ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Evaluación del impacto de actividades en eventos y de la afluencia de público mediante dispositivos móviles

AUTOR: Rubén García Díaz
DIRECTOR: Juan Carlos Sánchez Aarnoutse
FECHA: Febrero 2014

Autor	Rubén García Díaz
E-mail del Autor	ru9669@gmail.com
Director	Juan Carlos Sánchez Aarnoutse
E-mail del Director	juanc.sanchez@upct.es
Codirector(es)	
Título del PFC	Evaluación del impacto de actividades en eventos y de la afluencia de público mediante dispositivos móviles
Descriptores	Android, MySQL

Resumen

Desarrollo de una aplicación para dispositivos Android que permita, por un lado, medir la afluencia de asistentes a una feria y, por otro, obtener una evaluación de las actividades o talleres presentados a la misma.

Para incentivar a los asistentes a la descarga y uso de la aplicación se propondrá un juego competitivo que consistirá en una serie de preguntas tipo test para cada actividad o taller que se desarrolle en la feria, con la posibilidad de mostrar mediante pantallas expositivas la clasificación en tiempo real.

Por último, la evaluación de las actividades se realizará mediante una breve encuesta que se mostrará al término de cada batería de preguntas pertenecientes al juego. De esta forma se garantizará que cada usuario realiza la encuesta de evaluación.

Toda esta información será almacenada en una base de datos MySQL alojada en un servidor para tal efecto. Posteriormente los datos podrán ser tratados para realizar los estudios que se estimen oportunos.

Titulación	Ingeniero Técnico de Telecomunicación, especialidad en Telemática
Intensificación	
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Febrero - 2014

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

Capítulo 1 - Introducción

La participación en eventos es una alternativa que utilizan las organizaciones como escaparate para darse a conocer y fomentar el interés de los asistentes en una determinada actividad.

Una de las grandes debilidades de este sistema es la incapacidad para cuantificar el impacto de la actividad desarrollada por los diferentes agentes participantes de manera rápida y sencilla. En este sentido, una evaluación por parte de los asistentes ayudaría a la mejoría de los distintos talleres repercutiendo de manera positiva en el desarrollo de futuras ediciones.

1.1. Motivación

Por este motivo se plantea el desarrollo de una aplicación para dispositivos móviles aprovechando el elevado porcentaje de usuarios de *smartphones* en España. De este modo, la gran mayoría de los asistentes podrán tener acceso a ella dejando constancia de sus impresiones y sugerencias. Además, esta tarea se puede llevar a cabo de una manera menos compleja y requiriendo también de una menor demora en su cumplimentación.

1.2. Objetivos del proyecto

Desarrollar una aplicación para dispositivos móviles que permita, por un lado, medir la afluencia de asistentes a una feria y, por otro, obtener una evaluación de las actividades o talleres presentados a la misma. Para esta actividad se implementará una aplicación que dispondrá de adaptaciones para las principales plataformas de dispositivos móviles existentes en el mercado.

La aplicación se podrá descargar a través del enlace mediante diversos códigos *QR* disponibles en puntos claramente accesibles. Cada usuario tendrá que crear un perfil (básicamente un nombre de usuario y opcionalmente algunos datos personales como la edad, estudios, etc.). Con esto se podrá tener una estimación de la afluencia de público al evento.

Para incentivar a los asistentes a la descarga y uso de la aplicación se propondrá un juego competitivo que consistirá en una serie de preguntas tipo test para cada actividad o taller. El asistente o grupo de asistentes (se podría contemplar la competición por equipos) que consiga una puntuación mayor gana. A modo de aliciente extra, la aplicación mostrará en tiempo real el estado del ranking de puntuaciones para fomentar la competencia en el

juego. Si la feria dispone de pantallas expositivas también se podría mostrar esta información en las mismas en tiempo real.

En cuanto a la evaluación de cada actividad o taller, la misma aplicación presentará una breve encuesta para valorar la misma. Para garantizar que cada usuario realiza la encuesta, no se podrá acceder a la parte del juego para un taller o actividad hasta que no se haya rellenado correctamente la encuesta de evaluación.

Capítulo 2 – Entorno de desarrollo

Este capítulo tiene por finalidad realizar un análisis superfluo de las aplicaciones, tecnologías y lenguajes de programación que han sido necesarios para alcanzar el objetivo final del proyecto. Durante la descripción de cada uno de ellos se justificará el motivo por el cual se ha elegido de entre todas las opciones disponibles en el mercado, además de familiarizar al lector con los conceptos que se utilizarán a lo largo de la memoria.

2.1. Android OS

La telefonía móvil está cambiando la sociedad actual de una forma tan significativa como lo ha hecho Internet. Esta revolución no ha hecho más que empezar, los nuevos terminales ofrecen unas capacidades similares a un ordenador personal, lo que permite que puedan ser utilizados para leer nuestro correo o navegar por Internet. Pero a diferencia de un ordenador, un teléfono móvil siempre está en el bolsillo del usuario. Esto permite un nuevo abanico de aplicaciones mucho más cercanas al usuario. De hecho, muchos autores coinciden en que el nuevo ordenador personal del siglo XXI será un terminal móvil.

El lanzamiento de *Android* como nueva plataforma para el desarrollo de aplicaciones móviles causó una gran expectación y está teniendo una importante aceptación tanto por los usuarios como por la industria hasta el punto de convertirse en el sistema operativo móvil más utilizado con una cuota de mercado del 79% del total de los terminales móviles vendidos a nivel mundial[1]. En España las cifras aumentan considerablemente y 9 de cada 10 *smartphones* nuevos utilizan *Android OS*[2]. Éste es el principal motivo por el que se ha decido implementar para esta plataforma la aplicación objeto de esta memoria.

2.1.1. Un poco de historia.

Google adquiere *Android Inc*. en el año 2005. Se trataba de una pequeña compañía, que acababa de ser creada, orientada a la producción de aplicaciones para terminales móviles. Ese mismo año empiezan a trabajar en la creación de una máquina virtual *Java* optimizada para móviles (*Dalvik VM*).

[1] Estudio realizado por *Gartner Group* para ventas de *smartphones* nuevos durante el segundo cuatrimestre de 2013.
[2] Datos para el primer trimestre de 2013 según *Kantar Worldpanel*.

En el año 2007 se crea el consorcio *Open Handset Alliance* con el objetivo de desarrollar estándares abiertos para móviles. Está formado por *Google*, *Intel*, *Texas Instruments*, *Motorola*, *T-Mobile*, *Samsung*, *Ericsson*, *Toshiba*, *Vodafone*, *NTT DoCoMo*, *Sprint Nextel* y otros. Una pieza clave de esta alianza es promover el diseño y difusión de la plataforma *Android* publicando una parte importante de su propiedad intelectual como código abierto bajo licencia *Apache v2.0*.

En noviembre del 2007 se lanza una primera versión del *Android SDK*. Al año siguiente aparece el primer móvil con *Android*. En octubre *Google* libera el código fuente de *Android* principalmente bajo licencia de código abierto *Apache* (licencia *GPL v2* para el núcleo). Ese mismo mes se abre *Android Market*, para la descarga de aplicaciones. En abril del 2009 *Google* lanza la versión 1.5 del *SDK* que incorpora nuevas característica como el teclado en pantalla. A finales del 2009 se lanza la versión 2.0 y durante el 2010 las versiones 2.1, 2.2 y 2.3.

Durante el año 2010 *Android* se consolida como uno de los sistemas operativos para móviles más utilizados, con resultados cercanos al *iPhone* e incluso superando al sistema de *Apple* en EE.UU.

En el 2011 se lanzan la versión 3.0, 3.1 y 3.2 específica para tabletas y la 4.0 tanto para móviles como para tabletas. Durante este año *Android* se consolida como la plataforma para móviles más importante alcanzando una cuota de mercado superior al 50%.

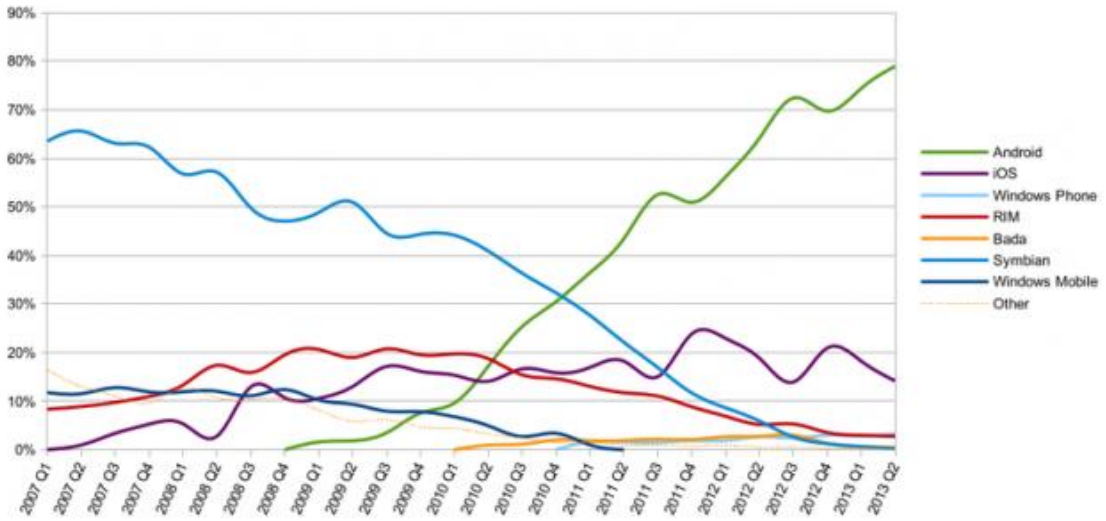

Ilustración 1 - Ventas de smartphones a nivel mundial

En 2012 *Google* cambia su estrategia en su tienda de descargas online, reemplazando *Android Market* por *Google Play Store* y unificando en un solo portal tanto la descarga de aplicaciones como de contenidos. En este año aparecen las versiones 4.1 y 4.2 del *SDK*.

Android mantiene su espectacular crecimiento, alcanzando a finales de año una cuota de mercado del 75%.

2.1.2. Cuota de mercado de las versiones

Una de las cosas más importantes antes de desarrollar una aplicación es realizar un análisis de las necesidades y recursos de los que hará uso. Desde que saliera la primera versión se han ido añadiendo y modificando utilidades dando lugar al nacimiento de nuevas versiones de *Android OS*. Si se desea que la aplicación pueda ser ejecutada en el mayor número de dispositivos, se debe realizar un estudio previo de la cuota de mercado con las diferentes opciones y ésta deberá ser compatible con la versión más antigua a la que se requiera dar soporte.

