ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Desarrollo de Aplicaciones para la Plataforma Android. Un caso de estudio para el intercambio de libros.

AUTOR: Francisco Alarcón Barceló
DIRECTOR: D. Pedro Sánchez Palma
D. Juan Ángel Pastor Franco

Junio/2014

Autor	Francisco Alarcón Barceló
E-mail del Autor	Fran1_8@msn.com
Director	D. Pedro Sánchez Palma D. Juan Ángel Pastor Franco
E-mail del Director	pedro.sanchez@upct.es
Título de PFC	Desarrollo de Aplicaciones para la Plataforma Android. Un caso de estudio para el intercambio de libros

Resumen

El proyecto consiste en realizar un estudio de las posibilidades que ofrece la plataforma Android y su situación actual en el mercado del desarrollo de Apps y en el diseño e implementación de una aplicación para dispositivos con sistema operativo Android. Esta app permitirá realizar consultas a una base de datos MySQL creada por el alumno alojada en un servidor web Apache, las cuales harán posible que los usuarios introduzcan información de los libros que están dispuestos a donar, prestar o cambiar, y en la cual buscar libros que este necesite, permitiendo poner en contacto ya sea por email, o teléfono a los usuarios interesados que de otra manera no podrían conocerse.

Titulación	I.T.T, especialidad Telemática
Departamento	Lenguajes y Sistemas Informáticos
Fecha de Presentación	Junio 2014

Índice

FIGURAS Y TABLAS

INTRODUCCION

Desarrollo de Aplicaciones para la Plataforma Android. Un caso de estudio para el intercambio de libros.

Resumen:

El proyecto consiste en realizar un estudio de las posibilidades que ofrece la plataforma Android y su situación actual en el mercado del desarrollo de Apps y en el diseño e implementación de una aplicación para dispositivos con sistema operativo Android. Esta app permitirá realizar consultas a una base de datos MySQL creada por el alumno alojada en un servidor web Apache, las cuales harán posible que los usuarios introduzcan información de los libros que están dispuestos a donar, prestar o cambiar, y en la cual buscar libros que éste necesite, permitiendo poner en contacto, ya sea por email o teléfono, a los usuarios interesados, que de otra manera no podrían conocerse.

Planteamiento inicial

Debido a la crisis a nivel mundial muchas familias han visto reducido su nivel adquisitivo a casos insostenibles, recientes informes de la ONG Save the Children anuncian que un total de 2.826.549 niños vive en riesgo de pobreza o de exclusión social en España, cifra que se traduce en uno de cada tres niños.

El informe analiza cómo esta situación de pobreza o exclusión se materializa en la vida cotidiana de los niños y niñas convirtiéndose en un serio obstáculo para el disfrute de sus derechos esenciales reconocidos en la Convención sobre los derechos del niño de Naciones Unidas.

Actualmente el porcentaje del PIB destinado a políticas de protección social es del 25, 19%, 3,7 puntos por debajo de la media europea; cuando España es el octavo país de la Unión con mayor tasa de pobreza infantil, después de Bulgaria, Rumanía, Hungría, Letonia, Grecia, Italia e Irlanda.

La dramática situación económica de muchas familias en España les va a impedir este año asumir el gasto que implica, cada principio de curso escolar, la compra de todos los libros de texto que les exigen a los hijos en el colegio. En concreto, uno de cada tres niños en edad escolar tendrá serias dificultades para llevar los manuales requeridos, según han confirmado los presidentes de la Confederación Española de Asociación de Padres y Madres de Alumnos (CEAPA) y de la Asociación Giner de los Ríos, Jesús María Sánchez y José Luis Pazos, respectivamente.

También ha subrayado que el precio de los libros ha subido este año una media del 2,39% a lo que se une el **aumento del IVA del 4 al 21%** en el caso del material escolar, de uso imprescindible en las aulas. Las asociaciones denuncian la práctica desaparición de las ayudas para libros y comedor.
Así nace la idea de esta app sumándose a otras alternativas de toda la vida como los mercadillos de libros o los préstamos entre las propias familias que, en la mayoría de los casos, no es suficiente.

Combinando el uso de las tecnologías y lenguajes aprendidos durante la carrera y otras adquiridas por cuenta propia como la programación Android, se creara una aplicación compatible con dispositivos con sistema operativo Android, dicha aplicación ofrecerá a los usuarios una manera de darse a conocer y ponerse en contacto.

Objetivos:

- Estudio de la plataforma Android, de las posibilidades que ofrece la misma y su situación actual en el mercado del desarrollo de Apps.
- Creación de una base de datos.
- Diseño e implementación de aplicación Android que permita acceder a los datos servidos en una base de datos MySQL a través de un servidor web.
- Realización de pruebas de funcionamiento y depuración de posibles fallos.

Casos de uso:

- Prestar o donar: Querer contribuir a la comunidad o con los más desfavorecidos prestando o donando los libros que ya han sido utilizados por sus hijos, ayudando así a aquellos que no pueden o les afectaría mucho en la economía la compra de los libros.

- En el caso de necesitar los libros también se pueden pedir a la espera de que alguien nos los pueda prestar o dejar.

- Intercambiar: Cambiar nuestros libros usados por otros que necesitemos permitiéndonos un ahorro considerable en el gasto de libros anual.

- En el caso de no querer o no poder realizar el trámite en persona se podrían dejar los libros en su centro de enseñanza.

HISTORIA DE ANDROID

Todo comenzó con la creación de Android Inc. Google compro la empresa en Agosto de 2005, fecha en la cual Android Inc. ya contaba con 22 meses de vida. A Partir de entonces empiezan a correr rumores acerca de que google planeaba construir su propio dispositivo móvil libre y gratuito, obviamente no fue cierto, pero al final Android resulto ser algo muy interesante y revolucionario; un sistema operativo para móviles de código abierto propulsado por google que, el 5 de Noviembre de 2007, se produjo el anuncio oficial de que Android llegó a los medios.

VERSIONES DE ANDROID

Veamos por encima algunas versiones más relevantes de Android:

Android 1.0: Apple pie

Lanzado el 22 de octubre de 2008, el HTC Dream también conocido por entonces como Google Phone fue el primer dispositivo en incorporar el sistema operativo de Google.
Este incluyó la primera versión de la Android Market; un Navegador Web, soporte para mensajes de texto SMS y MMS, y una aplicación para tomar fotos que no contaba con los ajustes de blancos y resolución.

Android 1.5: cupcake 

Con la introducción de Android 1.5, el 30 de abril de 2009, empezamos a oír el nombre de Cupcake en referencia a la primera actualización importante del sistema operativo de Google.
Esta actualización le dio un poco más pulido a Android en algunas áreas, pero sus principales características fueron la introducción del teclado virtual en la pantalla y la posibilidad de insertar widgets.
Además, se incluyeron otras funciones bastante demandadas por los usuarios como copiar y pegar en el navegador, la grabación de vídeo y reproducción en formatos MPEG-4 y 3GP, la capacidad de subir videos a YouTube directamente, transiciones animadas entre las pantallas, la opción de auto-rotación, auto-sincronización y soporte para Bluetooth A2DP y AVRCP.

Android 1.6: Donut 

En septiembre de 2009 apareció Android 1.6 Donut con mejoras en las búsquedas, compatibilidad con pantallas de alta y baja densidad, indicador de uso de batería y la posibilidad de poder realizar VPN (Red Privada Virtual), que consiste en extender una red local sobre la red pública para realizar conexiones privadas.

Android 2.0: Eclair 

Android 2.0 Eclair incluía varias nuevas características y aplicaciones precargadas que requerían de una nueva generación de móviles, compatibilidad con la funcionalidad multitáctil, teclas virtuales, gestión de cuentas centralizada; además, sorprendió con su integración social, permitiendo sincronizar los contactos de Facebook y más tarde los de twitter, lo que permitió tener todos los contactos de todas las redes sociales en un solo lugar.

Android 2.2: Froyo

Lanzada el 20 de mayo de 2010, fue una de las actualizaciones que consagró al S.O como competencia del IOS 4 de Apple, dotando a sus terminales con un notable incremento de la velocidad de todo el sistema, tanto en la navegación de internet como en sus aplicaciones.

Froyo incorpora el motor de Java V8 y ofrece a los usuarios un aumento de velocidad gracias al compilador JIT, que permite iniciar las solicitudes más rápido y mejorar el rendimiento general del sistema. A su vez, Android 2.2 incluye la posibilidad de hacer tretheing, es decir, compartir la conexión 3G a través del wifi del teléfono con otros dispositivos y se consigue el soporte para Adobe Flash.

Android 2.3: Gingerbread

El 6 de diciembre de 2010 Google presentó de forma oficial Android 2.3 Gingerbread. Gingerbread incorporó una gran cantidad de novedades a nivel estético, con una renovada interfaz de usuario incluyendo incrementos de velocidad y simpleza. También se preparó para la llegada de los Smartphones de doble núcleo, al cambiar al sistema de archivos EXT4 y de pantallas más grandes, con el soporte para resoluciones WXGA y mayores.

Android 3.0: Honeycomb

El 22 de febrero de 2011 Google comenzó a desdoblar el sistema operativo con la actualización de Android 3.0 Honeycomb y su correspondiente SDK, algo que tendría poca vida debido al alto costo que supone mantener dos plataformas separadas.

Como principales características, una nueva interfaz, una barra de sistema para estatus global y notificaciones, pantallas principales personalizables y aplicaciones recientes para ver fácilmente la multitarea entre otras.

Android 4.0: Ice Cream Sándwich

La llegada de Android 4.0 Ice Cream Sándwich el 19 de octubre de 2011 significó un importante paso en la evolución de Android, que no solo vio renovada casi por completo su interfaz de usuario con el nuevo diseño Holo, sino que también volvió a integrar el sistema operativo en sus versiones para Tablets y Smartphones.

Android 4.1: Jelly Bean

Jelly Bean aún resuena como la última actualización importante del sistema operativo de Google, que dicho sea de paso, fue presentada el 27 de junio de 2012, y llegó al mercado el 13 de julio con el Nexos 7, el primer Tablet de Google.

El objetivo primordial de Android Jelly Bean fue mejorar la estabilidad, funcionalidad y rendimiento de la interfaz de usuario, para lo cual, se implementó el núcleo de Linux 3.0.31 y una serie de mejoras en, lo que se llamó, Project Butter, que permitió aumentar hasta 60 FPS las transiciones en la interfaz de usuario, dando una experiencia realmente fluida.

Android 4.4: KitKat

Finalmente, tras varias versiones sin cambios grandes, Google ha decidido dar un paso adelante con el desarrollo de Android. Lo suficiente como para que su próxima versión vaya a recibir un nuevo nombre de postre. Sí, el aperitivo de chocolate y galleta que hace la empresa alimentaria Nestlé, que no necesita más presentación. Nestlé va a hacer una campaña muy agresiva. Varios millones de tabletas van a salir con la imagen de Android, además de sortear miles de Nexus 7 y tarjetas regalo para Google Play, en lo que puede ser uno de los mayores acuerdos de márketing conjunto que podamos ver en mucho tiempo.

En cuanto al apartado técnico, Android ha simplificado sus procesos para reducir el uso de memoria, por lo cual, la plataforma operativa se vuelve más eficiente y rápida. Para ello ha ejecutado diferentes cambios y ajustes en el código del sistema, llamado Proyecto *Svelte.*

Es de hacer notar que esta mejora es únicamente para los nuevos dispositivos, y no para los ya existentes. Así que los fabricantes de celulares podrán tener la última versión de Android también en sus teléfonos de gama baja.

Android 5.0: Key Lime Pie, el futuro de Android. 

Poco se sabe de esta nueva versión de Android, la cual aún no tiene fecha de salida, pero ya se conocen algunas características de las que va a disponer.
Gestión del rendimiento del equipo, sistema de perfiles o administrador de estados, destinado a optimizar estos dos parámetros.
Soporte para múltiples dispositivos, por ejemplo, que si estamos viendo una película en el Smartphone y queremos seguir viéndola en la tableta, podamos hacerlo en el mismo punto donde la dejamos y no tener que empezar de nuevo o buscar donde nos quedamos.
Nuevos métodos de escritura, una de las características de los Smartphones con pantalla táctil es la posibilidad de introducir textos de distintas maneras: con el teclado virtual, con trazos, con la voz.
Video chat, al igual que las redes sociales, es necesario una mayor integración con el equipo (tipo Skype pero integrada en Android), la mejor forma de aprovechar la doble cámara que llevan ya casi todos los terminales inteligentes.
Una alternativa que podría incluir Android 5.0 es que apareciera la opción de "nunca actualizar" con relación a las aplicaciones. De esta manera, si el usuario no lo desea, no se producirían actualizaciones "accidentales".

Android en la actualidad

Aplicaciones Android: un caso de éxito

La plataforma Android va ganando terreno y lo hace bastante rápido.
Sin duda, el crecimiento exponencial de Android ha ocurrido, en gran medida, por sus cuestiones comerciales más que de rendimiento o popularidad mediática. Y es que el hecho de que cualquier fabricante pueda tomar Android como Sistema Operativo base para sus equipos, convirtiéndolos así en Smartphones, ha llevado a una verdadera revolución en lo que a dispositivos móviles se refiere, con distintos modelos y equipos variados inundando los mercados internacionales cada semana.

Apple creó la primera plataforma con aplicaciones para descargar en el Smartphone cuando lanzó el iPhone y la App Store. Desde entonces, Google ha mantenido una lucha con Apple desde el principio, tratando de superar el dominio que tenía la compañía de Cupertino, y poco a poco lo ha ido consiguiendo. Ahora, Android ya tiene mayor tráfico publicitario que iOS. Concretamente, el 42,83% del tráfico publicitario total es de usuarios que utilizan Android. IOS queda relegado a una segunda posición con un 38,17%.

Figura 1: trafico publicitario de los SO

Cada día hay más dispositivos, y no sólo eso, sino que cada día hay más fabricantes dispuestos a trabajar con su sistema operativo.
Es la plataforma en la que más fabricantes confían, y en la que más hardware encontramos disponible (y aumentando cada día que pasa…).
El hecho de que cada vez la cuota de mercado que Google le quita a RIM o Nokia sea mayor, hace que el hecho de programar para Android sea sinónimo de programar para un mayor porcentaje del mercado. Y ese hecho supone, además, que muchos desarrolladores hacen un esfuerzo extra para adaptarse a las múltiples configuraciones de hardware disponibles.

Modelos de negocio para las apps

1. La app gratuita: No todo el mundo sabe o puede pagar apps de pago, por lo que la cantidad de descargas de este tipo de apps es muy alta. Además, al ser gratuita permite al usuario darle una oportunidad aunque, a priori, no esté interesado.
La rentabilidad es nula, pero es una forma interesante de darse a conocer a los usuarios para la creación de otras apps con otro modelo de negocio.

2. La app gratuita con anuncios: casi igual a la app gratuita, pero añadiendo banners de publicidad. A priori, puede ser la solución más rentable de todas, aunque muchos usuarios no lo vean con buenos ojos. Inconscientes del trabajo que lleva detrás y a la rentabilidad que debe de tener su creador.

¿Cómo funciona este modelo publicitario?
El programador utiliza un intermediario como, por ejemplo, Google AdMob, el cual sirve los anuncios en las apps haciendo posible la publicidad y cobrando un pequeño porcentaje a cambio cuando se efectúa la acción.

La acción puede ser:

- **Coste por cada mil impresiones:** cuando 1000 usuarios visualizan el anuncio.
- **Coste por click:** cada vez que un usuario hace click en el anuncio.
- **Coste por lead:** el usuario hace click, rellena un formulario o se registra. Los ingresos son mayores, pero es más complicado que se lleve a cabo.
- **Coste por acción:** Se lleva a cabo cuando el usuario realiza una acción determinada, como hacer una compra o realizar una descarga. Los ingresos suelen ser bastante altos, normalmente una comisión por cada producto vendido.

3. la app de pago: el usuario sólo puede instalarla si realiza un pago. Tanto en esta modalidad como en la del modelo publicitario, Google se lleva el 30% de los ingresos, y el resto el programador. Estas apps necesitan mucha suerte para triunfar, pues tienen un alcance bastante bajo y necesitan mucha promoción.

4. freemium:combinación del metodo gratuito con el de pago. En el que se ofrece una versión gratuita con ciertas limitaciones o desventajas, como los anuncios, los cuales desaparecen si compras la versión de pago existente.

