el objeto JSON, analizar las etiquetas y obtener y mostrar el texto deseado por pantalla.

En la figura 40, se observa el proceso de la petición http a la dirección del archivo PHP en el servidor. Como se detalla, el método devolverá un JSON Array, con varios objetos (que realmente será artículos) en formato JSON.

```
public class Apiconnection {

    public JSONArray getJData(){

        String url="http://w2003php.itaca.upv.es/salupedia/mobileSalupedia/mobile_con2.php";
        HttpEntity httpEntity = null;

        try{

            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpGet httpGet = new HttpGet(url);

            HttpResponse httpResponse = httpClient.execute(httpGet);
            httpEntity = httpResponse.getEntity();

        } catch(ClientProtocolException e){

            e.printStackTrace();

        } catch(IOException e){

            e.printStackTrace();
        }
```

Figura 40 – Petición http al fichero PHP.

En la figura 41, simplemente se observa como obtiene la respuesta, la introduce en un objeto JSON Array y la devuelve, para que cuando se llame al método lo que se obtenga sea, efectivamente, el objeto JSON Array deseado.

```
    JSONArray jsonArray = null;

    if(httpEntity != null){
        try{
            String entityResponse = EntityUtils.toString(httpEntity);

            Log.e("Entity Response : ", entityResponse);

            jsonArray = new JSONArray(entityResponse);
        }catch (JSONException e){
            e.printStackTrace();
        }catch (IOException e){
            e.printStackTrace();
        }


    }

    return jsonArray;

}
```

Figura 41 – Devolución de JSON Array.

La segunda clase, **Basedatos**, se encarga de obtener el objeto JSON Array y mostrarlo por pantalla. La particularidad de este código ha sido el uso de las tareas asíncronas **AsyncTask**. Cuando una aplicación Android requiere un flujo computacional extenso, puede perder la capacidad de "refrescar" los campos gráficos y mantener toda su actividad en tratar de obtener la información de los ficheros. Con el uso de las tareas asíncronas, se consigue dividir los procesos en hilos de ejecución totalmente distintos, de modo que el hilo que se encarga de procesar y obtener los datos corre en segundo plano (*background*) y no imposibilita ni afecta al hilo principal de la aplicación, por lo que si un error ocurre la aplicación no se colgará. Los métodos para ejecutar una tarea asíncrona son los siguientes:

Figura 42 – Métodos típicos de las tareas asíncronas.

De los métodos típicos (no es necesario implementarlos todos), se han implementado:

- **doInBackground**: Recoge las acciones que se realizan en segundo plano, o sea las que requieren de mayor carga computacional.
- **onPostExecute**: Recoge las acciones que se realizan posteriormente obteniendo los resultados de la ejecución previa.

En las posteriores figuras se detalla el proceso de programación de la clase Basedatos. En la figura 43, se realiza la inicialización y se ejecuta la tarea asíncrona GetJDataTask.

```
public class Basedatos extends Activity {
    private TextView response;
    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.jview);

        this.response = (TextView) this.findViewById(R.id.response);

        new GetJDataTask().execute(new Apiconnection());
    }
```

Figura 43 – Inicio clase Basedatos.

En la figura 44, se muestra la tarea asíncrona GetJDataTask. Como se observa, las tareas asíncronas reciben parámetros, lanzan procesos y devuelven resultados, siguiendo la estructura AsyncTask<parámetros, progresos, resultados>. Esta tarea recibe como parámetros un objeto de la clase Apiconnection, creada por nosotros previamente, no usa los progresos, y devuelve como resultado un objeto JSON Array. El método doInBackgorung devuelve un objeto JSON Array, ya que ejecuta el método getJData a un objeto de la clase Apiconnection (previamente explicados) y lo envía al método onPostExecute. Este JSON Array es manipulado por onPostExecute mediante un método explicado en la figura 45.

```
    private class GetJDataTask extends AsyncTask<Apiconnection, Long, JSONArray>{

        @Override
        protected JSONArray doInBackground(Apiconnection... params) {
            // Se ejecuta en un hilo paralelo
            return params[0].getJData();
        }
        @Override
        protected void onPostExecute(JSONArray jsonArray){
            setTextToTextView(jsonArray);
        }
    }

}
```

Figura 44 – Código tarea asíncrona.

```
public void setTextToTextView(JSONArray jsonArray){

    String s="";
    for(int i=0;i<jsonArray.length();i++){

        JSONObject json = null;
        try{
            json=jsonArray.getJSONObject(i);
            s=s+
                    json.getString("TITLE")+"\n"+
                    "Descripción : "+json.getString("DESCRIPTION")+"\n"+
                    "Link : "+json.getString("LINK_URL")+"\n"+
                    "Fecha : "+ json.getString("DATE_REG")+"\n\n";
        }catch (JSONException e){
            e.printStackTrace();
            response.setText("JSON Error");
        }
    }
    response.setText(s);
}
```

Figura 45 – Adaptador JSON a cadena de texto.