Actualmente, la versión 2.2 –nombre en clave *Froyo*- es la que más tiempo lleva en el mercado y cuya cuota de dispositivos, cercana al 2,2%, no es despreciable. Por este mismo motivo y llegando a la conclusión de que la aplicación a desarrollar no necesita de unos requerimientos añadidos en versiones posteriores, se ha decido realizarla para ésta llegando así a casi el 100% de los dispositivos móviles con *Android OS*.

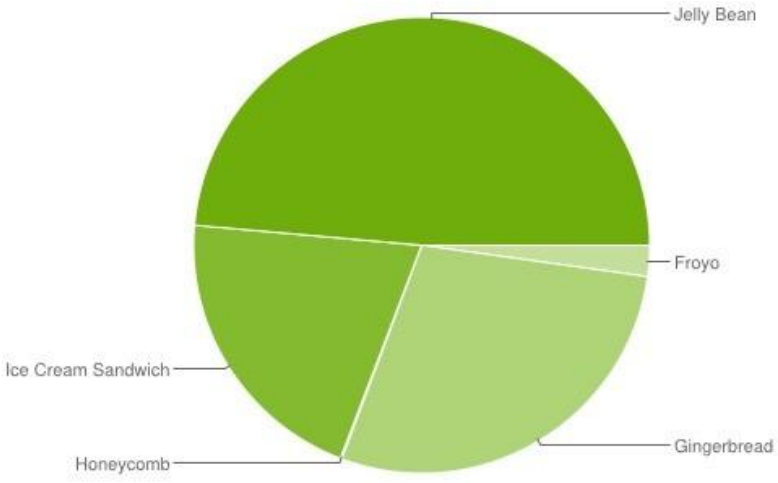

Ilustración 2 - Cuota de mercado para las versiones de Android OS

2.1.3. Características de Android

Como es sabido, existen muchas plataformas para móviles (*iOS*, *Symbian*, *Windows Phone*, *Blackberry OS*, *Firefox OS*, etc), sin embargo *Android* presenta una serie de características que lo hacen diferente. Es el primero que combina en una misma solución las siguientes cualidades:

- **Plataforma realmente abierta**. Es una plataforma de desarrollo libre basada en *Linux* y de código abierto. Una de sus grandes ventajas es que se puede usar y customizar el sistema sin pagar royalties.

- **Adaptable a cualquier tipo de hardware**. *Android* no ha sido diseñado exclusivamente para su uso en teléfonos y tabletas. Hoy en día podemos encontrar relojes, cámaras, electrodomésticos y gran variedad de sistemas embebidos que se basan en este sistema operativo. Este hecho tiene sus evidentes ventajas, pero también va a suponer un esfuerzo adicional al programador. La aplicación ha de funcionar correctamente en dispositivos con gran variedad de tipos de entrada, pantalla, memoria, etc. Esta característica contrasta con la estrategia de *Apple*. En *iOS* tenemos que desarrollar una aplicación para *iPhone* y otra diferente para *iPad*.
- **Portabilidad asegurada**. Las aplicaciones finales son desarrolladas en Java lo que nos asegura que podrán ser ejecutadas en cualquier tipo de CPU, tanto presente como futuro. Esto se consigue gracias al concepto de máquina virtual.
- **Arquitectura basada en componentes inspirados en Internet.** Por ejemplo, el diseño de la interfaz de usuario se hace en *XML*, lo que permite que una misma aplicación se ejecute en un móvil de pantalla reducida o en un TV.
- **Filosofía de dispositivo siempre conectado a Internet**.
- **Gran cantidad de servicios incorporados**. Por ejemplo, localización basada tanto en GPS como en redes, bases de datos con *SQL*, reconocimiento y síntesis de voz, navegador, multimedia.
- **Aceptable nivel de seguridad**. Los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja que hereda de Linux. Además, cada aplicación dispone de una serie de permisos que limitan su rango de actuación (servicios de localización, acceso a Internet, etc.)
- **Optimizado para baja potencia y poca memoria**. Por ejemplo, *Android* utiliza la Máquina Virtual *Dalvik*. Se trata de una implementación de *Google* de la máquina virtual de *Java* optimizada para dispositivos móviles.
- **Alta calidad de gráficos y sonido**. Gráficos vectoriales suavizados, animaciones inspiradas en *Flash*, gráficos en 3 dimensiones basados en *OpenGL*. Incorpora los *codecs* más comunes de audio y vídeo, incluyendo *H.264* (*AVC*), *MP3*, *AAC*, etc.

En conclusión *Android* nos ofrece una forma sencilla y novedosa de implementar potentes aplicaciones para diferentes tipos de dispositivos.

2.1.4. Arquitectura de Android

El siguiente gráfico muestra la arquitectura de *Android*. Como se puede ver está formada por cuatro capas cada una de las cuales basadas en software libre.

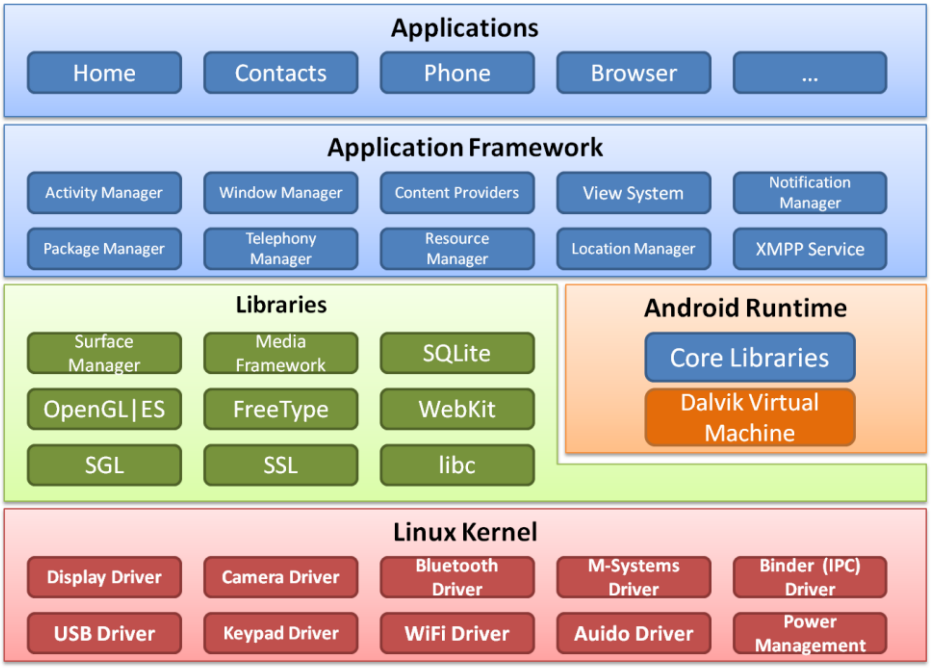

Ilustración 3 - Arquitectura Android OS

2.1.4.1. Núcleo Linux

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

Actúa como capa de abstracción entre el hardware y el resto de la pila. Por lo tanto, es la única que es dependiente del hardware.

2.1.4.2. Runtime de Android

Está basado en el concepto de máquina virtual utilizado en *Java*. Dado las limitaciones de los dispositivos donde ha de correr *Android* (poca memoria y procesador limitado) no fue posible utilizar una máquina virtual Java estándar. *Google* tomó la decisión de crear una nueva, la máquina virtual *Dalvik*, que respondiera mejor a estas limitaciones.

Algunas características de la máquina virtual *Dalvik* que facilitan esta optimización de recursos son:

- Ejecuta ficheros *Dalvik* ejecutables (.dex) –formato optimizado para ahorrar memoria-.
- Está basada en registros.
- Cada aplicación corre en su propio proceso *Linux* con su propia instancia de la máquina virtual *Dalvik*.

- Delega al *kernel* de *Linux* algunas funciones como *threading* y el manejo de la memoria a bajo nivel.

También se incluye en el *Runtime* de *Android* el "*core libraries*" con la mayoría de las librerías disponibles en el lenguaje *Java*.

2.1.4.3. Librerías nativas

Android incluye un conjunto de librerías compiladas en C/C++ código nativo del procesador. Algunas de estas librerías son:

- ***System C library***: Una derivación de la librería BSD de C estándar (libc), adaptada para dispositivos embebidos basados en *Linux*.
- ***Media Framework***: Librería basada en *PacketVideo's OpenCORE*. Soporta *codecs* de reproducción y grabación de multitud de formatos de audio, vídeo e imágenes.
- ***Surface Manager***: Maneja el acceso al subsistema de representación gráfica en 2D y 3D.
- ***WebKit***: Soporta un moderno navegador web utilizado en el navegador *Android* y en la vista *webview*. Se trata de la misma librería que utiliza *Google Chrome* y *Safari* de *Apple*.
- ***SGL***: Motor de gráficos 2D.
- **Librerías 3D**: Implementación basada en *OpenGL* ES 1.0 API. Las librerías utilizan el acelerador hardware 3D si está disponible, o el software altamente optimizado de proyección 3D.
- ***FreeType***: Fuentes en bitmap y renderizado vectorial.
- ***SQLite***: Potente y ligero motor de bases de datos relacionales disponible para todas las aplicaciones.
- ***SSL***: Proporciona servicios de encriptación *Secure Socket Layer*.

2.1.4.4. Entorno de aplicación

Proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones (sensores, localización, servicios, barra de notificaciones,). Esta capa ha sido diseñada para simplificar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas (sujetas a las restricciones de seguridad). Este mismo mecanismo permite a los usuarios reemplazar componentes.

Una de las mayores fortalezas del entorno de aplicación de *Android* es que se aprovecha el lenguaje de programación *Java*. El *SDK* de *Android* no acaba de ofrecer todo lo disponible para su estándar del entorno de ejecución *Java* (*JRE*), pero es compatible con una fracción muy significativa de la misma.

Los servicios más importantes que incluye son:

- ***Views***: Extenso conjunto de vistas que forman la parte visual de la aplicación.
- ***Resource Manager***: Proporciona acceso a recursos que no son en código.
- ***Activity Manager***: Maneja el ciclo de vida de las aplicaciones y proporciona un sistema de navegación entre ellas.
- ***Notification Manager***: Permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.
- ***Content Providers***: Mecanismo sencillo para acceder a datos de otras aplicaciones (como los contactos).

2.1.4.5. Aplicaciones

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

Normalmente las aplicaciones *Android* están escritas en *Java*. Para desarrollar aplicaciones en Java podemos utilizar el *Android SDK*. Existe otra opción consistente en desarrollar las aplicaciones utilizando C/C++. Para esta opción podemos utilizar el *Android NDK* (*Native Development Kit*).