5. donación: el usuario puede realizar una donación al programador si a quedado muy satisfecho o le ha supuesto un ahorro de dinero, mostrando su gratitud al programador por el trabajo realizado. También pueden ser apps únicamente destinadas a las donaciones de organizaciones benéficas.

6. Demostración: es aquella que muestra un ejemplo del producto como un periodo de prueba o el acceso a los primeros niveles de un juego. Y luego invita al usuario a comprar la versión final si le ha gustado.

7. Añadidos in-app: Nos permite la compra de características o funcionalidades dentro de una aplicación, así como ventajas sobre otros usuarios,paquetes de niveles, etc.

Arquitectura Android

Android es una plataforma para dispositivos móviles que contiene una pila de software, donde se incluye un sistema operativo, **middleware** y aplicaciones básicas para el usuario.

A continuación, daré una versión global de las capas, cada una de estas capas utiliza servicios ofrecidos por las anteriores y ofrece, a su vez, los suyos propios a las capas de niveles superiores.

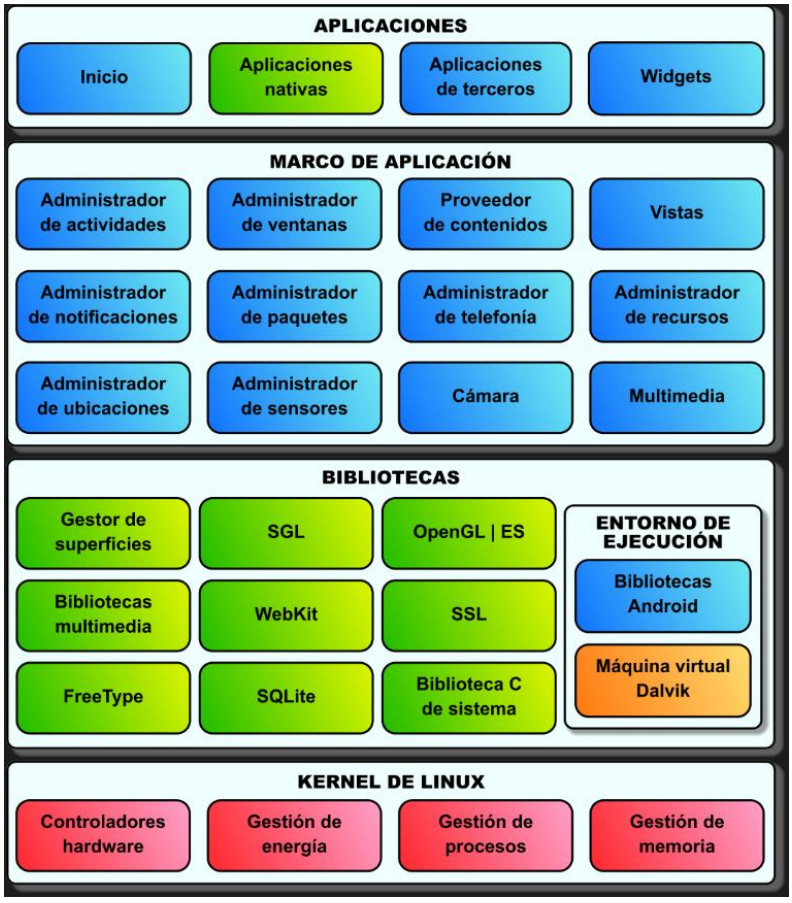


Figura 2: Arquitectura Android

- **Aplicaciones:** En esta última capa se incluyen las aplicaciones tanto nativas (programadas en C o C++) como administradas (programadas en java) , las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente .Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.En esta capa encontramos también la aplicación principal del sistema: Inicio (Home) o lanzador (launcher), que permite ejecutar otras aplicaciones mediante una lista, mostrando diferentes escritorios donde se pueden colocar accesos directos a aplicaciones y widgets.

- **Framework de Aplicaciones:** Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo "framework", representado por este nivel.

- **Tiempo de ejecución de Android:** Al mismo nivel que las librerías de Android se sitúa el entorno de ejecución. Éste lo constituyen las Core Libraries, que son librerías con multitud de clases Java y la máquina vistual Dalvik.

  Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute. La ventaja de esto es que las aplicaciones se compilan una única vez y, de esta forma, estarán listas para distribuirse con la total garantía de que podrán ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera la aplicación.

- **Núcleo Linux**: Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro de este kernel de Linux embebido en el propio Android.
  El kernel también se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación (*networking*), etc.

Estructura de una aplicación

A la hora de desarrollar una aplicación para Android tendremos múltiples componentes independientes comunicados entre sí a través del sistema operativo, los cuales, implementan una serie de métodos que son invocados por el sistema operativo cada vez que se cumplan las condiciones necesarias para llamarlos. Por lo tanto, las aplicaciones en Android se comportan de forma asíncrona y es el SO el que se encarga de gestionar estas llamadas según las peticiones del usuario.

Componentes que permite declarar el API de Android:

Servicios

Un servicio es una tarea del sistema que se ejecuta en segundo plano o Background sin que el usuario esté interactuando con ella, es decir, sin interfaz gráfica. Los Servicios son usados para realizar operaciones de larga duración, de manera que el usuario no tenga que esperar a que se procese. Un ejemplo de esto es un reproductor de música en el que, una vez que se ha iniciado el usuario, puede realizar otras acciones diferentes.

Actividades

Un Activity es el componente de Android que proporciona una ventana en una aplicación con la que los usuarios pueden interactuar para realizar una acción como, buscar un contacto, realizar una foto, enviar una foto etc.
Entre las diversas cantidades de actividades que pueden formar una aplicación, una de ellas se marcará como la actividad principal o “Main”, que inicia la aplicación.
Las actividades pueden iniciar otras actividades con el objetivo de componer la aplicación.

En el siguiente grafico mostraremos el ciclo de vida de una actividad en Android en la que onCreate () y onDestroy () representan el principio y el fin de la actividad:

OnStart () y onStop () representan la parte visible de la aplicación, es decir, la actividad será visible para el usuario aunque no tenga el foco de acción.

onResume() y onPause() son las que delimitan la parte útil del ciclo de vida, es decir, además de ser visible tendrá el foco de acción y el usuario podrá interactuar con ella.

El proceso que mantiene a la actividad sólo puede ser eliminado cuando se encuentra en onPause () o en onStop (), cuando no tenga el foco de aplicación. Aunque el proceso sea eliminado, puede ser restaurado gracias a una copia, volviendo así a estar activo como si no hubiera sido eliminado. Además, la activity puede haber estado en un segundo plano invisible y puede ser despertada pasando por el estado onRestart ().

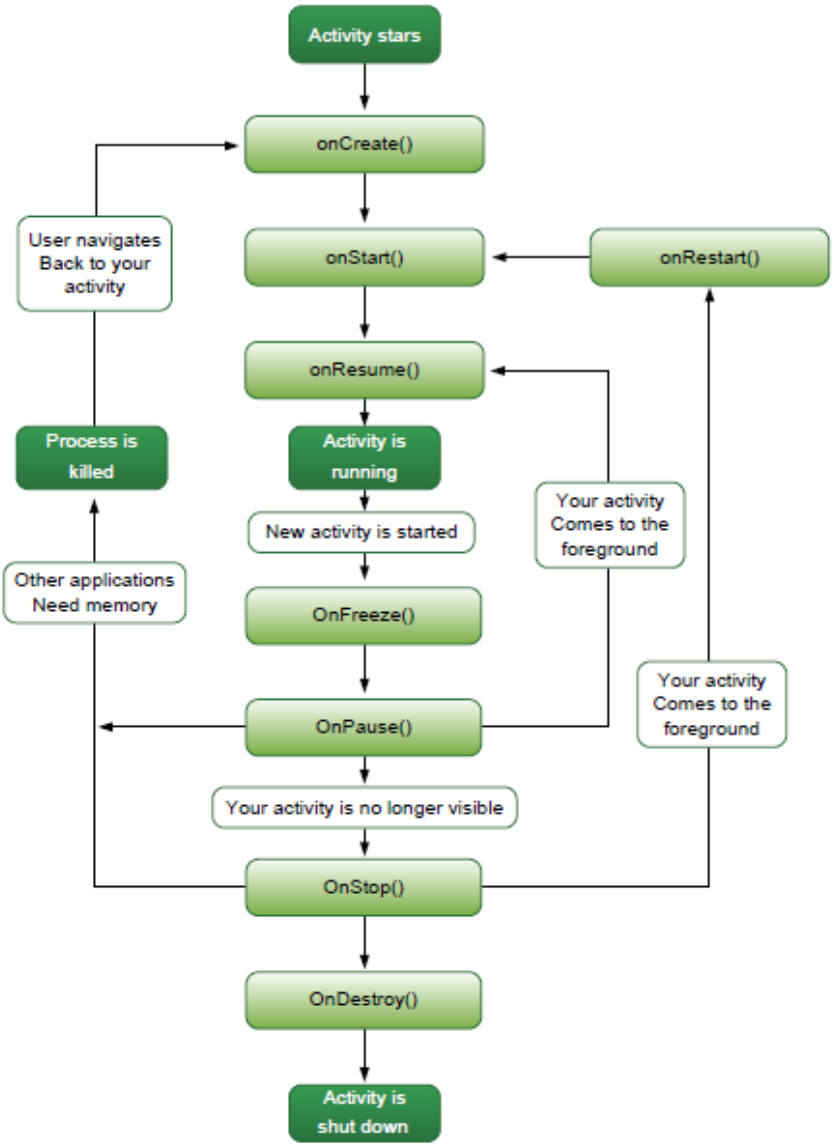


Figura 3: Ciclo de vida de una actividad en Android

Broadcast Receivers

Es el componente que está destinado a recibir y responder ante eventos globales generados por el sistema, como: “batería baja “, “sms recibido”, “tarjeta SD insertada” o por otras aplicaciones. Estos componentes son representados por clases Java, lo que da a entender que cuando la aplicación se ejecuta se crean instancias de éstos, pudiéndose crear más de una instancia del mismo componente.

Intent

Es un mecanismo para describir una acción específica, declarar algo que necesitamos. Son fragmentos de información que describen la acción o servicio deseado. Una clase que permite especificar una activity a ejecutar, llamando a uno de los métodos de la clase Activity con esos intent de parámetro.

Un intent es el elemento básico de comunicación entre los distintos componentes Android que hemos descrito anteriormente. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones. Mediante un intent se puede mostrar una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc.

Content Provider

Es la forma de compartir datos entre aplicaciones que se han definido en Android, sin tener que mostrar detalles de su almacenamiento interno, su estructura o su implementación. De esta manera la aplicación puede tener acceso a los datos de otras aplicaciones que se hayan definido usando el content provider. También puede almacenar datos en los ficheros del sistema, en la base de datos SQLite de la aplicación, una página Web o en cualquier otro lugar de almacenamiento.

Entorno de desarrollo y componentes

Durante el desarrollo del proyecto he intentado usar el software y las herramientas más adecuadas para realizarlo, las cuales detallaré a continuación.

El SDK de Android (Software Development Kit)

El SDK de Android incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales.

Las plataformas de desarrollo soportadas incluyen Linux, Mac OS X 10.4.9 o posterior, y Windows XP o posterior. La plataforma integral de desarrollo (IDE, Integrated Development Environment) soportada oficialmente es Eclipse, junto con el complemento ADT (Android Development Tools plugin). Aunque también puede utilizarse un editor de texto para escribir ficheros Java y Xml, y utilizar comandos en un terminal (se necesitan los paquetes JDK, Java Development Kit y Apache Ant) para crear y depurar aplicaciones. Además, pueden controlarse dispositivos Android que estén conectados.

Las Actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad.

Eclipse Versión: 4.3.1

Eclipse 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 , 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 está siendo 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.

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. Este mecanismo de módulos es una plataforma ligera para componentes de software.

Ventajas en la utilización de Eclipse:

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

2- Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente permite a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python, los cuales permiten a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de gestión de base de datos.

3- La arquitectura plug-in permite escribir cualquier extensión deseada en el ambiente, como sería Gestión de la configuración. Se provee soporte para Java y CVS en el SDK de Eclipse. Y no tiene por qué ser usado únicamente para soportar otros Lenguajes de programación.

En cuanto a la utilización de eclipse para la creación de aplicaciones clientes se puede decir que:

1- Eclipse provee al programador con Frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de Software, Aplicaciones web, etc. Por ejemplo, GEF (Graphic Editing Framework - Framework para la edición gráfica) es un plug-in de Eclipse para el desarrollo de editores visuales, que pueden ir desde procesadores de texto wysiwyg hasta editores de diagramas UML, interfaces gráficas para el usuario (GUI), etc. Dado que los editores realizados con GEF "viven" dentro de Eclipse, además de poder ser usados conjuntamente con otros plugins, hacen uso de su interfaz gráfica personalizable y profesional.

2- El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código.

EasyPHP 14.1

"EasyPHP" es un programa que instala en un solo paso el servidor Apache, junto con el modulo para programación en PHP y la base de datos MySQL.

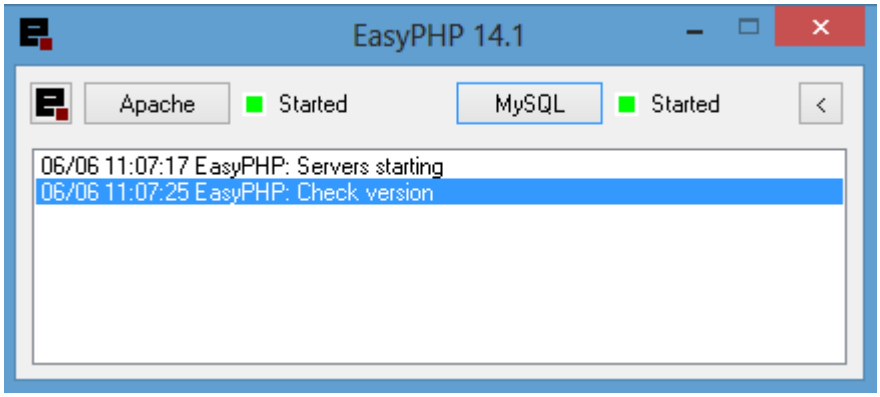


Figura 4: pantalla principal EasyPHP

Debemos comprobar que todo está correcto. De un modo visual, podemos observar dos luces verdes correspondientes al servidor web y a la base de datos, que deben de estar en verde.