Finalmente, este método se encarga de interpretar el objeto JSON Array, recorriéndolo con un bucle obteniendo el texto correspondiente a cada etiqueta e introduciéndolo en una cadena de texto. Después de esto, se asigna al visor de texto (Textview) "response" el contenido de la cadena de texto, para que se pueda visualizar el contenido obtenido. Nótese que la vista XML asociada a este apartado es un simple visor de texto, introducido en una vista deslizante para que se le cargue toda la información y s epoda interactuar deslizando para leer todos los contenidos.

A esta interfaz de acceso a la base de datos se accede mediante el botón de la página principal "Conecta", con el icono en forma de lupa. En la figura 46 se observan los resultados finales de el acceso a estos contenidos, en este caso particular la consulta SQL que se ha realizado muestra artículos relacionados con los problemas infantiles.

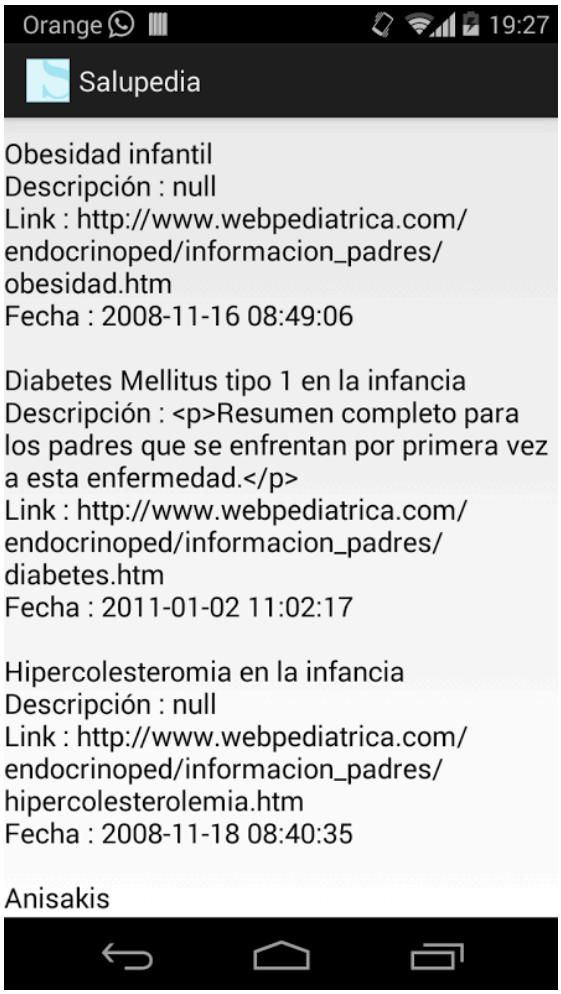

Figura 46 – Vista acceso base de datos.

Capítulo 5 – Conclusiones y propuestas de trabajo futuro

5 – Conclusiones y propuestas de trabajo futuro

5.1 – Discusión personal

Después de la realización de este proyecto puedo afirmar que ha supuesto un reto para mi, tanto en habilidades técnicas como personales. El hecho de afrontar un proyecto con una carga de trabajo considerable, además orientado a resultados ya que se está programando una aplicación que se espera que realice ciertas funciones, en un periodo determinado de tiempo implica un proceso de organización e implicación muy distinto a las simples memorias de prácticas a las cuales te acostumbras durante el periodo de la carrera universitaria.

El hecho de haber cursado este año universitario dos asignaturas relacionadas con Android me impulsó a escoger este proyecto, no obstante he requerido invertir gran parte del proyecto en aprender contenidos acerca de la programación Android que extendían, unas veces sólo pinceladas, pero otras muchas en gran cantidad lo que se había visto en clase.

No obstante, no me arrepiento en absoluto de haber escogido este proyecto, ya que la programación es una tarea que me apasiona, que te proporciona muchos momentos de incertidumbre que son claramente superados por la satisfacción de conseguir que lo que estamos desarrollando realice las tareas para las que ha sido diseñada. Además, el hecho de haber invertido tanto tiempo en documentación y desarrollo de la aplicación ha mejorado con creces mis habilidades de desarrollo de software.

El desarrollo del proyecto en general me ha hecho ver la responsabilidad de gestionar y desarrollar una tarea asignada y asumir sus resultados, aportando madurez a la hora de atender a opiniones y propuestas de cambios.

Estoy satisfecho con el código desarrollado, ya que es fiable y da lugar a mucha expansión, y me gustaría que en un futuro se partiera del mismo para sacar la aplicación al mercado y pudiese ser de utilidad para la gente, que es principalmente el objetivo que persigue el ITACA con la plataforma Salupedia.

5.2 – Conclusiones

Objetivo principal.

El objetivo principal se ha conseguido ya que se ha programado una aplicación que efectivamente muestra la información que Salupedia ofrece en un formato ameno para la interfaz Android.