2.1.5. Elementos de un proyecto Android

Un proyecto *Android* está formado básicamente por un descriptor de la aplicación (*AndroidManifest.xml*), el código fuente en *Java* y una serie de ficheros con recursos. Cada elemento se almacena en una carpeta específica siguiendo una estructura común a todos los proyectos existentes.

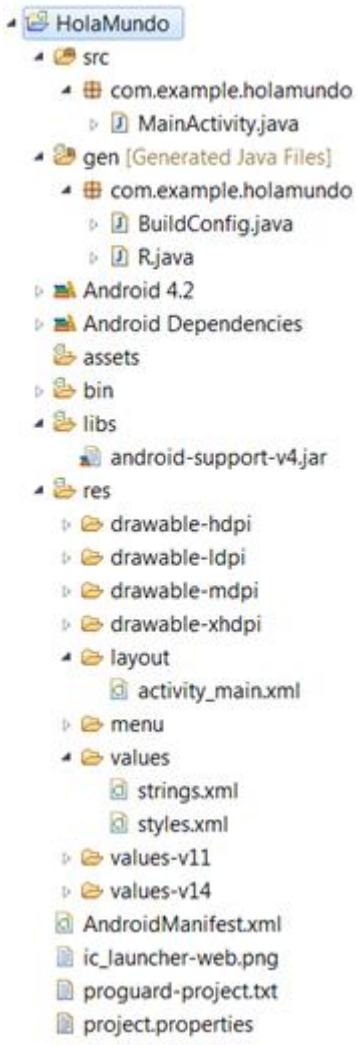

Ilustración 4 - Estructura proyecto Android

Seguidamente se procederá a describir cada uno de los elementos para saber qué contiene y cuál es su cometido:

- ***/src/***: Carpeta que contiene el código fuente de la aplicación.
- ***/gen/***: Carpeta que contiene el código generado de forma automática por el *SDK*. En su interior se encuentra:
  - ***BuildConfig.java***: Define la constante *DEBUG* para que desde *Java* se pueda saber si la aplicación está en fase de desarrollo.
  - ***R.java***: Define una clase que asocia los recursos de la aplicación con identificadores para que puedan ser accedidos desde *Java*.
- ***/Android* X.X/**: Código *JAR*, el *API* de *Android* según la versión seleccionada.
- ***/Android Dependencies/***: Librerías asociadas al proyecto.
- ***/assets/***: Carpeta que puede contener una serie arbitraria de ficheros o carpetas que podrán ser utilizados por la aplicación (ficheros de datos, fuentes, etc).

- ***/bin/*:** En esta carpeta se compila el código y se genera el .apk, fichero comprimido que contiene la aplicación final lista para ser instalada.
- ***/libs/*:** Código *JAR* con las librerías que serán usadas en el proyecto. En este caso concreto, la librería añadida sirve para dar soporte a las nuevas funcionalidades que surgieron en versiones posteriores a la *API* 4.
- ***/res/*:** Carpeta que contiene los recursos usados por la aplicación. Las subcarpetas pueden contener un sufijo para especificar que el recurso solo se cargue si cumple una condición. Los diferentes tipos de recursos de deberán distribuir entre las siguientes carpetas:
  - */res/drawable/.* Contiene los ficheros de imágenes que se van a utilizar.
  - */res/layout/*. Almacena los ficheros *XML* con las vistas de la aplicación. Las vistas nos permitirán configurar las diferentes pantallas que compondrán la interfaz de usuario de la aplicación.
  - */res/menu/.* Ficheros *XML* con los menús de cada actividad.
  - */res/values/*. Contiene ficheros *XML* para indicar valores del tipo *string* (*strings.xml*), color (*colors.xml*) o estilo (*styles.xml*). De esta manera se podrán cambiar los valores sin necesidad de ir al código fuente.
  - */res/xml/*. Contiene los ficheros *XML* utilizados por la aplicación.
  - */res/raw/*. Carpeta para ficheros adicionales que no se encuentran en formato *XML* y que no se incluyan en el resto de carpetas de recursos.
- ***AndroidManifest.xml*:** Este fichero describe la aplicación *Android*. En él se indican las actividades, intenciones, servicios y proveedores de contenido de la aplicación. También se declaran los permisos que requerirá la aplicación.
- ***Proguard-project.txt*:** Fichero de configuración de la herramienta *ProGuard*, que permite optimizar y ofuscar el código generado para evitar la ingeniería inversa.

2.1.6. Componentes de una aplicación

Existen una serie de elementos genéricos clave que resultan imprescindibles para desarrollar aplicaciones en *Android*. A continuación se detallarán cuales son los más importantes:

- ***Activity*:** Una aplicación en *Android* va a estar formada por un conjunto de elementos básicos de visualización. A cada uno de estos elementos se le conoce como *actividad* y su función principal es la creación del interfaz de usuario. Las diferentes *actividades* creadas serán independientes entre sí aunque trabajarán para un objetivo común.
- ***View*:** Las *vistas* son los elementos que componen la interfaz de usuario de una aplicación. Habitualmente se suelen definir las vistas mediante etiquetas

utilizando un fichero *XML* –de forma similar al diseño de una página web utilizando código *HTML*- aunque también se pueden definir por código *Java*.

- ***Service*:** Los *servicios* son componentes que se ejecutan en segundo plano y puesto que no existe una interacción con el usuario no será necesario definir una interfaz gráfica. Su funcionamiento es muy parecido al de un *demonio* en *Unix* o un *servicio* en *Windows*. En *Android* disponemos de dos tipos de servicios, los locales, que pueden ser utilizados por aplicaciones del mismo terminal y, los remotos, que pueden ser utilizados desde otros terminales
- ***Content Provider*:** Los *proveedores de contenido* es el mecanismo estándar definido para que las aplicaciones puedan compartir datos sin necesidad de comprometer la seguridad del sistema de ficheros. Haciendo uso de ello podremos acceder a datos de otras aplicaciones como la lista de contactos o las últimas llamadas realizadas.
- ***Broadcast Receiver*:** Un *receptor de anuncios* es un componente capaz de detectar y reaccionar ante eventos de tipo *broadcast*. Existen muchos eventos de este tipo originados por el sistema como el anuncio de batería baja o llamada entrante. En cualquier caso, también se podrá lanzar este tipo de eventos desde las aplicaciones.
- ***Intent*:** Un *intención* es el elemento básico de comunicación entre los distintos componentes *Android* que hemos descrito anteriormente y representa la voluntad de realizar una acción. Se pueden entender como los *mensajes* o *peticiones* que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones.

2.2. XAMPP

Es una herramienta de software libre independiente de la plataforma que consiste principalmente en la base de datos *MySQL*, el servidor web *Apache* y los intérpretes para lenguajes de *script*: *PHP* y *Perl*. También incluye otros módulos como *OpenSSL* y *phpMyAdmin*. En el proyecto que nos atañe será necesario configurar todas las herramientas que nos facilita esta herramienta de fácil instalación a excepción del intérprete *Perl* que no será utilizado. El nombre proviene del acrónimo de **X** (para cualquiera de los diferentes sistemas operativos), ***A**pache*, ***M**ySQL*, ***P**HP*, ***P**erl*.

El programa está liberado bajo la licencia *GNU* y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente *XAMPP* está disponible para *Microsoft Windows*, *GNU/Linux*, *Solaris* y *MacOS X*.

2.2.1. Base de datos MySQL

Es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. *MySQL AB* —desde enero de 2008 una subsidiaria de *Sun Microsystems* y ésta a su vez de *Oracle Corporation* desde abril de 2009— desarrolla *MySQL* como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la *GNU GPL* para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en *ANSI C*.

Al contrario de proyectos como *Apache*, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, *MySQL* es patrocinado por una empresa privada, que posee el *copyright* de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado.

Su popularidad como aplicación web está muy ligada a *PHP*, que a menudo aparece en combinación con *MySQL*.

2.2.2. Servidor web Apache

Es un servidor web *HTTP* de código abierto, para plataformas *Unix* (*BSD*, *GNU/Linux*, etc.), *Microsoft Windows*, *Macintosh* y otras, que implementa el protocolo *HTTP*/1.1.

El servidor *Apache* se desarrolla dentro del proyecto *HTTP Server* (httpd) de la *Apache Software Foundation*. Es altamente configurable gracias a su arquitectura modular pudiendo ser ampliado con la inclusión de módulos externos al servidor básico.

Ampliamente aceptado en la red -desde 1996 es el servidor *HTTP* más usado-, alcanzó su máxima cuota de mercado en 2005 siendo empleado en el 70% de los sitios web en el mundo. Sin embargo, en los últimos años ha sufrido un descenso en su cuota de mercado[3].

2.2.3. PHP: PHP Hypertext Pre-processor

Es un lenguaje de programación originalmente diseñado para el desarrollo web de contenido dinámico siendo uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento *HTML* en lugar de llamar a un archivo externo que procese los datos. Fue creado por Rasmus Lerdorf

[3] Estadísticas históricas y de uso diario proporcionadas por Netcraft.

aunque actualmente se encarga de su desarrollo *The PHP Group* publicándolo bajo la *PHP License*, una licencia considerada como software libre por la *Free Software Foundation*.

Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de *PHP*. Éste procesa el script solicitado que generará el contenido de manera dinámica -por ejemplo obteniendo información de una base de datos-. El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente.

Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún coste.

El lenguaje *PHP* se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores siendo el módulo *Apache* más popular entre las computadoras que utilizan a éste como servidor web.

2.2.4. phpMyAdmin

Es una herramienta escrita en *PHP* cuyo propósito es facilitar la administración de bases de datos *MySQL* haciendo uso de una interfaz web a través de la cual se puede entre otras muchas opciones crear y eliminar bases de datos o crear, eliminar y alterar tablas.

Este proyecto se encuentra vigente desde el año 1998 bajo licencia *GPL*, siendo el mejor evaluado en la comunidad de descargas de *SourceForge.net* como la descarga del mes de diciembre del 2002.

2.3. Eclipse + ADT

Es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés *IDE*), como el *IDE* de *Java* llamado *Java Development Toolkit* (*JDT*) y el compilador (*ECJ*) que se entrega como parte de *Eclipse* (y que son usados también para desarrollar el mismo *Eclipse*).

Eclipse fue desarrollado originalmente por *IBM* como el sucesor de su familia de herramientas para *VisualAge*. *Eclipse* es ahora desarrollado por la *Fundación Eclipse*, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse fue liberado originalmente bajo la *Common Public License*, pero después fue re-licenciado bajo la *Eclipse Public License*. La *Free Software Foundation* ha dicho que ambas licencias son licencias de software libre.

El entorno de desarrollo integrado (*IDE*) de *Eclipse* emplea módulos (en inglés *plug-in*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no.