Configuración básica

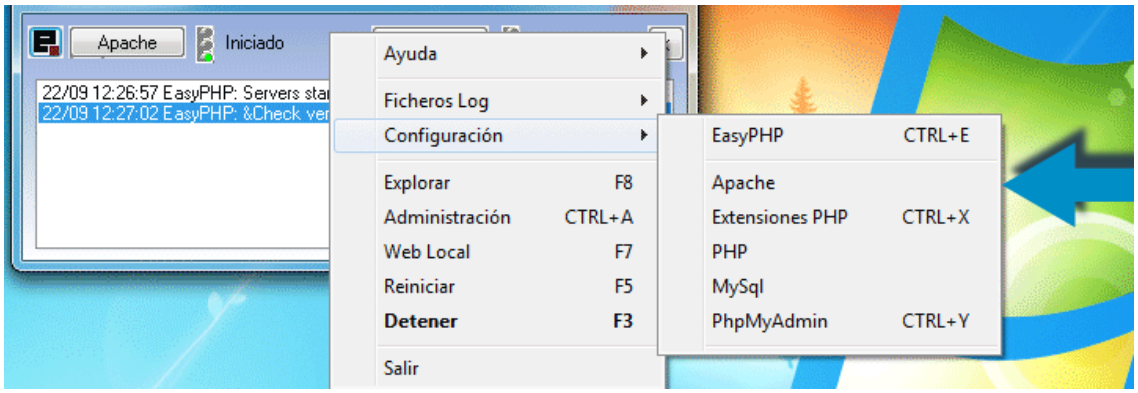


Figura 5: Configuración básica EasyPHP

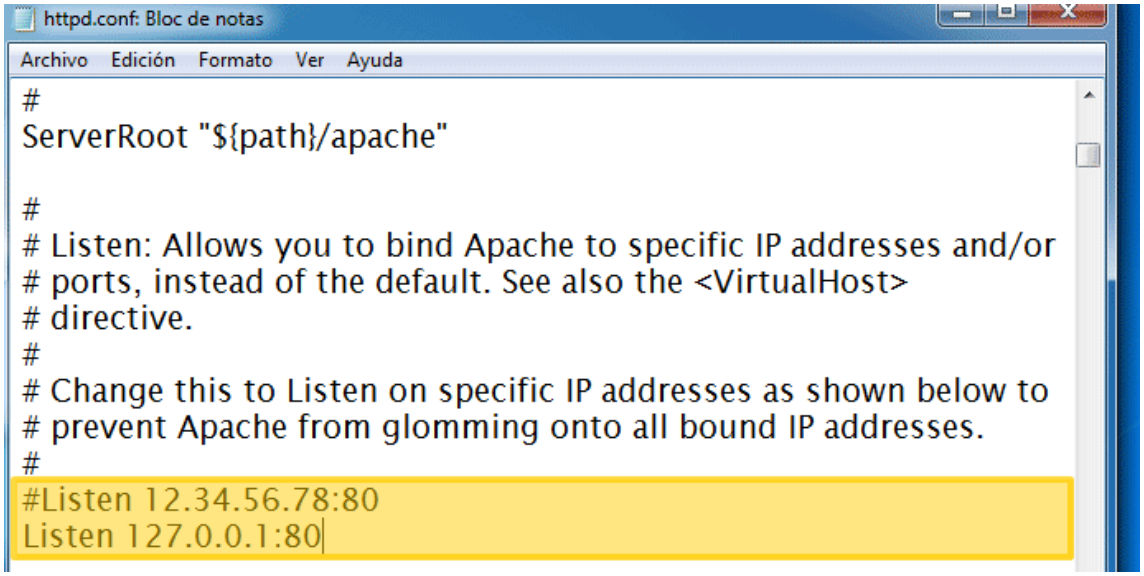


Figura 6: listen loopback

La instrucción "listen" le indica al servidor que escuche todas las llamadas que le entran por la dirección IP 127.0.0.1 (loopback), a la cual conoceremos como "localhost".El puerto 80 es por defecto el puerto de comunicaciones para http://

En informática, en el contexto de redes TCP/IP,"localhost" es un nombre reservado que tienen todas las computadoras, router o dispositivos independientemente de que disponga o no de una tarjeta de red Ethernet. El nombre "localhost" es traducido como la dirección IP de loopback 127.0.0.1.

El Servidor Web Apache

Un Servidor Web es un programa que se ejecuta continuamente en un servidor, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. El servidor web se encarga de contestar a esta petición de forma adecuada entregando como resultado una página web o información relativa a los comandos solicitados.

Gracias a los avances en conectividad y la gran disponibilidad de banda ancha, hoy en día es muy común establecer los servidores web dentro de la propia empresa, sin tener que recurrir a caros alojamientos en proveedores externos. Esto es posible gracias a Apache, uno de los mejores y el más utilizado entre los servidores web gracias a su gran estabilidad y confiabilidad.

Entre las ventajas que presenta un servidor como Apache se encuentran las siguientes:

- Es personalizable, la arquitectura modular de Apache permite construir un servidor hecho a la medida. Además permite la implementación de los últimos y más nuevos protocolos.
- En cuanto a la administración, los archivos de configuración de Apache están en ASCII, por lo que tiene un formato simple, y pueden ser editados tan solo con un editor de texto. El servidor puede ser administrado vía línea de comandos, lo que hace la administración remota muy conveniente.
- Por otra parte se trata de un servidor muy eficiente, mucho esfuerzo se ha puesto en optimizar el rendimiento de código C de Apache. Como resultado, éste corre rápido y consume menos recursos de sistema en comparación con otros servidores. Ademas, Apache corre en una amplia variedad de sistemas operativos, incluyendo varias versiones de UNIX, Windows o MacOS (Sobre Power PC).

PHP

PHP es un lenguaje de programación del lado del servidor , es decir ,la ejecución la hace el servidor y no el cliente, de este modo lo que le llega al cliente cuando accede a una página creada en código HTML es código HTML , el cual lo interpreta nuestro navegador.
PHP es una alternativa a las tecnologías de Microsoft ASP y ASP.NET(que utiliza C# y Visual Basic .NET como lenguajes), a ColdFusion de la empresa Adobe, a JSP/Java, CGI/Perl y a Node.js/JavaScript.
Aunque su creación y desarrollo se da en el ámbito de los sistemas libres, bajo la licencia GNU, existe además un entorno de desarrollo integrado comercial llamado Zend Studio.

PHP es comúnmente utilizado para:

- Acceder a bases de datos.
- Procesar datos de formularios.
- Enviar correos desde la interfaz web.
- Manipular información como textos, imágenes, campos o formularios de forma dinámica.
- Crear sitios web y aplicaciones dinámicas.
- Crear foros, portales de noticias, tablones, etc.

Base de datos MySQL

Una base de datos es un conjunto de datos que están organizados para un uso determinado. Los programas que permiten gestionar estos datos se llaman Sistemas Gestores de Bases de Datos. Estos tratan la información utilizando el modelo de gestión de bases de datos relacional, en el que los datos se organizan en tablas.
Las tablas almacenan información sobre un tema, como pueden ser los clientes de una empresa, o los pedidos realizados por cada uno de ellos.
Para el presente proyecto he empleado la base de datos MySQL porque es una de las más populares, pues es de código abierto y ofrece soporte para más de 20 plataformas como Linux, Windows, NetWare, etc.
Pueden utilizarse infinidad de lenguajes de programación para acceder a MySQL. Existen interfaces para C, C++, PHP, C#, y Java, entre otras.
Es multihilo, por lo que es capaz de trabajar con más de un procesador simultáneamente. Ofrece un alto rendimiento, fiabilidad y facilidad de uso. Proporcionando además una amplia gama de herramientas de bases de datos, por lo que es la mejor opción para las muchas empresas y para mí en este caso.

Notepad++ v6.6.1

Es un editor básico para diversos lenguajes de programación, disponible en varios idiomas, incluido el español. Es de resaltar su sistema de búsqueda y coincidencia de texto con capacidad de filtros de expresiones regulares.
Con este pequeño y poderoso “blog de notas “podemos abrir y editar archivos HTML, PHP y otros lenguajes fácilmente.
El programa reconoce el tipo de formato con el que estoy trabajando y modifica sus características acorde a él. Así un archivo de tipo PHP, HTML o CSS tendrá resaltadas variables y valores con distintos colores, para facilitar la edición.

Tareas en segundo plano Android y AsyncTask

Introducción

A veces tenemos aplicaciones que tardan mucho tiempo en realizar una tarea pesada y queremos que esta no bloquee la aplicación mientras se está ejecutando. Si estamos realizando aplicaciones complejas como tener que realizar una conexión con el servidor o enviar y recibir datos, como es el caso de mi app, es obligatorio en las últimas versiones del sistema operativo Android, que esto se haga sin interrumpir el hilo principal, ya que no será permitido, produciéndose múltiples errores de código en la app.
El hilo principal del usuario es el principal en mi aplicación. Normalmente es el único que se utiliza en app sencillas, sin embargo, existen aplicaciones complejas en las que tenemos múltiples hilos que se ejecutan a la vez. Como por ejemplo, una en la que utilizamos el hilo principal para movernos por la interfaz de pantalla mientras que, en segundo plano, se están recibiendo datos de un servidor.

¿Cómo se soluciona y que elecciones tenemos?

Para resolver esto normalmente se hace uso de los Threads (hilos). Para ello, creamos un hilo adicional para nuestra tarea y continuamos con el hilo normal para el resto de la aplicación. El problema es que la interfaz gráfica (*UI*) de Android no permite llamadas desde otros hilos que no sea el suyo, por lo que si realizamos alguna actualización, como añadir información nuestra aplicación se cerrará.

Usando threads (hilos) de java.

Mediante el uso de hilos, se pueden crear aplicaciones que responden cuando se está trabajando con procesos que requieren mucho tiempo.
En Java se puede crear una clase que extiende de la clase Thread y anula el método public void run (), después se llama al método start () para ejecutar el proceso que consume más tiempo. Si la clase ya extiende de otra clase, se puede implementar la interfaz Runnable.
Cuando empieza la actividad en el método **onCreate ()** creamos un objeto de tipo **Thread** que está construido con un objeto **Runnable**.
El método de la clase **Runnable run ()** se ejecutará cuando llamemos al método **start ()** de la clase **Thread**. Desde aquí se pude llamar a otro método o varios métodos y operaciones que requieren mucho tiempo y que, de otra forma, harían que la aplicación no respondiese.
Normalmente, cuando nosotros trabajamos con hilos obtenemos unos resultados que queremos enseñar al usuario. Si intentamos actualizar la interfaz gráfica de usuario desde el **Thread** (no desde el **Thread** principal) la aplicación se bloqueará.

Los **Thereads** creados e inicializados seguirán funcionando incluso si el usuario sale de la aplicación. Se puede hacer un seguimiento de los **Threads** y decirles que se detengan, esto normalmente se hace con un valor booleano. Para estar seguros de que el **Thread** se detiene cuando el usuario sale de la aplicación, antes de llamar al método **start ()** el objeto **Thread**, lo estableceremos como un **Thread** demonio.

Utilizando la clase AsyncTask

Esta segunda solución es más que suficiente, y además es más sencilla y más limpia de implementar. Considero que ésta es mejor que las demás, utilizando la clase AsyncTask.
Uno de los problemas de los sistemas operativos Android es que no es posible realizar una conexión al servidor en una única tarea principal, por lo que es necesario crear tareas adicionales en el servidor para poder hacer esta conexión.
Debido a los problemas que me han aparecido a la hora de solventar esta cuestión, tuve que buscar una manera de realizar este tipo de conexión. Finalmente encontré que se pueden realizar estas tareas en segundo plano con la clase AsyncTask.

Una estructura típica de una clase de AsyncTask es la siguiente:

```
private class miTarea extends AsyncTask<x, y, z>

protected void onPreExecute(){

}
```

- **X:** Datos que pasaremos al comenzar la tarea.
- **Y:** Parámetros que necesitaremos para actualizar la *UI*.
- **Z:** Dato que devolveremos una vez terminada la tarea.

OnPreExecute (). Se ejecutará antes de la tarea en segundo plano. Se suele utilizar para preparar la ejecución de la tarea, inicializar la interfaz, etc.

EJEMPLO DE MI APP:

```
class CargaNoticia extends AsyncTask<String, String, String> {


}
```

Quiero crear una aplicación que descargue un archivo de internet. A la tarea le pasaré la dirección del fichero (*String*). Conforme vaya descargándolo irá mostrando un mensaje informándonos de qué está pasando (*String*), y una vez terminado devolverá una cadena de caracteres (*String*).

Si alguno de los parámetros no lo necesitamos, se puede sustituir por void.

```
protected Z doInBackground(X...x) {
}
```

Al crear una clase que extiende de AsyncTask tendremos que implementar un método abstracto (ya que AsyncTask es una clase abstracta), llamado **doInBackground(),** el cual contendrá el código de la tarea que queremos ejecutar en segundo plano. Podemos decir que es el más importante de la clase AsyncTask. Éste método será el encargado de realizar la tarea en segundo plano de la aplicación, en un hilo diferente al principal. Aquí la entrada es un conjunto de objetos x, donde, cómo podemos ver en la cabecera, son los que le entran a la extends de AsyncTask, devolviendo un objeto Z.

```
protected void onProgressUpdate(Y y){

}
```

OnProgressUpdate (). Se ejecutará cada vez que llamemos al método **publishProgress (y)** desde el método **doInBackground ()**. Normalmente, se utiliza para actualizar la interfaz de usuario cuando queremos mostrar algún progreso o información en pantalla principal, mostrando la operación que se está haciendo en el fondo.

```
protected void onPostExecute(Z z){

}
```

onPostExecute(). Se ejecutará cuando finalice nuestra tarea en segundo plano, es decir, cuando termine el método **doInBackground()**.
Este método es utilizado cuando, después de toda la operación que se hizo en background, lo manda llamar y le entra como parámetro de entrada, el parámetro de salida del método de doInBackground.

Entonces podemos decir que los valores que entran a la AsyncTask son X, que es el tipo de variable de entrada que tenemos para el proceso del fondo. La Y es el tipo de objetos que se van a introducir en el método **onProgressUpdate**, y la Z es el tipo de objetos que tendremos del resultado de las operaciones hechas en doIbBackground.

Para hacer llamar esta tarea lo que tenemos que hacer es:

```
miTarea tarea1 = new miTarea ();
```

```
tarea1.execute(x);
```

Donde x es la entrada del parámetro del tipo X.
Podemos, una vez que se esté ejecutando, saber el estatus con el siguiente método.

```
tarea1.getStatus();
```

resultados:

Running: Indicando que la tarea se está ejecutando.
Pending: Indicando que la tarea no se está ejecutando.
Finished: Indicando que onPostExecute(); ha terminado.

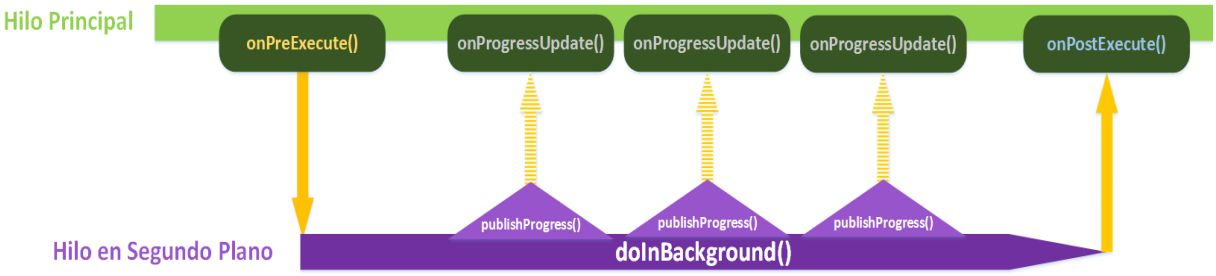


Figura 7 : proceso completo AsyncTask

JSON como metodo de intercambio de datos con el servidor.

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes esto se implementa como arreglos, vectores, listas o secuencias.

En JSON, se presentan de estas formas:

Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con {(llave de apertura) y termine con} (llave de cierre). Cada nombre es seguido por: (dos puntos) y los pares nombre/valor están separados por, (coma).

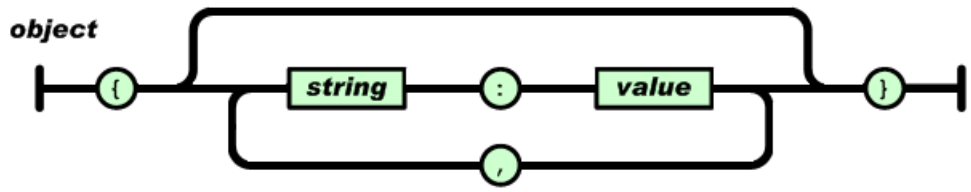


Figura 8: representación JSON objeto

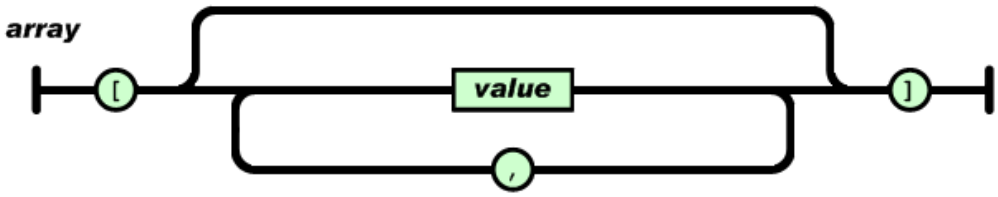


Figura 9: representación JSON array

En resumen JSON es una forma de codificar objetos, arrays o cualquier otra serie de datos en un string y, posteriormente, poder descodificarlo sin muchos problemas. Es especialmente útil para enviar información entre cliente y servidor de una forma muy sencilla, puesto que cada componente descodifica la información según le convenga, indistintamente del resto del sistema.

¿Porque utilizar esta opción y no utilizar otras opciones posibles como XML?

Durante un buen tiempo, XML ha sido la única opción para compartir datos libremente. No había otros formatos abiertos disponibles y XML fue considerado como la solución para todos los problemas de intercambio de datos. Este único formato podía manejar datos clásicos como números y texto, pero también podía manejar documentos, formatos, imágenes, audio, vídeo, y mucho más. Ahora que otras opciones están disponibles, XML puede ser una exageración en muchos casos.