Facilidad de uso:

La aplicación es sencilla, evitando diseños complejos que "abrumen" al usuario, con el fin de perseguir el objetivo de facilidad. No obstante, este atributo esta sujeto a la percepción de cada usuario, por lo que resulta complejo evaluar si realmente es una aplicación de usos sencillo o no.

Rapidez:

La aplicación ha logrado el objetivo de rapidez, puesto que no se toma excesivo tiempo en procesar y mostrar los resultados, por lo tanto en ningún momento pierde la interactividad con el usuario.

Robustez:

Efectivamente, la robustez ha sido un objetivo perseguido y alcanzado, ya que al usar tareas asíncronas en la sección que más problemas puede acarrear, se evita en gran parte posibles problemas de funcionamiento interno.

Utilidad:

En cuanto al objetivo de utilidad, la aplicación ofrece servicios de consumo de contenidos, ofreciendo una utilidad sí, pero quizá limitada. Si se busca además del consumo, la generación e introducción de los mismos en la plataforma se requeriría de simples pero necesarias modificaciones en las consultas SQL y añadir unos campos de introducción de texto.

En cuanto a los **objetivos secundarios**, el código de la aplicación está correctamente espaciado, comentado y además incluido en javadoc. Además, se ha proporcionado al equipo de ingenieros del ITACA para su uso y futura ampliación, por lo que se ha cubierto este objetivo propuesto.

En los **objetivos "no técnicos"**, se ha aprendido totalmente el proceso de gestión y desarrollo del proyecto en todas sus fases, ya que este es un proyecto individual por lo que el desarrollo del mismo recae sobre el alumno, siempre respaldado por el tutor que cuida de que el alumno consiga los objetivos previstos.

5.3 – Propuestas de trabajo futuro

Al realizarse el apartado de conexión con la base de datos se abrió un abanico de posibilidades enorme para el potencial de la aplicación. Como propuestas de trabajo futuro sugeriría aprovechar esta interfaz que se ha desarrollado para permitir, no solo acceder a los contenidos de la base de datos directamente, sino también **realizar modificaciones en los mismos**.

Por ejemplo, permitir la **incorporación de cuentas de usuario con distintos privilegios**, los pacientes y los médicos, pudiendo estos últimos crear un artículo y subirlo directamente desde la aplicación a la base de datos.

Por otra parte, sería interesante permitir valorar las noticias desde la aplicación y mostrar un **filtrado por los artículos más votados**. No obstante, la posibilidad de ejecutar cualquier consulta SQL permite la realización de estas mejoras propuestas, por lo que se hace sencillo partir del código que actualmente existe.

Por último, las tendencias actuales siempre exigen la **incorporación de las redes sociales** a nuestras aplicaciones. Pese a que la penetración de las redes sociales es volátil con el tiempo, encontraría interesante incorporar una interfaz al menos con Facebook, Twitter y Google+, siendo a mi parecer Twitter la mejor compañera de Salupedia, ya que permitiría con un simple "Tweet" que un artículo de Salupedia fuese leído por miles de personas.

5.4 – Agradecimientos

Principalmente a mis padres por haberme brindado la oportunidad de estudiar una carrera en una de las mejores universidades de España y por haberme apoyado tanto en los días de éxito como en los días de agobio.

A mis compañeros que siempre han estado dispuestos a hacerme ver que nada es suficiente difícil como para que no lo pueda conseguir, en especial Guillermo.

A mi tutor Vicente por su apoyo tanto en el presente proyecto como en las decisiones sobre mi futuro, y por estar dispuesto a revisar el trabajo incluso en fin de semana.

Al equipo de ingenieros de ITACA por mostrarme el camino de las soluciones cuando no lo encontraba y hacerme ver que siempre se puede mejorar el trabajo ya realizado.

Bibliografía

Bibliografía

[1] - https://www.pfizer.es/docs/pdf/noticias/Resultados_encuesta_Pfizer.pdf

[2] - www.20minutos.es/noticia/2153582/0/wikipedia/errores/enfermedades

[3] - Apuntes asignatura Aplicaciones Telemáticas (4º curso)

[4] - Apuntes asignatura Sistemas Complejos Bioinspirados (4º curso)

[5] - https://developer.android.com/develop/index.html

[6] - http://www.sgoliver.net/blog/?p=2610 (Servicios REST)

[7] - http://www.sgoliver.net/blog/?p=2665 (JSON)

[8] - http://www.sgoliver.net/blog/?p=3099 (Asynktask)

[9] - http://www.jsonformat.com

[10] - http://docs.oracle.com/javase/7/docs/api/java.html

[11] - http://json.org

[12] - http://javarevisited.blogspot.com.es/2011/05/example-of-arraylist-in-java-tutorial.html

[13] - http://stackoverflow.com

[14] – Apuntes programación (1º curso)