Para este caso particular será necesario instalar el módulo para el desarrollo de aplicaciones *Android*. *ADT plug-in* (*Android Development Tools*) está diseñado para brindar al programador un entorno potente e integrado ampliando las posibilidades de *Eclipse*. Entre otras cosas, gracias a él se podrá configurar rápidamente un nuevo proyecto *Android*, crear una interfaz de usuario, depurar las aplicaciones usando las herramientas del *SDK Android* o exportar archivos .apk firmados (o no) con el fin de distribuir la aplicación.

El desarrollo en *Eclipse* con *ADT* es muy recomendable y es la manera más rápida para empezar.

2.4. JSON: JavaScript Object Notation

Es un formato ligero para el intercambio de datos completamente independiente del lenguaje de programación utilizado, lo que facilita la comunicación entre distintos dispositivos. La simplicidad de *JSON* ha dado lugar a la generalización de su uso, especialmente como alternativa a *XML* en *AJAX*. El beneficio de *JSON* frente a otras alternativas es que representa mejor la estructura de los datos y requiere menos codificación y procesamiento.

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor.
- Una lista ordenada de valores.

2.5. Código QR

Es un código de barras bidimensional creado por la compañía japonesa *Denso Wave*, subsidiaria de *Toyota*, en 1994. Se caracteriza por los tres cuadrados que se encuentran en las esquinas y que permiten detectar la posición del código al lector. La sigla *QR* se deriva de la frase inglesa *Quick Response* (Respuesta Rápida en español), pues los creadores tenían como objetivo que el código permitiera que su contenido se leyera a alta velocidad.

Ilustración 5 - Código QR para el taller "Pila de combustible"

Para nuestro caso particular, el código *QR* permitirá el acceso a las cuestiones de un taller en particular de forma rápida e inequívoca.

Capítulo 3 – Desarrollo del proyecto

En este capítulo se explicarán la estructura utilizada para el correcto desarrollo del proyecto y la implementación de la aplicación realizando un análisis detallado de las características más relevantes sin entrar en profundidad en las líneas de código empleadas.

3.1. Arquitectura del proyecto

El proyecto está compuesto de dos partes claramente diferencias. Por un lado está la aplicación cliente donde el usuario contestará a las preguntas que se le planteen y podrá informarse tanto de sus estadísticas personales como de su posición en un ranking. Por otro lado se tiene la parte del servidor donde se almacenan todos los datos relevantes para un correcto funcionamiento de la aplicación. Ésta brindará la información de que disponga cuando le sea requerida por parte del cliente.

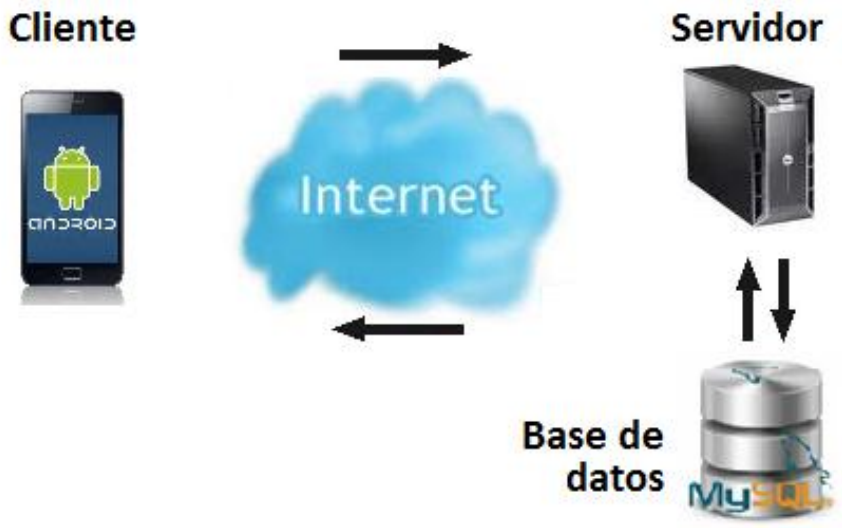

Ilustración 6 - Arquitectura del proyecto

3.2. Cliente

Como se indica en la primera parte de esta memoria, la aplicación cliente se ha desarrollado para *Android OS*. A lo largo de los siguientes apartados se podrá comprobar cómo la estructura del proyecto es similar a la vista en apartados anteriores. En cualquier caso se realizará un análisis con mayor detenimiento de cada uno de los elementos que componen la aplicación.

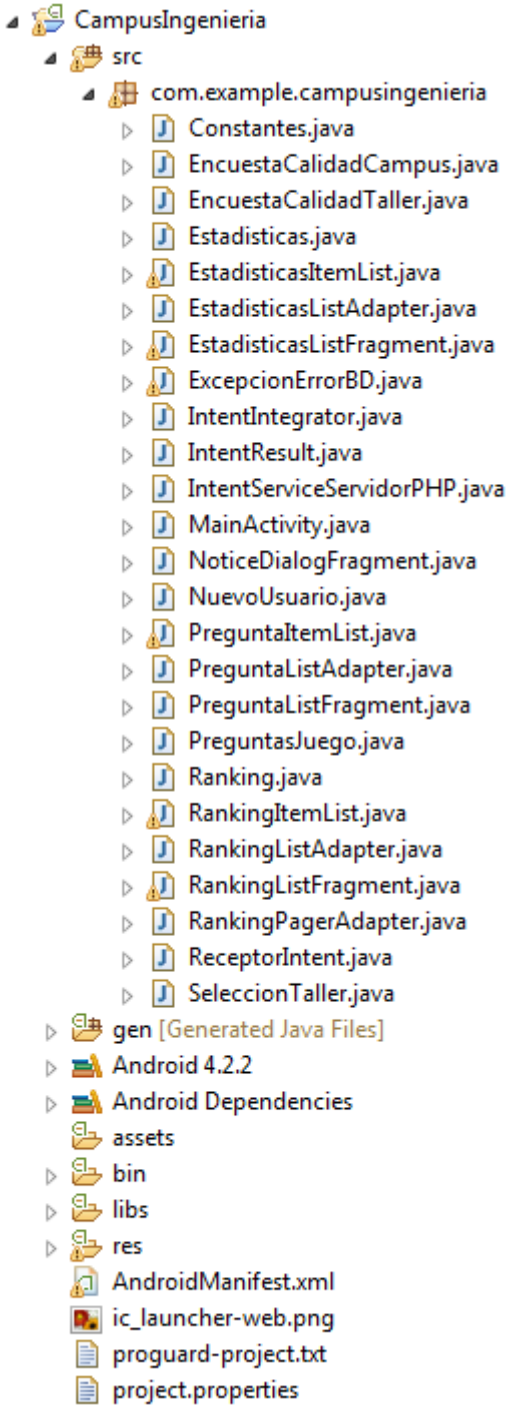

Ilustración 7 - Estructura de la aplicación cliente

Para terminar de documentar de forma general la estructura de la aplicación cliente se muestra en la siguiente ilustración el diagrama *UML* (*Unified Modeling Language*) de las clases que forman parte del proyecto donde se detalla las relaciones estructurales y de herencia.

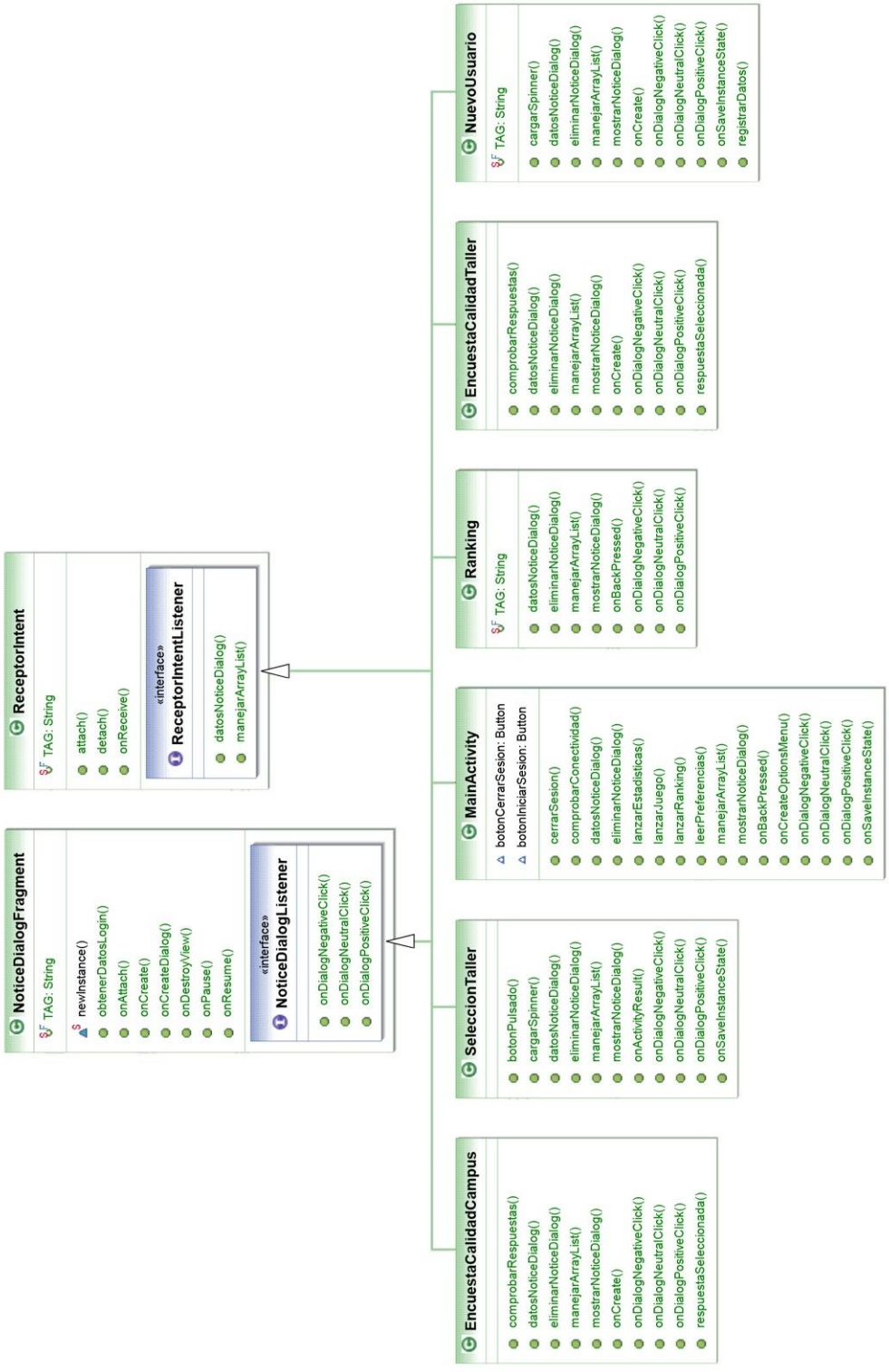

Ilustración 8 - Diagrama de clases UML – Parte 1

Ilustración 9 - Diagrama de clases UML – Parte 2

3.2.1. Vistas principales

La aplicación está formada por una actividad principal –*MainActivity.java*- a modo de menú a partir de la cual se puede acceder al resto dependiendo del estado de ejecución en el que se encuentre. En todo momento se ha asegurado que se cumple el ciclo de vida de las actividades que forman parte de la aplicación.