Estabilidad

Con JSON, se está limitado a almacenar sólo datos clásicos como texto y números. Sin embargo, XML le permite almacenar cualquier tipo de datos que se le pueda ocurrir. La capacidad para extender los atributos de los datos almacenados en los ficheros XML es lo que le permite ser más flexible que JSON. Sin embargo, también lo hace más difícil de leer. Esto hace a XML más extensible, pero puede que no sea algo bueno. Esto depende del tipo de la información que trata de transferir. Los documentos requieren extensibilidad para gestionar imágenes, tablas, gráficos, y otros elementos de formato. Sin embargo, los datos clásicos no requieren esta extensibilidad y pueden beneficiarse de la simplicidad de JSON.

Mayor legibilidad

Los ficheros JSON son más restrictivos y, por lo tanto, ligeramente más legibles. Esto se debe a que el número de formatos de datos permitidos por JSON es mucho menor que XML. Además, la estructura de los datos está más estandarizada con los ficheros JSON debido al hecho de que existen menos opciones cuando se compara con el formato XML.

Integración completa con todos los formatos

Con XML es posible adjuntar cualquier fichero de cualquier formato. Por otro lado, JSON sólo soporta formatos de datos tradicionales. Esto significa que es posible incluir fotos, audio, vídeo, y otros ficheros dentro de un fichero XML. Mientras que esto puede parecer algo bueno al principio, también puede ser peligroso. Eso es porque podría incluir un fichero ejecutable que podría tener peligrosas consecuencias para la seguridad. La simplicidad de las estructuras de datos que JSON soporta hace imposible los ataques usando este formato.

Compartir datos tradicionales

JSON es la mejor herramienta para compartir datos. Esto se debe a que los datos están almacenados en vectores y registros, mientras que XML almacena los datos en árboles. Ambos tienen sus ventajas, pero las transferencias de datos son mucho más fáciles cuando los datos se almacenan en una estructura que está familiarizada a los lenguajes orientados a objetos. Esto hace que sea muy sencillo importar datos desde un fichero JSON a Perl, Ruby, JavaScript, Python, y otros muchos lenguajes. Para hacer lo mismo con XML, necesitaría primero transformar los datos antes de que puedan ser importados. Por este motivo, JSON es un formato de fichero superior para las APIs web.

La opción XML pasa por:

- Convertir el array en un XML.
- Pasarle el XML al servidor.
- Parsear el XML en el servidor.
- Convertir los datos en un array.

Además de ser un pesado sistema cuando cambian las especificaciones de datos, puedes llevarte la sorpresa de que el cliente ahora quiera los datos en un array multidimensional.

Compartir documentos

Cuando se quiera compartir documentos, XML es la herramienta adecuada para ello. Esto se debe a que permite incluir tipos de datos como imágenes, tablas, y gráficos. Además, XML ofrece opciones para transferir la estructura, o formato de los datos, junto con los verdaderos datos. JSON sólo ofrece opciones para la transferencia de los datos sin formato, y sólo

utiliza formatos de datos tradicionales. Esto hace a XML el formato superior para documentos.

XML y JSON comparación de análisis de datos

El cuadro comparativo muestra el tiempo más rápido de análisis de JSON para los datos más pequeños. Sin embargo, aumentando el tamaño de los datos XML se realiza un análisis más rápido. El tiempo de carga es más rápido con JSON y también el tamaño de los datos de JSON es menor.

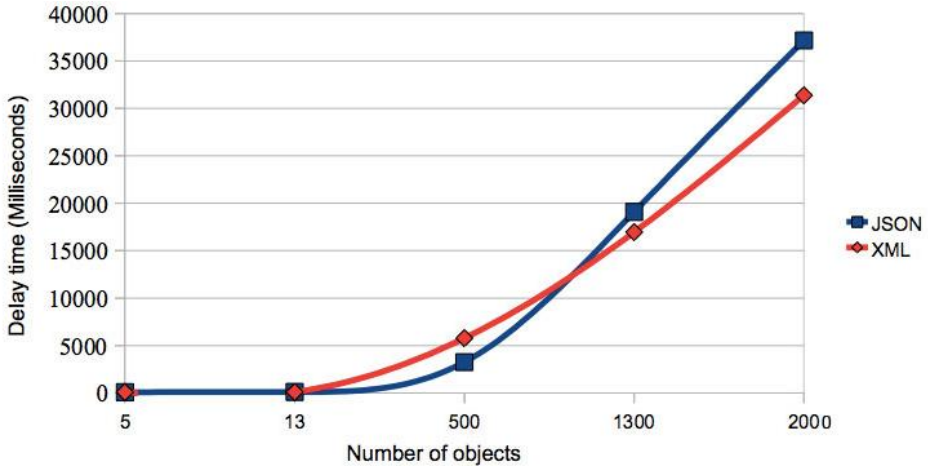
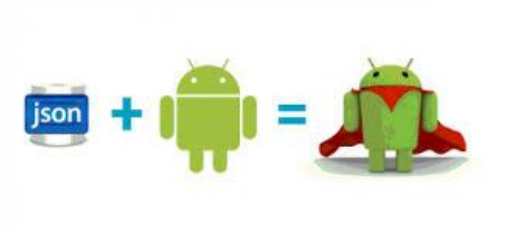


Figura 10: comparación XML vs JSON análisis de datos

Conclusión ¿Por qué JSON?

JSON es el mejor método para compartir datos tradicionales, ya que los datos están almacenados en vectores y en registros. Este tipo de datos es el que voy a usar en este proyecto, además de que con XML se deberían de transformar los datos.

La eficiencia en el uso de los recursos a la hora de desarrollar aplicaciones móviles es una piedra angular que no podemos obviar, sobre todo, en el caso del consumo de datos de internet, que deberemos optimizar lo máximo posible para ahorrar batería y aprovechar las tarifas de datos existentes actualmente. En este sentido JSON, gracias a su estructura, ahorra considerablemente respecto a XML.



Implementación de la aplicación Interbooks arquitectura y desarrollo

Requisitos de la aplicación

1. Plataforma

Los requisitos para una correcta instalación y uso de la aplicación Interbooks serán:

-La aplicación desarrollada deber ser compatible con el mayor número posible de dispositivos, por lo que he elegido como plataforma un Smartphone o Tablet con sistema operativo Android 2.2 (Froyo) o superior (API 8). Para que la aplicación sea compatible con prácticamente todos los dispositivos Android del mercado, no siendo determinante el tener un móvil medianamente moderno.

-El espacio disponible de al menos 1,4MB.

2. El Servidor Web

Quiero que el servidor web procese la aplicación del lado del servidor, realizando conexiones bidireccionales o unidireccionales y asíncronas con el cliente. En esta ocasión estoy utilizando EasyPHP, el cual contiene el servidor web Apache *2.4.7*. Se deberá configurar el archivo httpd.conf (figura 6) con la instrucción "listen", la cual le indica al servidor que escuche todas las llamadas que entran por la dirección IP 127.0.0.1 a la que denominaré como "localhost".
Dentro de la carpeta localweb, alojada dentro del programa EasyPHP, crearé las carpetas referentes al servidor web; y en el interior de éstas, los archivos PHP, que permiten acceder a la base de dados.

3. La Base de Datos

La base de datos que utilizo es MySQL *5.6.15* que, al igual que el servidor web, viene instalado con EasyPHP. Accederé a ella de manera local a través de localhost. Para ello, utilizaré como nombre "root" y ninguna contraseña. En MySQL crearé dos bases de datos; una llamada "logreg", en la que se almacenará el registro de los usuarios; y otra llamada interbook, donde almacenaré el contenido de los mensajes creados por los usuarios. La función que espero de esta base de datos es que me almacene los datos de manera sistemática para su posterior uso.

Arquitectura de la Aplicación

Me conectaría a una base de datos MySQL (3) remota desde dentro de mi aplicación (1). Pero por desgracia, una base de datos MySQL y una aplicación para Android no se pueden comunicar directamente, por lo que crearé un sencillo servicio web Apache basado en PHP, que permitirá a mi aplicación de Android insertar datos en la base de datos y acceder a ellos cuando quiera. El sistema constará de una parte Android, que será el cliente que se conectará a un servidor. En este caso, será con un servidor Apache (2) con PHP.

Por ejemplo, si quiero crear un nuevo usuario en la aplicación desde mi dispositivo Android (1), tendré que introducir algunos valores de entrada, como el nombre de usuario y la contraseña. Cuando el usuario rellena esta información, se pasa la información a nuestro servicio web PHP. El servicio web se conectará a la base de datos MySQL, y buscará la tabla "Usuarios", por ejemplo, e insertará una nueva fila en la base de datos MySQL con la información que se envió desde nuestro dispositivo Android. Después, si queremos utilizarlos, manipularemos los datos que nos envía el servicio web en JSON para poder mostrarlos en Android, en un ListView, por ejemplo.

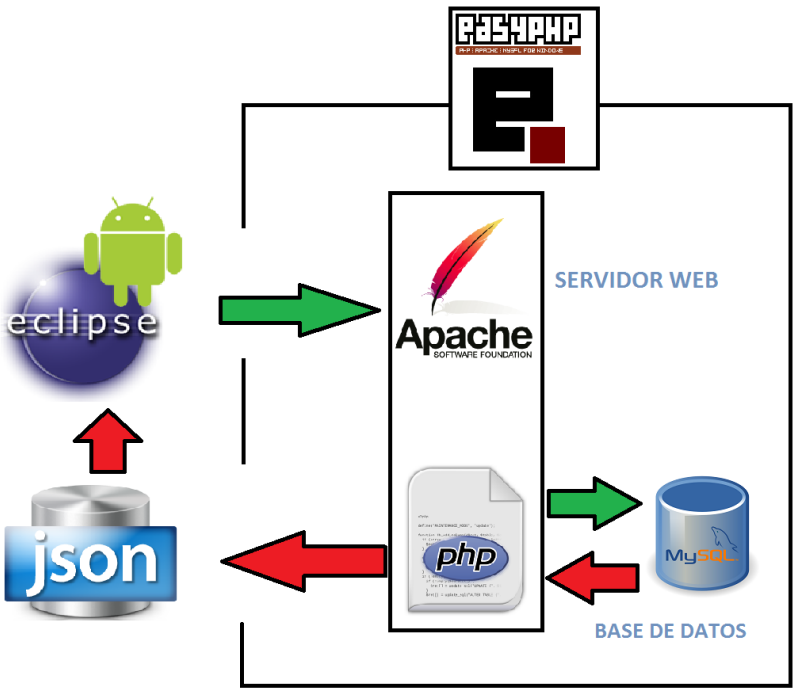


Figura 11: Arquitectura de la aplicación

Desarrollo del servidor

Voy a hablar del servidor web Apache. Como he mencionado en el punto anterior, lo haremos sobre php. Los archivos php con los que vamos a trabajar son los siguientes:

Archivos de inicio de sesión y registro. Carpeta:"logreg":

config.php.: Aquí pondremos nuestra configuración para poder conectar a la base de datos.

```
<?php
define("DB_HOST", "localhost");
define("DB_USER", "root");
define("DB_PASSWORD", "");
define("DB_DATABASE", "logreg");
?>
```

DB_Connect.php: Es el encargado de conectar a la base de datos o desconectarnos.

DB_Functions.php: contiene métodos para insertar o leer datos de la base de datos, como por ejemplo, para insertar el usuario que al crearlo devuelve un array con los nuevos datos; o buscarlo por email y contraseña, y comprobar si el usuario ya existe. También genera un identificador de usuario único y encriptará la contraseña utilizando el método **base64encode.**

index.php, él aceptará todas las peticiones GET y POST. Lo utilizaremos para gestionar todas las peticiones con la BD y dar respuesta en formato JSON.

La otra parte de los archivos php de la aplicación estará situada en la carpeta "Android_connect":

db_Config.php y db_connect.php : mismas funciones que las anteriores.

Crear_noticia.php: creará o insertará una nueva noticia (fila) en su tabla de la base de datos con todos sus campos (nombre, curso, asignaturas, acción, contacto y descripción).

Get_notices.php: nos permite formar una lista con todas las noticias.

get_detalles_noticia.php: vale para obtener los detalles individuales de las noticias, las noticias se identifican por su PID.

actualizar_noticia.php: actualizará La información de una noticia, las noticias se identifican por su PID.

borrar_noticia: borrara una noticia de la tabla de noticias, las noticias se identifican por su PID.

Desarrollo del cliente

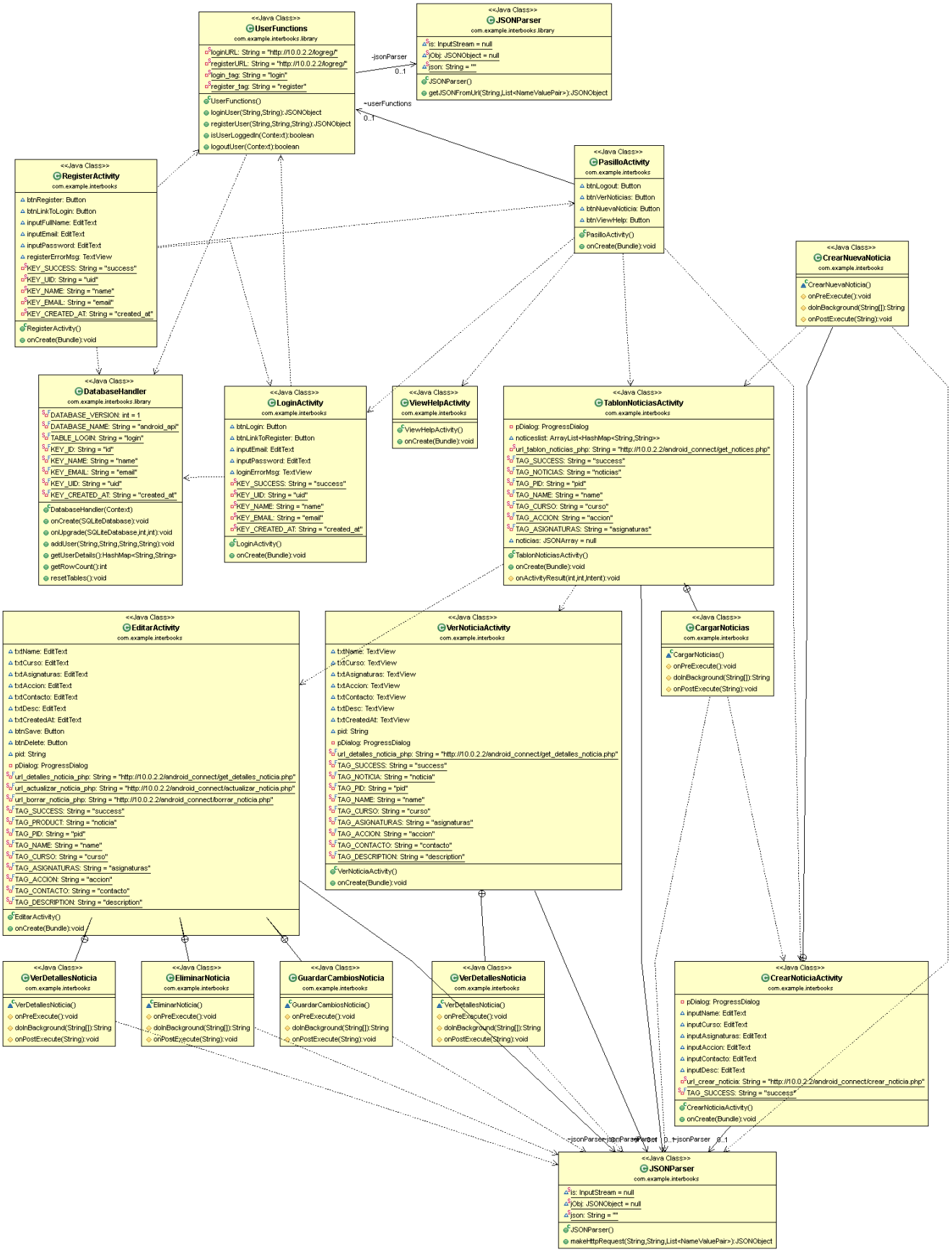


Figura 12: Diagrama UML de clases

Antes de entrar en detalles, daré un vistazo global del diagrama UML (figura 12) y del desarrollo del cliente de la aplicación.

De este diagrama comentaré las clases JSON Parser, las cuales utilizo para analizar las respuestas api JSON. Y con esa respuesta, la aplicación y el usuario conocerán si todo ha transcurrido correctamente o si, por el contrario, algún campo está vacío, o si el email y el password son incorrectos.

La otra clase admite dos métodos de solicitud HTTP GET y POST utilizados para obtener los datos json de las url. Éstos nos ayudarán a enviar y recibir información de la base de datos.

También comentaré el uso de la clase AsyncTask, ampliamente explicada en un punto anterior (pág. 27), con la que crearemos tareas en segundo plano para poder conectarnos al servidor, y realizar tareas como crear una noticia, editarla o eliminarla.

A continuación pasaré a explicar en detalle las actividades y el funcionamiento:

Lo primero que nos encontraremos, tras instalar la aplicación, será el icono de Interbooks. Después de pulsarlo pasaremos a la siguiente actividad.

Comenzaré por crear las Activities de login y Registro de cuenta.

Figura 13: login.xml y register.xml

LoginActivity.java - Actividad para manejar eventos de inicio de sesión. Con el botón Entrar (login.xml) pasamos el contexto de la activity, un email y una contraseña. Esta actividad se encargará de enviarlos al servidor para validar el proceso y almacenaremos los detalles del usuario en una base de datos SQlite.

Una vez logueados, y si todo se ha producido correctamente, nos llevará a la pantalla principal de la aplicación.

RegisterActivity.java

En el caso de que no estemos registrados pulsaremos otro botón (¿No tiene cuenta? registrarse) que nos llevará a la pantalla de registro para crear un nuevo usuario. Introduciremos el nombre, el email y la contraseña, la cual será encriptada, y nos mandará directamente a la pantalla principal de la aplicación.

Clases utilizadas que se encuentran en la librería:

UserFunctions.java

Funciones de la clase UserFunctions.java

loginUser(): solicitud de logueo de usuario, parámetros (email y password).

RegisterUser (): solicitud de registro de usuario, parámetros (name, email y password).

IsUserLoggedIn (): comprueba en nuestra aplicación si ya hay un usuario registrado.

LogoutUser (): elimina la información almacenada de un usuario en la BD del dispositivo.

DatabaseHandler.java: en la aplicación se almacena la información del usuario que se registra usando la base de datos no remota SQlite.

JSONParser.java: utilizado para analizar las respuestas api JSON, las cuales nos indican, por ejemplo, si un usuario se ha almacenado correctamente o no; si el usuario ya existía; o si se ha equivocado al introducir el email y el password.

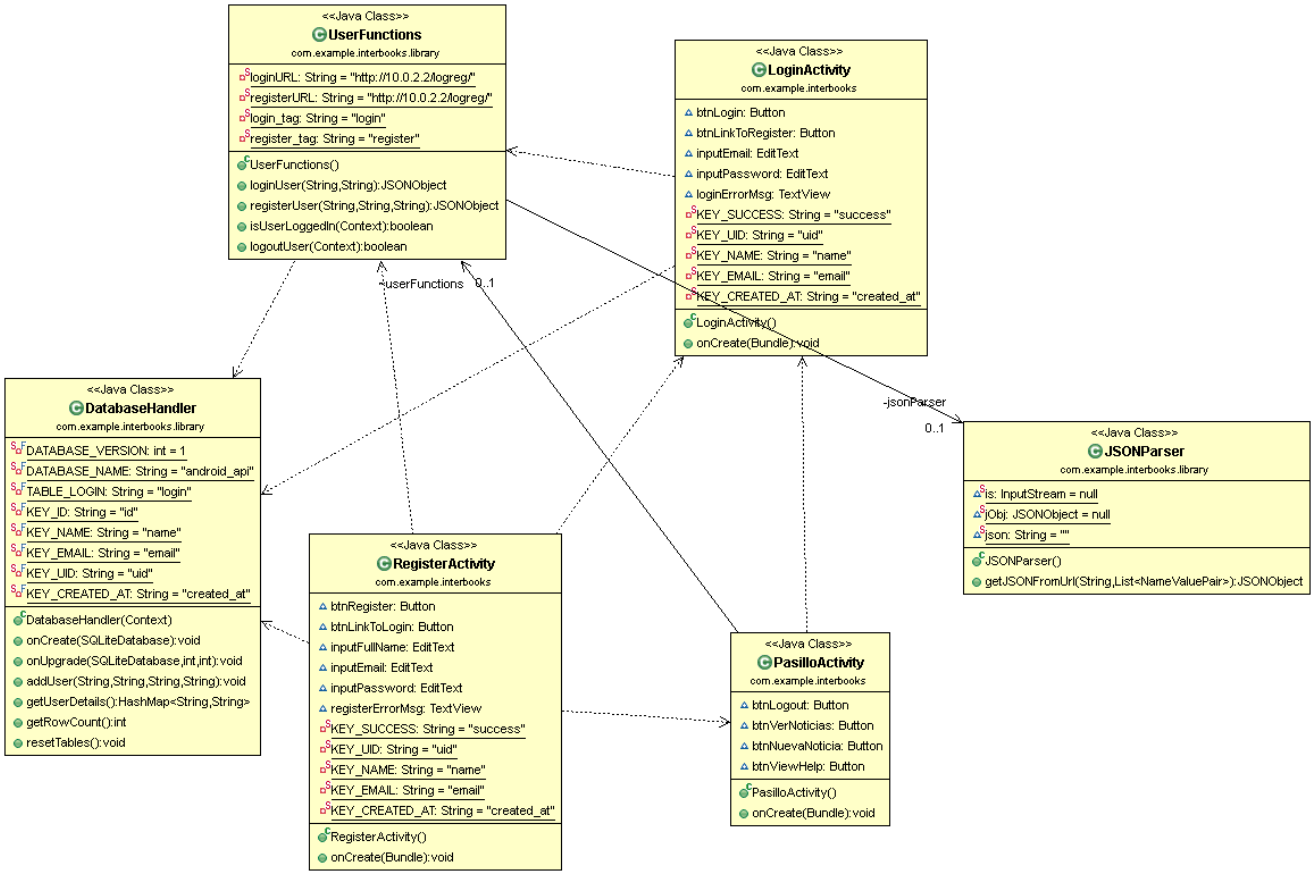


Figura 14: UML login y registro

En la figura anterior (figura 14) represento las clases y funciones explicadas en el punto anterior. Las cuales utilizo durante el registro, el logueo de la aplicación y la relación que existe entre ellas.

PasilloActivity.java

Tras iniciar sesión o registrarnos de manera correcta pasamos a la siguiente actividad **PasilloActivity.java** clase principal, que da función a los botones a través de eventos para movernos por otras actividades de la aplicación, como **TablonNoticiasActivity.java, CrearNoticiaActivity.java y ViewHelpActivity.java** o para cerrar la sesión. Si cerramos la sesión, al volver a abrir la aplicación habrá que volver a introducir el correo y el password.

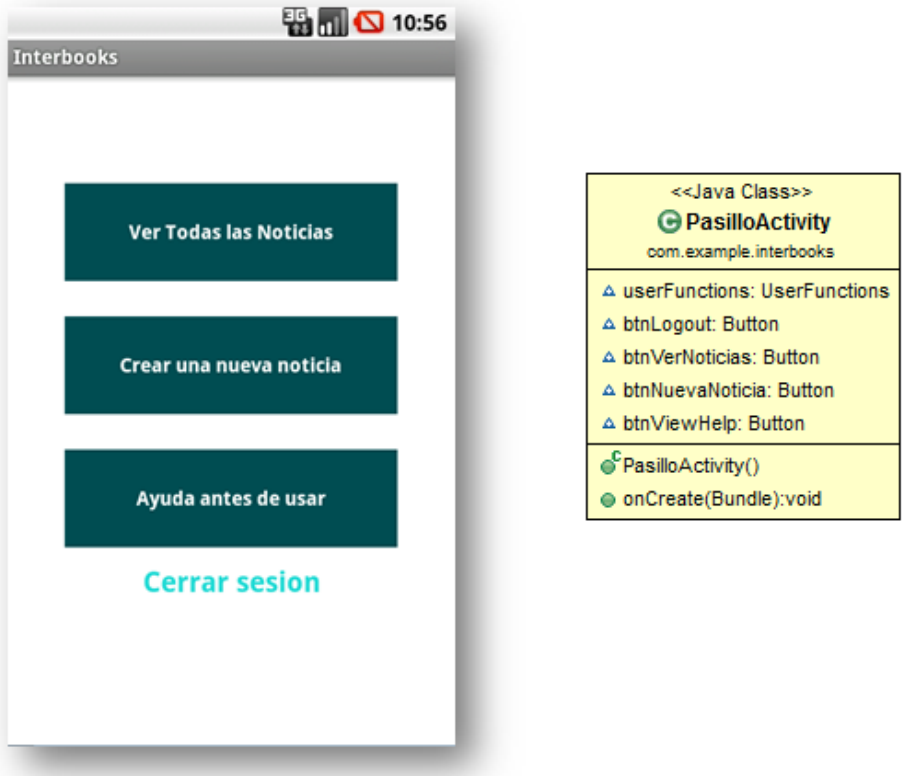


Figura 15: pasillo.xml y UML

TablonNoticiasActivity.java

Ahora vamos a crear una actividad que muestre todas las noticias o mensajes que los demás usuarios y nosotros hemos enviado en forma de lista o tablón. Para hacer cualquier tipo de lista en Android necesitaremos dos archivos XML; uno que contenga un listview; y otro que sea un elemento de esa lista. Ese elemento o fila de esa lista contendrá la información que queremos que aparezca de cada noticia o mensaje creado por el usuario.

En mi caso, dispongo de 4 campos: nombre, curso, asignaturas y acción que solicita el usuario, información indispensable para que otros usuarios conozcan si les interesa saber más sobre esta noticia.

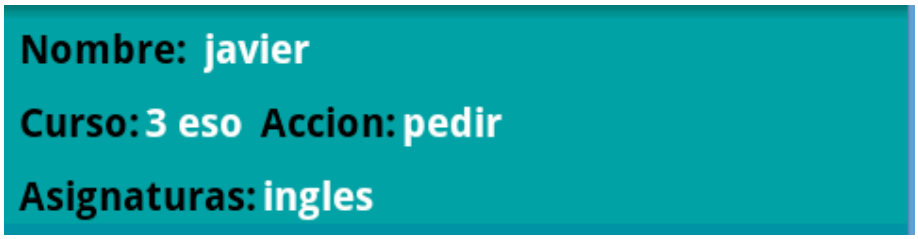


Figura 16: List_item.xml

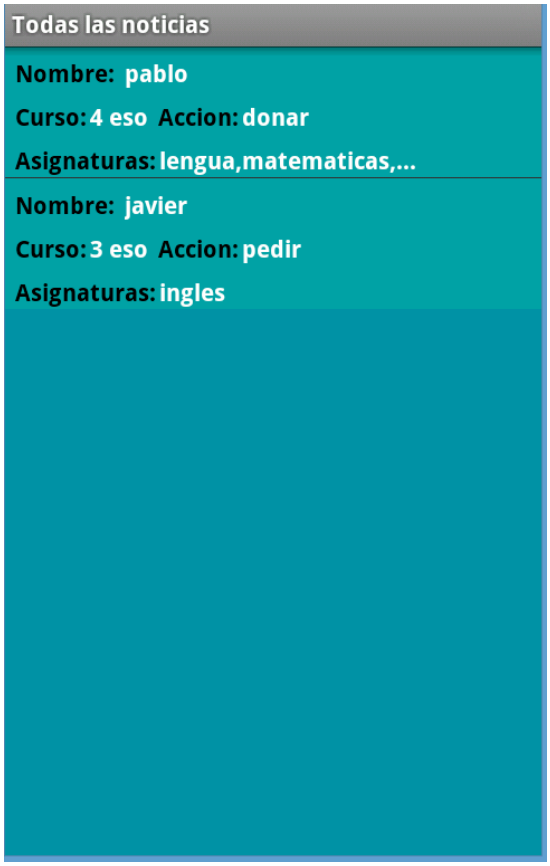


Figura 17: tablon_anuncios.xml

Desde Android accederemos al servidor web enviando una solicitud o petición en segundo plano AsyncTask, explicada en otro punto de la memoria al fichero php get_notices.php. Solicitando la lista con todas las noticias y sus respectivos datos.

Estos datos que recibiré de mi petición php los "parsearé" a un objeto JSONObjects y los adaptaré al listview para poder mostrarlos.

En el caso de que no exista ninguna noticia en el tablón lanzaremos la actividad CrearNoticiaActivity.java

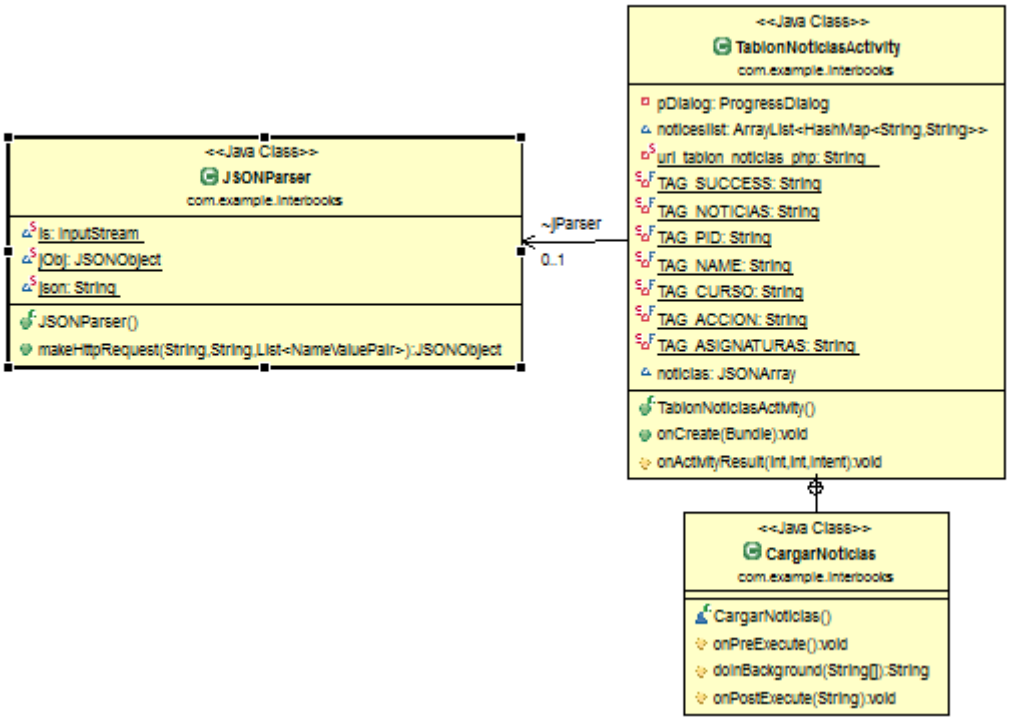


Figura 18: uml TablonNoticiasActivity.java

VerNoticiaActivity.java

Cuando pulsemos un elemento de la lista se nos pasará a otra actividad en la que se nos mostrará la noticia con todos los detalles y campos que nos faltaban. Los cuales nos describen más detalladamente lo que el usuario solicita y la forma de contactar con él. El servidor web y el archivo php get_detalles_noticia.php me devolverá en formato JSON los detalles de la noticia para poder utilizarla en mi app.