El usuario podrá realizar 4 acciones a destacar:

- Registrar un usuario nuevo: *NuevoUsuario.java* permite dar de alta a nuevos usuarios para que se pueda llevar cuenta de los avances individuales. Durante el registro se pedirán los datos para acceder a la parte restringida y datos personales para realizar análisis estadísticos. En esta parte se diferenciará si el usuario asiste a la feria en calidad de visitante o ponente.
- Participar en el juego: Aunque la aplicación tiene como finalidad recopilar las valoraciones que los asistentes tienen de los diferentes talleres participantes en la feria, también se ha desarrollado un juego de preguntas y respuestas para que sea más ameno y fomente la participación. Únicamente podrán acceder a este apartado aquellos que tengan el rol de visitante. En caso contrario se mostrará una encuesta para que los ponentes den su opinión sobre la organización de la feria.
- Mostrar estadísticas personales: Esta acción muestra el porcentaje de aciertos en cada uno de los bloques de preguntas. Está opción no aparecerá habilitada en el caso de que el usuario registrado sea un ponente ya que la participación en el juego está solo disponible para los visitantes.
- Mostrar ranking: Al estilo *tab* –navegación entre vistas deslizando el dedo- se muestran las diferentes clasificaciones atendiendo a distintos criterios de organización. Todos los usuarios de la aplicación podrán listar la clasificación general tanto individual como por centros de educación. Aquellos que estén registrados como visitantes además podrán acceder a clasificaciones filtradas por edad, centro y curso.

En cualquier momento el usuario podrá girar el dispositivo para mayor comodidad sin que se pierda el estado de la actividad que se esté ejecutando, a excepción de aquellas en las que la legibilidad de los datos a mostrar se vean comprometidos, como la actividad que representa los datos del ranking o las estadísticas personales.

3.2.2. Alertas

Todas las alertas que aparecen, tanto aquellas para informar de algún error como las que piden al usuario que introduzca algún dato necesario –por ejemplo para autenticarse-, se gestionan a partir de la misma clase *NoticeDialogFragment.java*. En ella se declara una interfaz que necesariamente tendrán que implementar las actividades que creen un objeto del tipo *NoticeDialogFragment*. De este modo, una vez que se lance la alerta, pulsando los botones que se presenten se podrá devolver el control de la aplicación a la actividad llamante y actuar en consecuencia.

```
public class NoticeDialogFragment extends DialogFragment {
    public static final String TAG = "NoticeDialogFragment";

    private NoticeDialogListener mListener;
    private int tipo;
    private EditText userText, passText;
    private String userTemp, passTemp;


    public interface NoticeDialogListener {
        public void onDialogPositiveClick(NoticeDialogFragment dialog, int tipo);
        public void onDialogNegativeClick(NoticeDialogFragment dialog, int tipo);
        public void onDialogNeutralClick(NoticeDialogFragment dialog, int tipo);
    }

    // Override the Fragment.onAttach() method to instantiate the NoticeDialogListener
    public void onAttach(Activity activity) {

    static NoticeDialogFragment newInstance(int tipo, String titulo, String mensaje,

    public void onCreate(Bundle savedInstanceState) {

    public Dialog onCreateDialog(Bundle savedInstanceState) {

    /**
     * Solución problema #17423. Bug en la librería de compatibilidad
     */
    public void onDestroyView() {

    public void onPause(){

    public void onResume(){

    public String[] obtenerDatosLogin(){
}
```

Ilustración 10 - Clase para lanzar y gestionar alertas

3.2.3. Comunicación con el servidor

La comunicación con el servidor se gestiona mediante la clase *IntentServiceServidorPHP.java* que hereda de *IntentService*, por lo que tendremos que implementar el método *onHandleIntent()* para manejar las intenciones como proceda en cada momento. Se ha elegido este método para evitar bloquear el hilo principal de ejecución de la aplicación, lo cual puede provocar fallos inesperados en tiempo de

ejecución. A pesar de que existe otra forma de manejar estas situaciones creando tareas asíncronas con *AsyncTask* en cada una de las actividades, decantarse por usar *IntentService* se debe a la intención de reutilizar el mayor código posible, aumentando la comprensión del código.

Decantarse por *IntentService* tiene una desventaja. Mientras que con *AsyncTask*, la respuesta del servidor se puede manejar fácilmente en la misma actividad que demanda esa información ya que proporciona métodos que se ejecutan en el hilo principal de la aplicación, mediante el uso de *IntentService* será necesario enviar mensajes de tipo *broadcast* para comunicar eventos al hilo principal. Para procesar este tipo de mensajes se facilita una clase llamada *BroadcastReceiver*.

```
public class IntentServiceServidorPHP extends IntentService {
    private final static String TAG = "IntentServiceServidorPHP";
    private InputStream is;
    private String salida;
    private String action;

    public IntentServiceServidorPHP() {...}

    protected void onHandleIntent(Intent intent) {...}

    // METODOS PARA REALIZAR LA CONEXION
    private ArrayList<NameValuePair> prepararPostParametros(Intent intent) {
        ArrayList<NameValuePair> parametros = new ArrayList<NameValuePair>();

        if (intent.hasExtra("idParam")) {
            ArrayList<String> idParam = new ArrayList<String>();
            idParam = intent.getStringArrayListExtra("idParam");

            for (int i = 0; i < idParam.size(); i++) {
                String nombre = idParam.get(i);
                String valor = intent.getStringExtra(nombre);
                parametros.add(new BasicNameValuePair(nombre, valor));
                Log.i(TAG, "Parametros enviados -> nombre= "+nombre+", valor= "+valor);
            }
        }
        return parametros;
    }

    private void httpPostConnect(ArrayList<NameValuePair> parametros) {...}

    private String getPostResponse() {...}
}
```

Ilustración 11 - Clase para gestionar la comunicación con el servidor

Como se puede apreciar todas las comunicaciones con el servidor se realizan mediante el protocolo *HTTP* y los datos se pasan por método *POST*.

3.2.4. Tratamiento de los datos

Una vez se establece la comunicación, el servidor envía los datos que le ha requerido el cliente junto con el estado final de la petición. Todo esto se procesa en la clase *ReceptorIntent.java* donde se incluye un manejador de datos *JSON*. Dependiendo del tipo de comunicación que se intente establecer es posible que sea necesario crear un nuevo tipo de datos que facilite su manejo y presentación al usuario a través de la pantalla del dispositivo.

```
public class ReceptorIntent extends BroadcastReceiver {
    public static final String TAG = "ReceptorIntent";
    private ReceptorIntentListener riListener;
    private SharedPreferences prefs;

    public interface ReceptorIntentListener {...}

    public void attach(Activity activity) {...}

    public void detach() {...}

    public void onReceive(Context context, Intent intent) {...}

    private JSONArray getJSONArray(String salida) {...}

    private boolean conexionOK(JSONArray jArray) {...}

    private void almacenarDatos(JSONArray jArray, int tipoArray) {...}

    private void comprobarRegistroUsuario(JSONArray jArray) {...}

    private void comprobarLogin(JSONArray jArray) {...}

    private void comprobarRegistroRespuestas(JSONArray jArray) {...}

    private void comprobarRegistroEncuesta(JSONArray jArray){...}

    private void almacenarRanking(JSONArray jArray) {...}

    private void almacenarEstadisticas(JSONArray jArray) {...}
}
```

Ilustración 12 - Clase para manejar los datos recibidos por parte del servidor

3.2.5. Captura código QR

Para ampliar las posibilidades a la hora de elegir el taller del cual se quieren contestar las preguntas, se ha añadido esta funcionalidad que permite al usuario capturar con la cámara que incorpora su *smartphone* el código *QR* disponible en cada taller y, automáticamente, acceder al cuestionario. Además, de esta forma, se reduce la probabilidad de elegir un taller distinto al que se visita. En cualquier caso, siempre estará disponible la opción para seleccionar a partir de un desplegable.

Esta funcionalidad ha sido integrada gracias al código fuente desarrollado por *XZing*. Es necesario tener instalada la aplicación *Barcode Scanner* para obtener los datos almacenados en un código *QR*. En caso contrario se informa mediante una alerta al usuario de este hecho y se facilita un enlace para su descarga en *Google Play*.

3.2.6. Diseño icono de acceso directo

Ilustración 13 - Logotipo de la UPCT junto al icono de la aplicación

Sabiendo que la feria para la que se ha diseñado la aplicación tiene como finalidad orientar y despertar el interés por la ciencia y las carreras técnicas de los visitantes, se ha decidido realizar un diseño tomando como base el logotipo de la UPCT y transformándolo en una brújula o rosa de los vientos abstracta.

3.3. Servidor

La parte del servidor contiene un servidor web *Apache*, un intérprete *PHP* y una base de datos *MySQL*. En los siguientes apartados se detalla el desarrollo y estructura de cada uno de ellos sin entrar a valorar la puesta en marcha de todos los servicios.

3.3.1. Procesado de las respuestas. Código PHP

Para una mayor comprensión y reducir la complejidad del código *PHP* utilizado se ha optado por dividir el código en 11 archivos, encargándose cada uno de ellos de gestionar las consultas a la base de datos necesarias dependiendo de los datos recibidos por parte de la aplicación cliente. De esta forma, se podría decir que cada archivo realiza una función diferente e independiente.