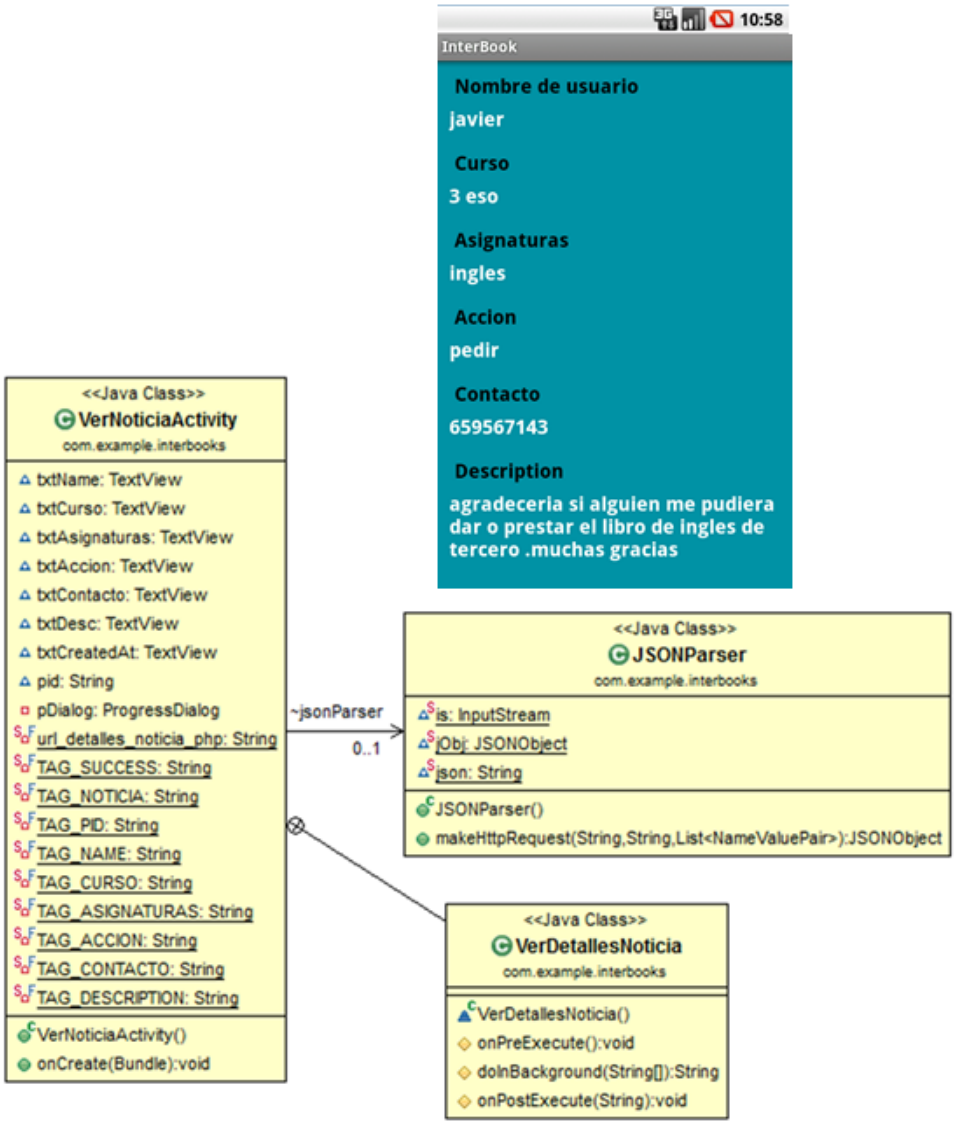


Figura 19: UML y ver_detalles_noticia.xml

CrearNoticiaActivity.java

En el caso de que apretemos el botón "crear una nueva noticia", de la actividad PasilloActivity.java, pasaremos a la actividad CrearNoticiaActivity.java, donde podremos crear una nueva noticia que agregar al Tablón de noticias y donde escribiremos la acción que deseamos realizar: intercambiar un libro con alguien, donárselo, prestarlo, solicitarlo o pedirlo. Será necesario rellenar todos los campos para poder crear la noticia, ya que todos son necesarios.

Cuando se pulse el botón crear nueva noticia, en el caso de que todos los campos estén llenos, llamará al método CrearNuevaNoticia ().

Nuestros datos se leerán en forma EditText y realizaremos una solicitud a crear_noticia.php para crear una nueva noticia con todos esos parámetros a través de HTTP POST. Cuando recibamos una respuesta json de crear_noticia.php, si se realiza con éxito (el bit es 1), pasaremos a la actividad TablonNoticiasActivity.java que se actualizará con una nueva noticia que acabamos de crear.

Figura 20: Crear_noticia.xml

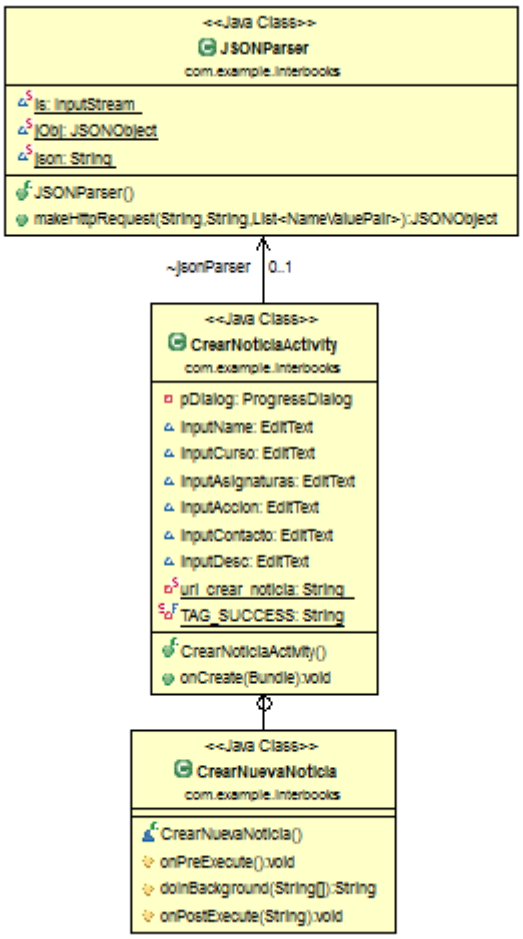


Figura 21: UML CrearNoticiaActivity.java

ViewHelpActivity.java

En el caso de que apretemos el botón Ayuda antes de usar, de la actividad PasilloActivity.java, pasaremos a la actividad ViewHelpActivity.java, donde se nos explicará de manera simple cómo utilizar mi aplicación.

Es una actividad sencilla en la que no hará falta pedir o enviar nada a la base de datos, sólo nos representará el layout con el texto fijo que le quiero poner, almacenado en el archivo String.xml.

```
package com.example.interbooks;

import android.app.Activity;
import android.os.Bundle;

public class ViewHelpActivity extends Activity{

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.view_help);

		}

}
```

Figura 22: UML y view_help.xml

EditarActivity.java

Esta última actividad será la encargada de modificar las noticias almacenadas en la base de datos, actualizarlas en el caso de que se nos haya pasado añadir algo, o eliminarla en el caso de que ya se haya producido la acción con otro cliente. Es decir, que se haya producido el intercambio; el préstamo; la donación; que por un cambio de idea no queramos prestar o donar nuestros libros; o que ya tengamos el libro que queríamos intercambiar o pedir.

Figura 23 : editar_noticia.xml

Obtenemos la identificación de la noticia (PID) de la intención que ha comenzado la actividad, la cual se envía desde el listview donde estaba la noticia.

Comenzaremos realizando una solicitud a get_detalles_noticia.php con la obtención de los detalles de la noticia en segundo plano y en formato json Mostrándose en varios EditText.
Después de mostrar los datos de la noticia, se esperará a que el usuario pulse alguno de los dos botones:

Si el usuario hace click en el botón guardar, se hace una solicitud HTTP a actualizar_noticia.php para almacenar los datos de las noticias actualizadas. Todo esto se realizará en un segundo plano AsyncTask.

Si el usuario hace click en el botón eliminar, se hace una solicitud HTTP a borrar_noticia.php, comenzar en segundo plano la eliminación de la noticia de la base de datos MySQL y la actualización de la vida del tablón de noticias.

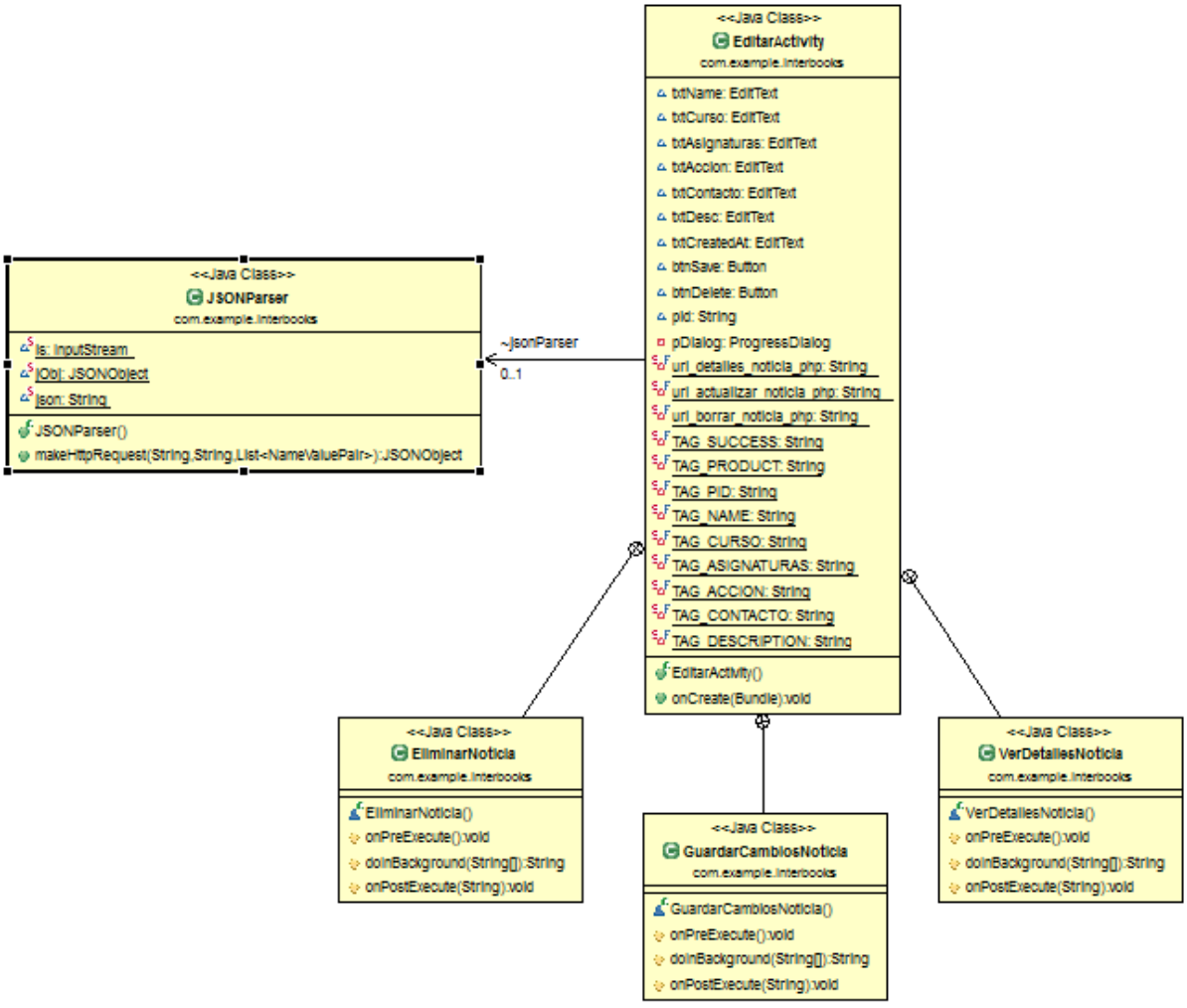


Figura 24 : UML EditarActivity.class

JSON Parser class: esta clase admite dos métodos de solicitud HTTP GET y POST para obtener los datos json de las url.

AndroidManifest.xml

Cada aplicación Android que se desarrolle debe de tener un archivo con el nombre AndroidManifest.xml. Este fichero contiene información sensible sobre la aplicación que el sistema operativo debe conocer y está situado en la raíz de cada aplicación. Esta información es obligatoria para que la aplicación pueda ejecutarse. El icono de la aplicación que aparecerá posteriormente al instalarse en los menús y nombre de la misma se mostrará definida aquí.

Añadiremos el siguiente permiso para que nuestra aplicación pueda conectarse a internet.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Entre las informaciones que maneja este archivo podemos encontrar:

- Contiene el nombre del paquete de desarrollo de la aplicación, el cual se utiliza como identificador único de la aplicación dentro del sistema.
- Define cada uno de los componentes de la aplicación, las activities, services, brodcast recievers y los content providers, los cuales componen una aplicación.
- Se declara el nivel mínimo y la versión para la que fue creada de la API, que el terminal Android debe tener instalado.
- Determina qué procesos son los que se encargan de acoger a los componentes implementados por la aplicación.
- Indica los tipos de permisos que debe tener la aplicación a la hora de querer acceder a partes protegidas de la API e interactuar con otras aplicaciones.

Conclusiones

Tras finalizar el proyecto y la memoria, llega la hora de reflexionar sobre todo lo que he aprendido, las dificultades que he encontrado y aquello que pueda mejorar.

La principal razón por la que elegí este proyecto fue las ganas de aprender a desarrollar aplicaciones para Android, que me ha supuesto el extra de recordar e incrementar los conocimientos adquiridos anteriormente sobre java. Este proyecto también me ha supuesto adquirir muchos más conocimientos que, previamente, no tenía para el mundo laboral, como XML, php, MySQL y servidores web. Por lo que todo el trabajo realizado ha merecido la pena.

Dificultades encontradas

Uno de los problemas de los sistemas operativos Android es que no es posible realizar una conexión al servidor en una única tarea principal, por lo que es necesario crear tareas adicionales en el servidor para poder hacer esta conexión. Por esta razón, tuve que buscar una manera, que desconocía, de solucionarlo.

Aquí muestro alguno de los fallos más frecuentes de su utilización:

1. No se puede llamar a un AsyncTask anidado o dentro de otro AsyncTask.
2. La llamada a un método de ejecución debe de realizarse desde el hilo de la interfaz de usuario.

También surgieron algunos problemas con la configuración del entorno de aplicación, aunque el entorno de desarrollo es bastante intuitivo. Si no se configura debidamente, o no tenemos las APIs adecuadas y la versión de java adecuada, puede dar muchos problemas. Con Windows 8 tuve que buscar una versión anterior de java SE.

Mejoras y posibles aplicaciones

- Mejorar la aplicación en cuanto a la globalización o comercialización de la misma, dividiéndola por comunidades, ciudades o institutos, haciéndola útil a una mayor cantidad de público.

- Comercialización de la misma, añadiendo anuncios o un método de donación, además de la subida a la Play Store.

- Aplicación o utilización de este modelo para el intercambio, venta o compra de otro tipo de productos como, por ejemplo, automóviles, juegos, dvds, etc.

- Implementación en un servidor externo protegido.

- Inserción de un sistema de búsquedas que mejore la búsqueda al usuario final.

Bibliografía

Cursos realizados por el autor del trabajo:

Programación en PHP
Impartido por la Plataforma de eFormación de la CARM (form@carm)
De 35 horas de duración.

Desarrollo de aplicaciones para Android
Impartido por la Plataforma de eFormación de la CARM (form@carm)
De 35 horas de duración.

Curso de la Universitat Politècnica de València:
Android: Programación de Aplicaciones
120 horas de duración.

[1] Página oficial de *Android*. Se encuentra toda la documentación de *Android*.
<http://www.android.com/>

[2] Página oficial de *MySQL*. Documentación de la base de datos *MySQL*.
<http://www.mysql.com/>

[3] El gran libro de Android.
Jesús Tomas Girones 2ªedicion (2012), Editorial: Alfaomega - Marcombo
[4] MySQL guía de aprendizaje
Ullman, Larry

[5] Evolución de Android
<http://androidzone.org/2013/05/historia-de-android-la-evolucion-a-lo-largo-de-sus-versiones/>

[6] Arquitectura de Android
<http://androideity.com/2011/07/04/arquitectura-de-android/>

[7] Stack Overflow: 'android' questions.
<http://stackoverflow.com/questions/tagged/android>

[8] Conferencia: Ganar dinero con Android
<http://www.emezeta.com/articulos/conferencia-ganar-dinero-con-android>

Anexo

CrearNoticiaActivity.java

```
package com.example.interbooks;

import java.util.ArrayList;
import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class CrearNoticiaActivity extends Activity {

	// Progress Dialog
	private ProgressDialog pDialog;

	JSONParser jsonParser = new JSONParser();
	EditText inputName;
	EditText inputCurso;
	EditText inputAsignaturas;
	EditText inputAccion;
	EditText inputContacto;
	EditText inputDesc;

	// url para crear una nuega noticia
	private static String url_crear_noticia =
"http://10.0.2.2/android_connect/crear_noticia.php";

	// JSON nombres del nodo
	private static final String TAG_SUCCESS = "success";

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.crear_noticia);

		// EditTexts
		inputName = (EditText) findViewById(R.id.inputName);
		inputCurso = (EditText) findViewById(R.id.inputCurso);
		inputAsignaturas = (EditText)
findViewById(R.id.inputAsignaturas);
		inputAccion = (EditText) findViewById(R.id.inputAccion);
		inputContacto = (EditText) findViewById(R.id.inputContacto);
		inputDesc = (EditText) findViewById(R.id.inputDesc);

		// creamos un boton , capturamos los objetos graficos que vamos a
utilizar
		Button btnCrearNoticia = (Button)
findViewById(R.id.btnCrearNoticia);

 // creamos un evento onclick para el boton, cada vez que
// se pulse , en el caso de que todos los campos esten llenos llamara al
metodo CrearNuevaNoticia
```

```
            btnCrearNoticia.setOnClickListener(new View.OnClickListener() {

                  @Override
                  public void onClick(View view) {

                        if (  ( !inputName.getText().toString().equals(""))
&& ( !inputCurso.getText().toString().equals("")) && (
!inputAsignaturas.getText().toString().equals("")) && (
!inputAccion.getText().toString().equals("")) && (
!inputContacto.getText().toString().equals("")) && (
!inputDesc.getText().toString().equals("")) )
                        {

                            // creamos una nueva noticia en segundo plano
                            new CrearNuevaNoticia().execute();

                        }else{

                            Toast.makeText(getApplicationContext(),"todos los
campos son obligatorios", Toast.LENGTH_SHORT).show();


                        }
                  }
            });
      }


       // Tarea en segundo plano AsyncTask para crear una nueva noticia

      class CrearNuevaNoticia extends AsyncTask<String, String, String> {

            @Override
            protected void onPreExecute() {
                  super.onPreExecute();
                  pDialog = new ProgressDialog(CrearNoticiaActivity.this);
                  pDialog.setMessage("Creando tu noticia..");
                  pDialog.setIndeterminate(false);
                  pDialog.setCancelable(true);
                  pDialog.show();
            }

            //Almacenarmos lo que metemos en los EditText en sus variables

            protected String doInBackground(String... args) {
                  String name = inputName.getText().toString();
                  String curso = inputCurso.getText().toString();
                  String asignaturas = inputAsignaturas.getText().toString();
                  String accion = inputAccion.getText().toString();
                  String contacto = inputContacto.getText().toString();
                  String description = inputDesc.getText().toString();

                  // Construimos los parametros
                  List<NameValuePair> params = new
ArrayList<NameValuePair>();
                  params.add(new BasicNameValuePair("name", name));
                  params.add(new BasicNameValuePair("curso",curso));
                  params.add(new BasicNameValuePair("asignaturas",
asignaturas));
                  params.add(new BasicNameValuePair("accion", accion));
                  params.add(new BasicNameValuePair("contacto", contacto));
                  params.add(new BasicNameValuePair("description",
description));

// Se realiza una solicitud a crear_noticia.php para crear una nueva noticia a
través de HTTP POST.
```

```
                    JSONObject json =
jsonParser.makeHttpRequest(url_crear_noticia,"POST", params);

                    Log.d("Crear respuesta", json.toString());

//obtendremos una respuesta json de crear_noticia.php, si se realiza con exito
pasaremos a la actividad //TablonNoticiasActivity.class que se actualizara con
una nueva noticia que acabams de crear
                    try {
                          int success = json.getInt(TAG_SUCCESS);

                          if (success == 1) {
                                // successfully created product
                                Intent i = new
Intent(getApplicationContext(), TablonNoticiasActivity.class);
                                startActivity(i);

                                finish();
                          } else {

                          }
                    } catch (JSONException e) {
                          e.printStackTrace();
                    }

                    return null;
             }


             protected void onPostExecute(String file_url) {
                    pDialog.dismiss();
             }

      }
}
```

EditarActivity.java

```
package com.example.interbooks;

import java.util.ArrayList;
import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class EditarActivity extends Activity {

      EditText txtName;
      EditText txtCurso;
      EditText txtAsignaturas;
      EditText txtAccion;
      EditText txtContacto;
```

```
        EditText txtDesc;
        EditText txtCreatedAt;
        Button btnSave;
        Button btnDelete;

        String pid;

        private ProgressDialog pDialog;

        // JSON parser class
        JSONParser jsonParser = new JSONParser();

        // ver detalles de la noticia
        private static final String url_detalles_noticia_php =
"http://10.0.2.2/android_connect/get_detalles_noticia.php";

        // actualizar noticia
        private static final String url_actualizar_noticia_php =
"http://10.0.2.2/android_connect/actualizar_noticia.php";

        // borrar noticia
        private static final String url_borrar_noticia_php =
"http://10.0.2.2/android_connect/borrar_noticia.php";

        // JSON Node names
        private static final String TAG_SUCCESS = "success";
        private static final String TAG_PRODUCT = "noticia";
        private static final String TAG_PID = "pid";
        private static final String TAG_NAME = "name";
        private static final String TAG_CURSO = "curso";
        private static final String TAG_ASIGNATURAS = "asignaturas";
        private static final String TAG_ACCION = "accion";
        private static final String TAG_CONTACTO = "contacto";
        private static final String TAG_DESCRIPTION = "description";

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.editar_noticia);

                // botones
                btnSave = (Button) findViewById(R.id.btnSave);
                btnDelete = (Button) findViewById(R.id.btnDelete);

                // obtenemos la intencion que a comenzado la actividad

                Intent i = getIntent();

                //Obtenemos la identificación de la noticia (PID) de la intención
                pid = i.getStringExtra(TAG_PID);

                // comenzamos la obtencion de los detalles de la noticia en un
segundo plano
                new VerDetallesNoticia().execute();

                // pulsar el boton que actualizara la noticia (guardar cambios)
                btnSave.setOnClickListener(new View.OnClickListener() {

                        @Override
                        public void onClick(View arg0) {
                                //comienza en segundo plano la actualizacion de la
noticia
                                new GuardarCambiosNoticia().execute();
                        }
                });

                // pulsar el boton que borrara la noticia (eliminar)
                btnDelete.setOnClickListener(new View.OnClickListener() {
```

```
                    @Override
                    public void onClick(View arg0) {
                          //comienza en segundo plano la eliminacionn de la
noticia
                          new EliminarNoticia().execute();
                    }
              });

       }

       // Tarea en segundo plano AsyncTask para obtener todos los detalles de
la noticia en sus EditText

       class VerDetallesNoticia extends AsyncTask<String, String, String> {


              @Override
              protected void onPreExecute() {
                    super.onPreExecute();
                    pDialog = new ProgressDialog(EditarActivity.this);
                    pDialog.setMessage("Cargando detalles de la noticia");
                    pDialog.setIndeterminate(false);
                    pDialog.setCancelable(true);
                    pDialog.show();
              }

                // Obtenemos los detalles en segundo plano

              protected String doInBackground(String... params) {

              // //actualizamos la ui Cinterfaz de usuario) a partir de la
tarea en segundo plano
                    runOnUiThread(new Runnable() {
                          public void run() {
                                int success;
                                try {
                                      // Construimos los parametros que vamos
a comprobar
                                List<NameValuePair> params = new
ArrayList<NameValuePair>();
                                      params.add(new
BasicNameValuePair("pid", pid));

                          //  Obtenemos los detalles del producto haciendo una
petición HTTP

              JSONObject json =
jsonParser.makeHttpRequest(url_detalles_noticia_php, "GET", params);

                                      // check your log for json response
                                      Log.d("detalles de la noticia",
json.toString());

                                      // json success tag
                                      success = json.getInt(TAG_SUCCESS);
                                      if (success == 1) {
                                            // successfully received product
details
                                JSONArray productObj =
json.getJSONArray(TAG_PRODUCT); // JSON Array
                                   //  obrenemos la primera (0) del JSONArray
                                JSONObject notice = productObj.getJSONObject(0);

                           //noticias con esa identificacion(pid)encontrados

                                    txtName = (EditText)
findViewById(R.id.inputName);
```

```
                            txtCurso = (EditText)
findViewById(R.id.inputCurso);
                           txtAsignaturas = (EditText)
findViewById(R.id.inputAsignaturas);
                            txtAccion = (EditText)
findViewById(R.id.inputAccion);
                              txtContacto = (EditText)
findViewById(R.id.inputContacto);
                           txtDesc = (EditText) findViewById(R.id.inputDesc);

                                 // Datos de la noticia que mostraremos en un
EditText
      txtName.setText(notice.getString(TAG_NAME));
txtCurso.setText(notice.getString(TAG_CURSO
      txtAsignaturas.setText(notice.getString(TAG_ASIGNATURAS));

      txtAccion.setText(notice.getString(TAG_ACCION));

      txtContacto.setText(notice.getString(TAG_CONTACTO));

txtDesc.setText(notice.getString(TAG_DESCRIPTION));

                                       }else{

                                       }
                                 } catch (JSONException e) {
                                       e.printStackTrace();
                                 }
                           }
                     });

                     return null;
               }

               //desechamos pDialog una vez terminado el proceso en segundo
plano
               protected void onPostExecute(String file_url) {
                     pDialog.dismiss();
               }
      }

       //Tarea en segundo plano AsyncTask para guardar los cambios realizados
en nuestra noticia

      class GuardarCambiosNoticia extends AsyncTask<String, String, String> {

               @Override
               protected void onPreExecute() {
                     super.onPreExecute();
                     pDialog = new ProgressDialog(EditarActivity.this);
                     pDialog.setMessage("Guardando cambios ...");
                     pDialog.setIndeterminate(false);
                     pDialog.setCancelable(true);
                     pDialog.show();
               }

                 // Actualizamos la noticia

               protected String doInBackground(String... args) {

                     //Almacenarmos lo que metemos en los EditText en sus
variables
                     String name = txtName.getText().toString();
                     String curso = txtCurso.getText().toString();
                     String asignaturas = txtAsignaturas.getText().toString();
                     String accion = txtAccion.getText().toString();
                     String contacto = txtContacto.getText().toString();
                     String description = txtDesc.getText().toString();
```

```
                    //    construyendo los parametros
                    List<NameValuePair> params = new
ArrayList<NameValuePair>();
                    params.add(new BasicNameValuePair(TAG_PID, pid));
                    params.add(new BasicNameValuePair(TAG_NAME, name));
                    params.add(new BasicNameValuePair(TAG_CURSO, curso));
                    params.add(new BasicNameValuePair(TAG_ASIGNATURAS,
asignaturas));
                    params.add(new BasicNameValuePair(TAG_ACCION, accion));
                    params.add(new BasicNameValuePair(TAG_CONTACTO, contacto));
                    params.add(new BasicNameValuePair(TAG_DESCRIPTION,
description));


                    // Enviamos los datos modificados a través de la solicitud
http

             JSONObject json =
jsonParser.makeHttpRequest(url_actualizar_noticia_php,"POST", params);

                    // check json success tag
                    try {
                           int success = json.getInt(TAG_SUCCESS);
                           // 100 is some code to identify the returning result
                           if (success == 1) {
                                  Intent i = getIntent();
                                  setResult(100, i);
                                  finish();
                           } else {
                           }
                    } catch (JSONException e) {
                           e.printStackTrace();
                    }

                    return null;
             }



             protected void onPostExecute(String file_url) {
                    pDialog.dismiss();
             }
       }

        //Tarea en segundo plano AsyncTask para eliminar una noticia

       class EliminarNoticia extends AsyncTask<String, String, String> {


             @Override
             protected void onPreExecute() {
                    super.onPreExecute();
                    pDialog = new ProgressDialog(EditarActivity.this);
                    pDialog.setMessage("Eliminando la noticia...");
                    pDialog.setIndeterminate(false);
                    pDialog.setCancelable(true);
                    pDialog.show();
             }

             // eliminando la noticia

             protected String doInBackground(String... args) {

                    int success;
                    try {
                           List<NameValuePair> params = new
ArrayList<NameValuePair>();
```

```
                        params.add(new BasicNameValuePair("pid", pid));

            JSONObject json =
jsonParser.makeHttpRequest(url_borrar_noticia_php, "POST", params);

                        // check your log for json response
                        Log.d("borrando noticia", json.toString());

                        // json success tag
                        success = json.getInt(TAG_SUCCESS);
                        if (success == 1) {
                              Intent i = getIntent();
                              setResult(100, i);
                              finish();
                        }
                  } catch (JSONException e) {
                        e.printStackTrace();
                  }

                  return null;
            }


            protected void onPostExecute(String file_url) {
                        pDialog.dismiss();

            }

      }
}
```

JSONParser.java

```
package com.example.interbooks;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.List;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocol
Exception;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONException;
import org.json.JSONObject;
import android.util.Log;
public class JSONParser {

      static InputStream is = null;
      static JSONObject jObj = null;
      static String json = "";

      // constructor
      public JSONParser() {

      }

      // function get json from url
```

```
	// by making HTTP POST or GET mehtod
	public JSONObject makeHttpRequest(String url, String method,
			List<NameValuePair> params) {

		// Making HTTP request
		try {

			// check for request method
			if(method == "POST"){
				// request method is POST
				// defaultHttpClient
				DefaultHttpClient httpClient = new
DefaultHttpClient();
				HttpPost httpPost = new HttpPost(url);
				httpPost.setEntity(new
UrlEncodedFormEntity(params));

				HttpResponse httpResponse =
httpClient.execute(httpPost);
				HttpEntity httpEntity = httpResponse.getEntity();
				is = httpEntity.getContent();

			}else if(method == "GET"){
				// request method is GET
				DefaultHttpClient httpClient = new
DefaultHttpClient();
				String paramString = URLEncodedUtils.format(params,
"utf-8");
				url += "?" + paramString;
				HttpGet httpGet = new HttpGet(url);

				HttpResponse httpResponse =
httpClient.execute(httpGet);
				HttpEntity httpEntity = httpResponse.getEntity();
				is = httpEntity.getContent();
			}


		} catch (UnsupportedEncodingException e) {
			e.printStackTrace();
		} catch (ClientProtocolException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		}

		try {
			BufferedReader reader = new BufferedReader(new
InputStreamReader(
					is, "iso-8859-1"), 8);
			StringBuilder sb = new StringBuilder();
			String line = null;
			while ((line = reader.readLine()) != null) {
				sb.append(line + "\n");
			}
			is.close();
			json = sb.toString();
		} catch (Exception e) {
			Log.e("Buffer Error", "Error converting result " +
e.toString());
		}

		// try parse the string to a JSON object
		try {
			jObj = new JSONObject(json);
		} catch (JSONException e) {
			Log.e("JSON Parser", "Error parsing data " + e.toString());
		}
```

```
            // return JSON String
            return jObj;

        }
}
```

LoginActivity.java

```
package com.example.interbooks;

import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.example.interbooks.library.DatabaseHandler;
import com.example.interbooks.library.UserFunctions;

public class LoginActivity extends Activity {
    Button btnLogin;
    Button btnLinkToRegister;
    EditText inputEmail;
    EditText inputPassword;
    TextView loginErrorMsg;

    // nombre de nodos JSON
    private static String KEY_SUCCESS = "success";
    private static String KEY_UID = "uid";
    private static String KEY_NAME = "name";
    private static String KEY_EMAIL = "email";
    private static String KEY_CREATED_AT = "created_at";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);

        //botones y EditText
        inputEmail = (EditText) findViewById(R.id.loginEmail);
        inputPassword = (EditText) findViewById(R.id.loginPassword);
        btnLogin = (Button) findViewById(R.id.btnLogin);
        btnLinkToRegister = (Button)
findViewById(R.id.btnLinkToRegisterScreen);
        loginErrorMsg = (TextView) findViewById(R.id.login_error);

        btnLogin.setOnClickListener(new View.OnClickListener() {

            public void onClick(View view) {
                String email = inputEmail.getText().toString();
                String password = inputPassword.getText().toString();
                UserFunctions userFunction = new UserFunctions();
                JSONObject json = userFunction.loginUser(email, password);

                //comprobamos la respuesta
                try {
                    if (json.getString(KEY_SUCCESS) != null) {
                        loginErrorMsg.setText("");
                        String res = json.getString(KEY_SUCCESS);
```

```
                        if(Integer.parseInt(res) == 1){
                          //el usuario se loguea con exito ==1
                          //almacenamos los detalles del usuario en una base
de datos SQlite
                            DatabaseHandler db = new
DatabaseHandler(getApplicationContext());
                            JSONObject json_user = json.getJSONObject("user");

                            // limpiamos todos los datos anteriores en la base
de datos
                            userFunction.logoutUser(getApplicationContext());
                            db.addUser(json_user.getString(KEY_NAME),
json_user.getString(KEY_EMAIL), json.getString(KEY_UID),
json_user.getString(KEY_CREATED_AT));

                            // pasamos a la actividad
                            Intent dashboard = new
Intent(getApplicationContext(), PasilloActivity.class);

                            // cerramos todas las actividades anteriores

dashboard.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                            startActivity(dashboard);


                            finish();
                        }else{
                            // ERROR AL INICIAR SESION
                            loginErrorMsg.setText("email|password
incorrectos");
                        }
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        });


        btnLinkToRegister.setOnClickListener(new View.OnClickListener() {

            public void onClick(View view) {
                Intent i = new
Intent(getApplicationContext(),RegisterActivity.class);
                startActivity(i);
                finish();
            }
        });
    }
}
```