Como nunca se está a salvo de usuarios malintencionados, se ha dotado al código de una mayor seguridad para evitar ataques de inyección de código *SQL*. Consiste en realizar una comprobación de los datos recibidos antes de proceder a consultar a la base de datos.

```php
<?php
//header('Content-type: text/html; charset=utf-8');

/*LOGIN*/

$usuario = $_POST['usuario'];
$passw = $_POST['password'];

require_once 'funciones_bd.php';
$db = new funciones_BD();

$salida = array();

if(!$db->comprobarConexionBD()){
    $salida[] = array("estadoConexion"=>"0");
}
else{
    $temp = $db->login($usuario,$passw);
    if(!$temp){
        $salida[] = array("estadoLogin"=>"0");
    }else{
        $salida[] = array("estadoLogin"=>"$usuario","rol"=>"$temp");
    }
}
echo json_encode($salida);
?>
```

Ilustración 14 - Ejemplo código PHP para comprobar la autenticación de usuarios

3.3.2. Base de datos

En el servidor se ha empleado una base de datos *MySQL* debido a su amplia utilización y su condición de gratuidad. Inicialmente se ha realizado un análisis de las necesidades de la aplicación cliente para posteriormente definir la estructura de las tablas y los tipos de datos utilizados en la base de datos. Para su creación se ha utilizado el software *phpMyAdmin* y a continuación se muestra su estructura:

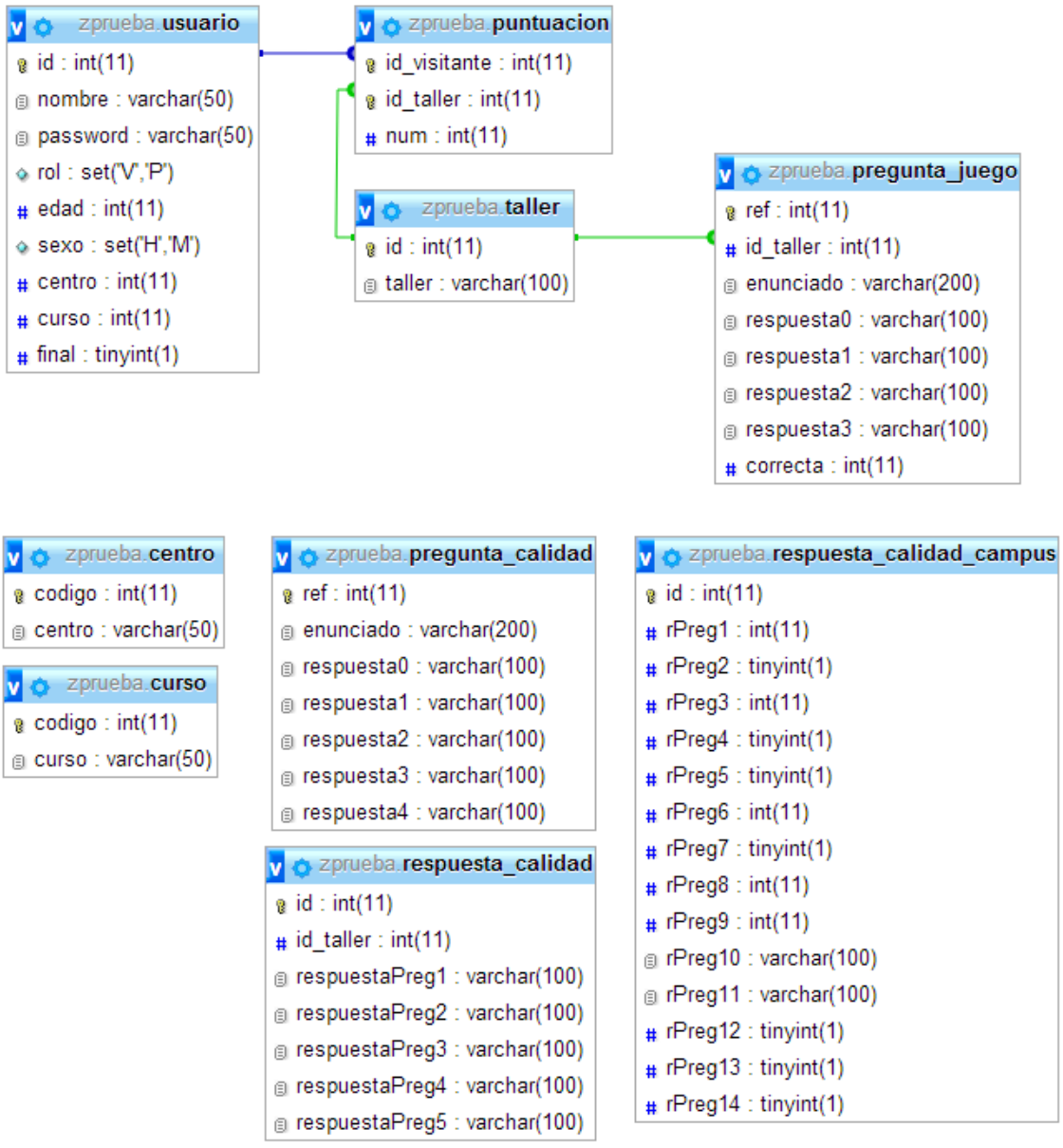

Ilustración 15 - Estructura de la base de datos

Para mejorar el rendimiento en las búsquedas se ha definido un identificador para cada entrada en cada una de las tablas. Los tipos de datos se han intentado ajustar lo máximo posible al valor que previsiblemente almacenarán evitando reservar espacio en la base de datos que difícilmente se iba a utilizar. Seguidamente se realiza una descripción de la función de cada tabla:

- **Usuario**: Cada entrada representará a un usuario que se haya dado de alta gracias a la aplicación móvil. Contiene una relación de los datos personales para filtrar las búsquedas y que, además, en un futuro se puede utilizar para obtener estadísticas

a partir de las cuales realizar estudios de rendimiento. Una vez se hayan contestado a todos los cuestionarios posibles se marcará con el valor correspondiente la variable *final*. De esta forma se sabrá cuando mostrar al usuario una encuesta de satisfacción general para la feria.

- **Centro**: Esta tabla contiene el nombre de los centros que vayan a visitar la feria. Será necesario para evitar una saturación excesiva del espacio disponible en los días de mucha afluencia y, de paso, ayudará a filtrar las búsquedas. Para ello se deberá avisar con antelación a la organización del evento.
- **Curso**: Relación de cursos para los que son aptos los temas a tratar por los talleres que componen la feria.
- **Taller**: Nombre de los talleres que participan en la feria.
- **Pregunta_juego**: Enunciados y respuestas que componen el juego de preguntas tipo test. Se señala al taller al que pertenecen y cuál de las respuestas posibles es la correcta.
- **Puntuación**: En esta tabla se almacenará el número de respuestas correctas que ha obtenido un usuario por taller.
- **Pregunta_calidad**: Como el cuestionario de calidad es fijo para todos los talleres, se ha decidido crear una tabla con los enunciados de las preguntas y las respuestas que lo componen.
- **Respuesta_calidad**: Las respuestas al cuestionario anterior se guardarán en esta tabla. Notar que para mantener la privacidad de los usuarios debido a que las encuestas de satisfacción son anónimas, no se guarda ningún dato adicional, únicamente las respuestas seleccionadas.
- **Respuesta_calidad_campus**: Al igual que la tabla anterior, en esta se guardarán también las respuestas de un cuestionario pero esta vez relativas al grado de satisfacción de la feria.

3.4. Codificación de los datos

Durante el desarrollo del proyecto se han registrado errores puntuales en la codificación de las comunicaciones entre las diferentes partes que lo componen. Finalmente el juego de caracteres utilizado ha sido *UTF-8* para poder representar correctamente los acentos y caracteres del idioma castellano. No obstante ha sido necesario modificar los siguientes apartados para adaptarlos a esta situación:

- Al establecer la comunicación *HTTP* entre cliente y servidor.
- Al establecer la conexión con la base de datos.
- Al crear las tablas de la base de datos.
- Al crear los archivos .php se tienen que codificar como *UTF-8* sin *BOM*.

Capítulo 4 – Manual de usuario

Este capítulo tiene por misión facilitar el manejo de la aplicación al usuario que se encuentra por primera vez con ella. A pesar de que haya sido realizada intentando que sea lo más intuitiva posible, ésta labor de aprendizaje no debe ser menospreciada debido a que inicialmente la aplicación está pensada para ser utilizada por niños en edad escolar.

4.1. Instalación / desinstalación de la aplicación

En *Android OS* la instalación de nuevas aplicaciones se suele realizar a través de *Google Play*. En el momento de la realización de esta memoria la aplicación desarrollada en este proyecto no estaba aún alojada en este *marketplace* por lo que se hace necesario detallar una forma alternativa para proceder a su instalación.

Cuando el archivo *CampusIngenieria.apk* esté disponible en el servidor elegido, los usuarios deberán comenzar su descarga desde el dispositivo móvil donde deseen realizar la instalación. Una vez se tenga en memoria, será necesario verificar que se tiene activada la opción de permitir aplicaciones desde orígenes desconocidos. Ésta se encuentra en "Ajustes -> Seguridad -> Orígenes desconocidos".

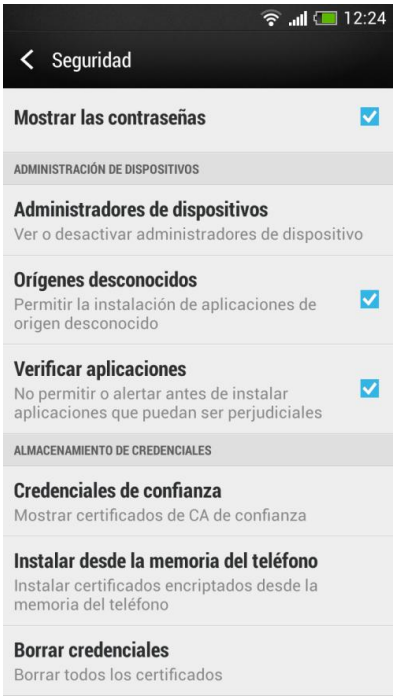

Ilustración 16 - Ajustes de seguridad

Seguidamente, mediante una herramienta de exploración de archivos, de las muchas que existen en el mercado, se iniciará la instalación. En este caso se ha utilizado *ES File Explorer*.

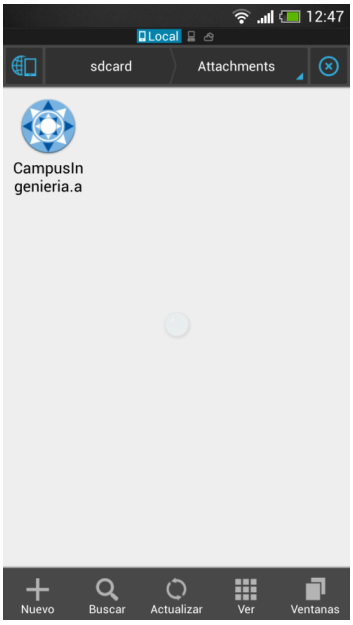

Ilustración 17 - ES File Explorer

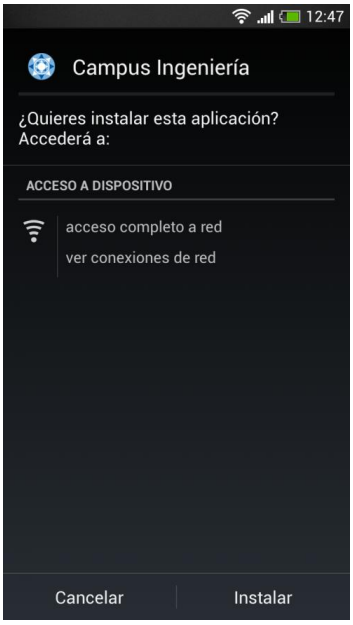

Ilustración 18 - Instalación de la aplicación y permisos requeridos

Por último, si lo que se desea es desinstalar la aplicación será necesario dirigirse a "Ajustes -> Aplicaciones -> Campus Ingeniería" y seleccionar el botón para tal efecto.

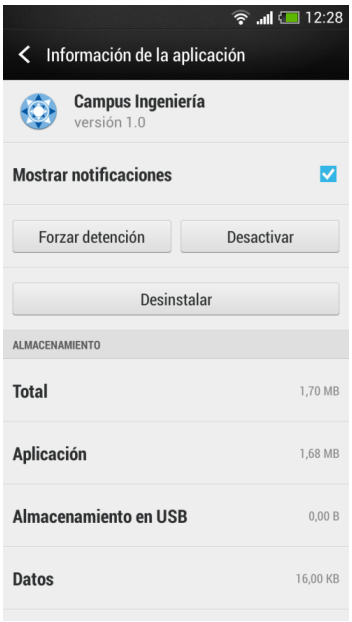

Ilustración 19 - Desinstalación de la aplicación

4.2. Funcionalidades generales. Casos de uso.

Una vez la aplicación haya terminado de instalarse, bastaría con irse al menú de aplicaciones y pulsar sobre el icono para iniciarla. A continuación se realizará un ejemplo de uso analizando qué es lo que se va a encontrar el usuario.

4.2.1. Pantalla principal

La pantalla principal es muy sencilla. Consta inicialmente de 3 botones que pasarán a ser 4 una vez nos identifiquemos. Este botón extra será utilizado cuando se quiera cerrar sesión con el usuario que esté actualmente activo. De esta forma un mismo dispositivo puede ser utilizado por distintos usuarios.

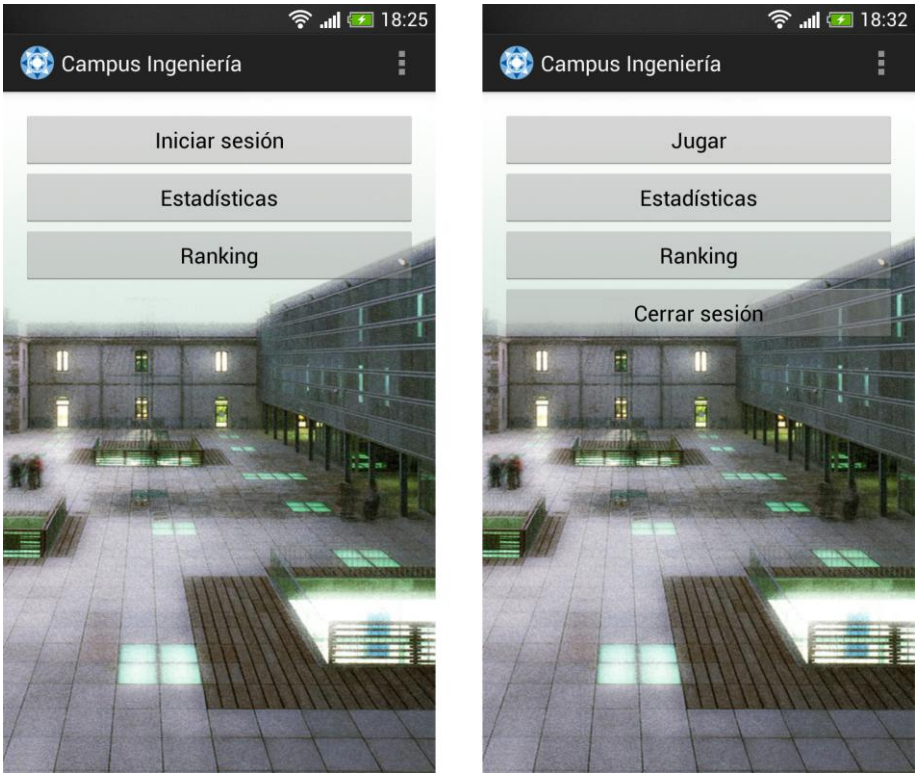

Ilustración 20 - Diferencias en la pantalla principal sin usuario registrado y con él

4.2.2. Iniciar sesión

El usuario que pretenda identificarse o crear un nuevo perfil deberá pulsar en el primer botón de la pantalla principal. Al hacerlo se mostrará una ventana donde se podrá ingresar tanto el nombre de usuario como la contraseña para aquellos que ya están registrados.

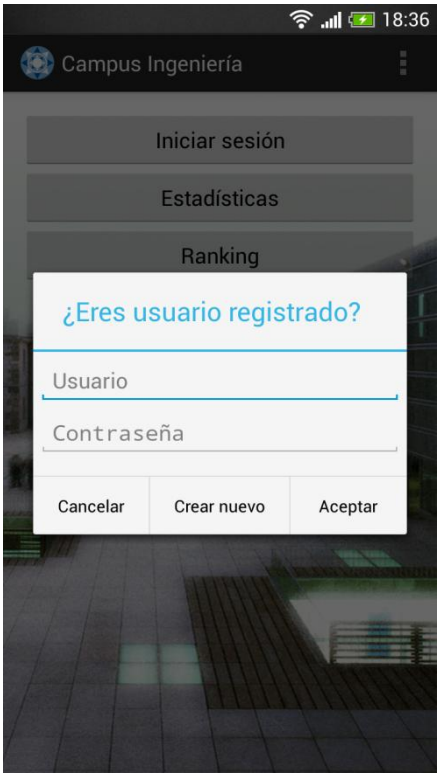

Ilustración 21 - Ventana para iniciar sesión

En caso de que la identificación se haya validado correctamente, ya se podrá comenzar el juego de preguntas y respuestas.

4.2.3. Crear un nuevo usuario

Si por el contrario lo que se desea es dar de alta un nuevo usuario habrá que pulsar sobre el botón central de la ventana anterior. Esto llevará al usuario a una nueva pantalla donde se le pedirán los datos necesarios para el correcto funcionamiento de la aplicación. Inicialmente se realizará una precarga de los centros y cursos que han confirmado su asistencia a la feria.

Una vez aquí se pueden dar 2 situaciones. Si el usuario a crear corresponde al rol de ponente se pedirán unos datos y si es al rol de visitante se ampliarán los datos requeridos, necesarios para realizar una filtración adecuada de las distintas clasificaciones que se mostrarán.

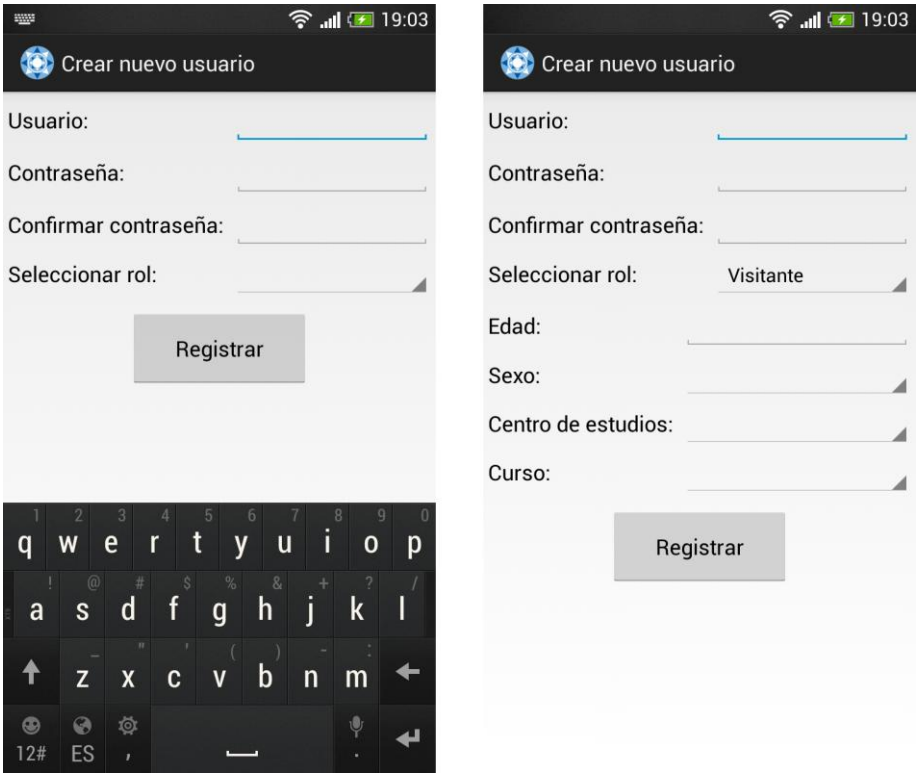

Ilustración 22 - Diferencias en los datos de registro entre Ponente y Visitante

Ésta diferenciación corresponde a la exigencia de recabar información sobre el grado de satisfacción con respecto a la feria en general, por parte de los ponentes de cada taller, sin la necesidad de que participen en el juego de preguntas y respuestas sobre los propios talleres. Aunque no tengan acceso a secciones como las estadísticas personales, obvio porque no habrá estadísticas que mostrar, si que podrán mostrar la clasificación general.

4.2.4. Comenzar a jugar: Selección de taller

Antes de comenzar a jugar se tiene que seleccionar el taller para el cual se van a contestar las preguntas. Este hecho se puede realizar de 2 formas diferentes:

- Eligiendo el taller de un desplegable. En él solo se incluirán los talleres para los que aun no se ha registrado ninguna respuesta.
- Realizando una captura del código *QR* situado en las cercanías del taller correspondiente.

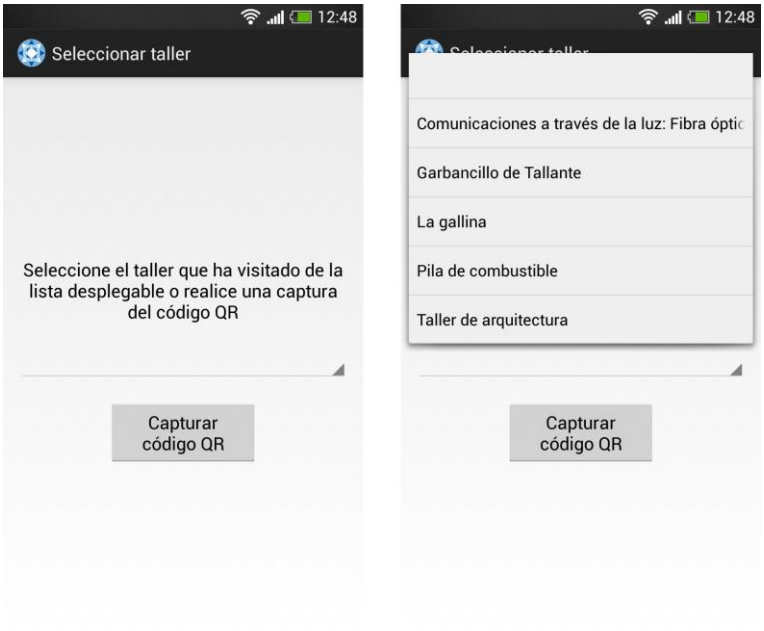

Ilustración 23 - Pantalla para seleccionar el taller visitado

Los datos se mandarán al servidor y se procederá a la descarga de las preguntas y respuestas.

4.2.5. Comenzar a jugar: Responder a las preguntas

En pantalla aparecerán todas las preguntas tipo test que hacen referencia al taller elegido. El usuario tendrá que marcar con la ayuda de las explicaciones dadas en la ponencia aquellas respuestas que considere correctas.

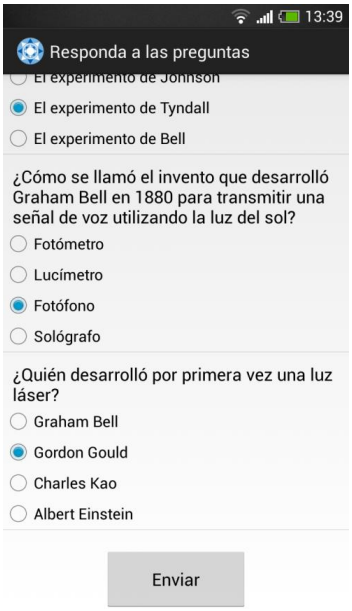

Ilustración 24 - Preguntas del taller "Comunicaciones a través de la luz: Fibra óptica"

4.2.6. Comenzar a jugar: Encuesta de calidad

Seguidamente se cargarán las preguntas para la encuesta de calidad. Al inicio aparecerá una ventana advirtiendo al usuario de que las respuestas para el juego únicamente serán tenidas en cuenta si participa en la encuesta de satisfacción.

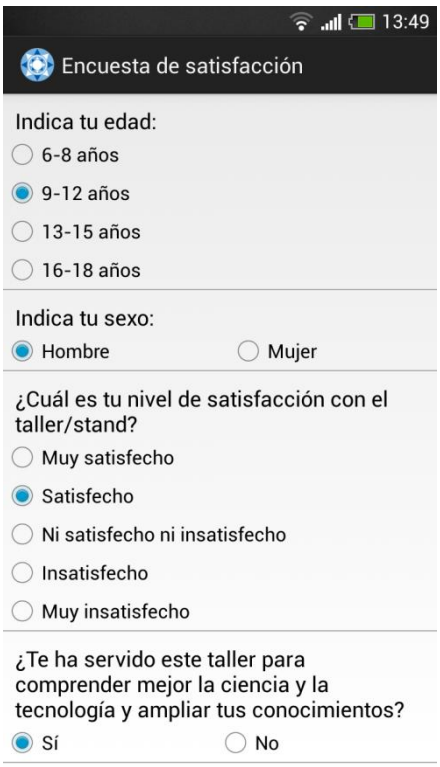

Ilustración 25 - Encuesta de satisfacción

Cuando responda a las preguntas que se le presenten, el servidor registrará las respuestas seleccionadas para los dos cuestionarios. Esta es la forma adoptada para evitar duplicidades y a la vez respetar la privacidad de los usuarios. Además, se está fomentando que los usuarios den su opinión respecto a su grado de satisfacción con el taller visitado ya que si desiste no se registrarán las respuestas relativas al juego.

4.2.7. Estadísticas personales

El usuario podrá comprobar su progreso en el juego entrando en la sección de las estadísticas personales. En ella se muestran los talleres para los que ya ha respondido al test, indicando el número de aciertos y fallos y el tanto por ciento de respuestas superadas.

Ilustración 26 - Estadísticas personales

4.2.8. Ranking

En caso de que ningún usuario se haya identificado en la aplicación, el *ranking* muestra los usuarios que mayor puntuación absoluta han obtenido así como una clasificación por centros donde se sumarían las puntuaciones de todos los usuarios registrados pertenecientes a un mismo centro.

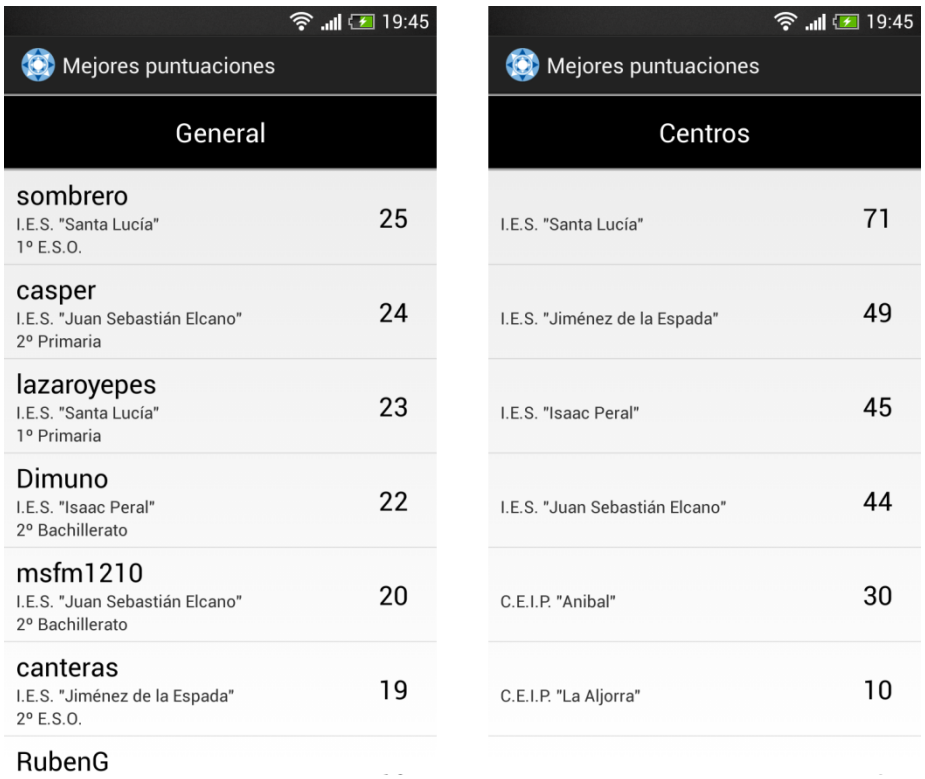

Ilustración 27 - Primeras posiciones de las clasificaciones generales

Si por el contrario existe un usuario registrado, además de las clasificaciones anteriores, también se mostrarán 3 clasificaciones más teniendo en cuenta el centro y

el curso con el que se ha dado de alta en la aplicación. La primera de ellas mostrará los usuarios con las mejores puntuaciones de su mismo centro. La segunda aquellos que estén realizando el mismo curso. Y la tercera filtrando por centro y curso, es decir, las mejores puntuaciones de su clase.

Ilustración 28 - Primeras posiciones de las clasificaciones personales

4.2.9. Cerrar la aplicación

Para salir de la aplicación correctamente es necesario darle al botón asignado en *Android OS* para retroceder estando en la pantalla principal. Aparecerá un cuadro de diálogo preguntando si se está seguro de ello tal y como muestra la siguiente captura.

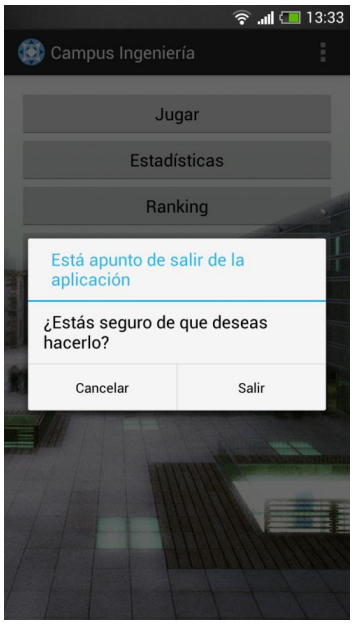

Ilustración 29 - Ventana para salir de la aplicación

Capítulo 5 – Conclusiones y líneas futuras

El proyecto se ha finalizado cumpliendo las expectativas iniciales a pesar de la cantidad de tecnologías diferentes a las que se han tenido que recurrir. Establecer la comunicación entre un cliente y un servidor siempre es un problema añadido y en este caso se ha terminado con éxito.

La multitud de herramientas y ayudas que se encuentran en internet siempre son bienvenidas pero no en todos los casos cubren las expectativas requeridas. Uno de los mayores problemas al que ha sido necesario hacer frente tiene que ver con la exigencia de hacer compatible la aplicación con versiones anteriores de *Android OS* y a su vez adaptarla a las facilidades de desarrollo que aporta el uso de *Fragments*. Así que se ha conseguido que funcione en la mayoría de dispositivos y que en futuras versiones se pueda explotar el uso de *Fragments* en *tablets* sin que sea necesaria una modificación profunda del código.

Las líneas futuras de desarrollo deben de ir en este camino ya que el uso de *tablets* o dispositivos móviles con pantallas por encima de 5” se está incrementando. También es cierto que el diseño de la aplicación debe de ser adaptado a las preferencias de sus usuarios, en este caso niños en edad escolar, dotándole a los elementos visuales de una estética más amigable y unos colores más vivos.

En cuanto al apartado de rendimiento, para futuras versiones sería interesante dotar a la aplicación de una base de datos local. El uso de proveedores de contenido –*Content Providers*- y *Loaders* que gestionen el acceso a estos datos evitaría la petición repetida al servidor de unos datos que quizás han sido requeridos con anterioridad. Todo esto disminuiría el tráfico en la red, la sobrecarga en el servidor y el tiempo de acceso a la información ya que se almacenaría temporalmente de forma local.

Capítulo 6 – Bibliografía

6.1. Referencias

http://www.gartner.com/newsroom/id/2573415

http://www.kantarworldpanel.com/es/Noticias/Noticias/Android-ya-est-en-9-de-cada-10-nuevos-smartphones

6.2. Bibliografía

http://www.wikipedia.org

http://developer.android.com/guide/index.html

http://www.stackoverflow.com

http://www.sgoliver.net

http://www.edu4android.com

http://www.androideity.com

http://www.grokkingandroid.com

http://androidresearch.wordpress.com

http://www.androidcurso.com