PasilloActivity.java

```
package com.example.interbooks;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import com.example.interbooks.library.UserFunctions;


public class PasilloActivity extends Activity{
       UserFunctions userFunctions;
```

```
	Button btnLogout;
	Button btnVerNoticias;
	Button btnNuevaNoticia;
	Button btnViewHelp;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);


		// Comprobamos que ya estamos logueados
	  userFunctions = new UserFunctions();
	  if(userFunctions.isUserLoggedIn(getApplicationContext())){
		setContentView(R.layout.pasillo);

		// Butones
		btnVerNoticias = (Button) findViewById(R.id.btnViewNotice);
		btnNuevaNoticia = (Button) findViewById(R.id.btnCreateNotice);
		btnViewHelp = (Button) findViewById(R.id.btnViewHelp);
		btnLogout = (Button) findViewById(R.id.btnLogout);



		btnVerNoticias.setOnClickListener(new View.OnClickListener() {

			@Override
			public void onClick(View view) {
				Intent i = new Intent(getApplicationContext(),
TablonNoticiasActivity.class);
				startActivity(i);

			}
		});


		btnNuevaNoticia.setOnClickListener(new View.OnClickListener() {

			@Override
			public void onClick(View view) {
				Intent i = new Intent(getApplicationContext(),
CrearNoticiaActivity.class);
				startActivity(i);

			}
		});


		btnViewHelp.setOnClickListener(new View.OnClickListener() {

			@Override
			public void onClick(View view) {
				// Launching create new product activity
				Intent i = new Intent(getApplicationContext(),
ViewHelpActivity.class);
				startActivity(i);

			}
		});

		btnLogout.setOnClickListener(new View.OnClickListener() {

			public void onClick(View arg0) {
				// TODO Auto-generated method stub
				userFunctions.logoutUser(getApplicationContext());
				Intent login = new Intent(getApplicationContext(),
LoginActivity.class);
			login.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
```

```
                    startActivity(login);
                    finish();
                    }
             });

    }else{
      //Usuario no Logueado
      Intent login = new Intent(getApplicationContext(), LoginActivity.class);
      login.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
      startActivity(login);
      finish();
    }
      }
}
```

RegisterActivity.java

```
package com.example.interbooks;

import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.interbooks.library.DatabaseHandler;
import com.example.interbooks.library.UserFunctions;

public class RegisterActivity extends Activity {
       Button btnRegister;
       Button btnLinkToLogin;
       EditText inputFullName;
       EditText inputEmail;
       EditText inputPassword;
       TextView registerErrorMsg;

       private static String KEY_SUCCESS = "success";
       private static String KEY_UID = "uid";
       private static String KEY_NAME = "name";
       private static String KEY_EMAIL = "email";
       private static String KEY_CREATED_AT = "created_at";

       @Override
       public void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.register);

              inputFullName = (EditText) findViewById(R.id.registerName);
              inputEmail = (EditText) findViewById(R.id.registerEmail);
              inputPassword = (EditText) findViewById(R.id.registerPassword);
       btnRegister = (Button) findViewById(R.id.btnRegister);
       btnLinkToLogin = (Button) findViewById(R.id.btnLinkToLoginScreen);
       registerErrorMsg = (TextView) findViewById(R.id.register_error);

              btnRegister.setOnClickListener(new View.OnClickListener() {

                     public void onClick(View view) {
```

```
                        if (  (
!inputFullName.getText().toString().equals("")) && (
!inputEmail.getText().toString().equals("")) && (
!inputPassword.getText().toString().equals(""))  )
                {

                    String name = inputFullName.getText().toString();
                    String email = inputEmail.getText().toString();
                    String password = inputPassword.getText().toString();
                          UserFunctions userFunction = new UserFunctions();
      JSONObject json = userFunction.registerUser(name, email, password);

                        try {
                              if (json.getString(KEY_SUCCESS) != null) {
                                    registerErrorMsg.setText("");
                        String res = json.getString(KEY_SUCCESS);
                                    if(Integer.parseInt(res) == 1){

      DatabaseHandler db = new DatabaseHandler(getApplicationContext());
                    JSONObject json_user = json.getJSONObject("user");


      userFunction.logoutUser(getApplicationContext());

      db.addUser(json_user.getString(KEY_NAME),
json_user.getString(KEY_EMAIL), json.getString(KEY_UID),
json_user.getString(KEY_CREATED_AT));

                                                Intent dashboard = new
Intent(getApplicationContext(), PasilloActivity.class);

      dashboard.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                                startActivity(dashboard);
                                                finish();
                                          }else{
                                                registerErrorMsg.setText("error
de registro");
                                          }
                                    }
                        } catch (JSONException e) {
                              e.printStackTrace();
                        }
                        }else{

                     Toast.makeText(getApplicationContext(),"todos los campos
son obligatorios", Toast.LENGTH_SHORT).show();


                  }
                   }
                    }
            );

            btnLinkToLogin.setOnClickListener(new View.OnClickListener() {

                   public void onClick(View view) {
                          Intent i = new Intent(getApplicationContext(),
                                LoginActivity.class);
                          startActivity(i);
                          finish();
                   }
            });
      }
}
```

TablonNoticiasActivity.java

```
package com.example.interbooks;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import org.apache.http.NameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.ListActivity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;

public class TablonNoticiasActivity extends ListActivity {

	// Progress Dialog
	private ProgressDialog pDialog;

	// Creating JSON Parser object
	JSONParser jParser = new JSONParser();

	ArrayList<HashMap<String, String>> noticeslist;

	// url para obtener la lista de todas las noticias
	private static String url_tablon_noticias_php =
"http://10.0.2.2/android_connect/get_notices.php";

	// JSON nombres del nodo
	private static final String TAG_SUCCESS = "success";
	private static final String TAG_NOTICIAS = "noticias";
	private static final String TAG_PID = "pid";
	private static final String TAG_NAME = "name";
	private static final String TAG_CURSO = "curso";
	private static final String TAG_ACCION = "accion";
	private static final String TAG_ASIGNATURAS = "asignaturas";


	// noticias JSONArray inicializamos
	JSONArray noticias = null;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.tablon_noticias);

		// Hashmap para el listview
		noticeslist = new ArrayList<HashMap<String, String>>();

		// Cargando noticias en un segundo plano
		new CargarNoticias().execute();

		// obtenemos el listview
		ListView lv = getListView();
```

```
            // Fijamos un evento onItemclick para el listview lv, cada vez
que
            // lo pulsemos se llamará a este método (que abrirá una actividad)

            lv.setOnItemClickListener(new OnItemClickListener() {

                  @Override
                  public void onItemClick(AdapterView<?> parent, View
view,int position, long id) {
                        // obtenemos los datos del listview seleccionado
String pid = ((TextView) view.findViewById(R.id.pid)).getText().toString();
                        // Creamos un intent con el nombre de la clase de la
actividad
Intent in = new Intent(getApplicationContext(),VerNoticiaActivity.class);
                        // enviamos el pid a la siguiente actividad para que
nos muestre esa en concreto
                        in.putExtra(TAG_PID, pid);
                        // iniciamos nueva actividad y esperamos respuesta
                        startActivityForResult(in, 100);
                  }
            });

      @Override
      protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
            super.onActivityResult(requestCode, resultCode, data);
            if (resultCode == 100) {
                  Intent intent = getIntent();
                  finish();
                  startActivity(intent);
            }

      }

      /**
       * hilo secundario para cargar las noticias mientras se solicita HTTP
Request
       * */
      class CargarNoticias extends AsyncTask<String, String, String> {


             //antes de comenzar el proceso en segundo plano mostramos un
mensaje informand

            @Override
            protected void onPreExecute() {
                  super.onPreExecute();
                  pDialog = new ProgressDialog(TablonNoticiasActivity.this);
                  pDialog.setMessage("Cargando Noticias..");
                  pDialog.setIndeterminate(false);
                  pDialog.setCancelable(false);
                  pDialog.show();
            }

            // obtendremos todas las noticias a traves de la url

            protected String doInBackground(String... args) {
                  //parametros de construccion
            List<NameValuePair> params = new ArrayList<NameValuePair>();
      // obtenemos la cadena JSON de la URL url_tablon_noticias_php

JSONObject json = jParser.makeHttpRequest(url_tablon_noticias_php, "GET",
params);

                  // revisaremos el logcat para respuestas JSON

                  Log.d("Tablon de noticias: ", json.toString());
```

```
                    try {
                          // comprobamos la Tag de exito , si es 1 exito
noticia encontrada y obtendremos el array de noticias
                          int success = json.getInt(TAG_SUCCESS);

                          if (success == 1) {

                                noticias = json.getJSONArray(TAG_NOTICIAS);

                                // bucle para recorrer todas nas noticias

                                for (int i = 0; i < noticias.length(); i++) {
                                JSONObject c = noticias.getJSONObject(i);

// Almacenaremos cada elemento que nos interesa que aparezca en la variable

                          String id = c.getString(TAG_PID);
                          String name = c.getString(TAG_NAME);
                          String curso = c.getString(TAG_CURSO);
                          String accion = c.getString(TAG_ACCION);
                          String asignaturas = c.getString(TAG_ASIGNATURAS);


                                      // creamos un nuevo hashmap
             HashMap<String, String> map = new HashMap<String, String>();

                                      //añadir cada nodo hijo a hashmap
clave->valor
                                      map.put(TAG_PID, id);
                                      map.put(TAG_NAME, name);
                                      map.put(TAG_CURSO, curso);
                                      map.put(TAG_ACCION, accion);
                                      map.put(TAG_ASIGNATURAS, asignaturas);

                                      noticeslist.add(map);
                                }
                          } else {
             //no hay noticias, nos envia a la actividad CrearNoticiaActivity
                                Intent i = new
Intent(getApplicationContext(),CrearNoticiaActivity.class);
                                // cerramos actividades anteriores
                                i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                startActivity(i);
                          }
                    } catch (JSONException e) {
                          e.printStackTrace();
                    }

                    return null;
             }

       // tras completarse la tarea en segundo plano descartamos el progreso de
dialogo onPreExecute()

             protected void onPostExecute(String file_url) {
                    pDialog.dismiss();
             //actualizamos la ui Cinterfaz de usuario) con los datos
analizados JSON en listview
                    runOnUiThread(new Runnable() {
                          public void run() {
                                ListAdapter adapter = new
SimpleAdapter(TablonNoticiasActivity.this, noticeslist,R.layout.list_item, new
String[] { TAG_PID,TAG_NAME,TAG_CURSO,TAG_ACCION,TAG_ASIGNATURAS},
                    new int[] { R.id.pid, R.id.name, R.id.curso, R.id.accion,
R.id.asignatura });
                                // actualizamos el listview
                                setListAdapter(adapter);
                          }
```

```
                });

            }

        }
}
```

VerNotiviasActivity.java

```
package com.example.interbooks;

import java.util.ArrayList;
import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

public class VerNoticiaActivity extends Activity {

        TextView txtName;
        TextView txtCurso;
        TextView txtAsignaturas;
        TextView txtAccion;
        TextView txtContacto;
        TextView txtDesc;
        TextView txtCreatedAt;


        String pid;

        private ProgressDialog pDialog;

        JSONParser jsonParser = new JSONParser();

        // ver detalles de la noticia
        private static final String url_detalles_noticia_php =
"http://10.0.2.2/android_connect/get_detalles_noticia.php";


        // JSON Node names
        private static final String TAG_SUCCESS = "success";
        private static final String TAG_NOTICIA = "noticia";
        private static final String TAG_PID = "pid";
        private static final String TAG_NAME = "name";
        private static final String TAG_CURSO = "curso";
        private static final String TAG_ASIGNATURAS = "asignaturas";
        private static final String TAG_ACCION = "accion";
        private static final String TAG_CONTACTO = "contacto";
        private static final String TAG_DESCRIPTION = "description";

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.ver_detalles_noticia);

                // obtenemos la intencion que a comenzado la actividad
                Intent i = getIntent();
```

```
// Obtenemos la identificación de la noticia (PID) de la intención
            pid = i.getStringExtra(TAG_PID);

// comenzamos la obtencion de los detalles de la noticia en un segundo plano
            new VerDetallesNoticia().execute();


      }
       // Tarea en segundo plano AsyncTask para obtener todos los detalles de
la noticia

      class VerDetallesNoticia extends AsyncTask<String, String, String> {


            @Override
            protected void onPreExecute() {
                  super.onPreExecute();
                  pDialog = new ProgressDialog(VerNoticiaActivity.this);
                  pDialog.setMessage("Cargando detalles de la noticia");
                  pDialog.setIndeterminate(false);
                  pDialog.setCancelable(true);
                  pDialog.show();
            }


              // Obtenemos los detalles en segundo plano

            protected String doInBackground(String... params) {

// //actualizamos la ui Cinterfaz de usuario) a partir de la tarea en segundo
plano

                  runOnUiThread(new Runnable() {
                        public void run() {
                              int success;
                              try {
                        // Construimos los parametros que vamos a comprobar
            List<NameValuePair> params = new ArrayList<NameValuePair>();
                        params.add(new BasicNameValuePair("pid", pid));

      //  Obtenemos los detalles del producto haciendo una petición HTTP
                        JSONObject json =
jsonParser.makeHttpRequest(url_detalles_noticia_php, "GET", params);
                        Log.d("detalles de la noticia", json.toString());

                        // json success tag
                                    success = json.getInt(TAG_SUCCESS);
                                          if (success == 1) {
      JSONArray noticiaObj = json.getJSONArray(TAG_NOTICIA); // JSONArray

                              //  obrenemos la primera (0) del JSONArray
                                 JSONObject notice = noticiaObj.getJSONObject(0);

                                    //noticias con esa
identificacion(pid)encontrados

            txtName = (TextView) findViewById(R.id.putName);
            txtCurso = (TextView) findViewById(R.id.putCurso);
            txtAsignaturas = (TextView) findViewById(R.id.putAsignaturas);
            txtAccion = (TextView) findViewById(R.id.putAccion);
            txtContacto = (TextView) findViewById(R.id.putContacto);
            txtDesc = (TextView) findViewById(R.id.putDesc);

      // Datos de la noticia que mostraremos en un TextView
      txtName.setText(notice.getString(TAG_NAME));
      txtCurso.setText(notice.getString(TAG_CURSO));

      txtAsignaturas.setText(notice.getString(TAG_ASIGNATURAS));
```

```
        txtAccion.setText(notice.getString(TAG_ACCION));

        txtContacto.setText(notice.getString(TAG_CONTACTO));

        txtDesc.setText(notice.getString(TAG_DESCRIPTION));

                                        }else{

        }
                                } catch (JSONException e) {
                                        e.printStackTrace();
                                }
                        }
                });

                return null;
        }



        protected void onPostExecute(String file_url) {
                pDialog.dismiss();
        }
    }

}
```

ViewHelpActivity.java

```
package com.example.interbooks;

import android.app.Activity;
import android.os.Bundle;

public class ViewHelpActivity extends Activity{



    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.view_help);



        }



}
```

Librerías

DatabaseHandler.java

```
package com.example.interbooks.library;

import java.util.HashMap;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

//servirá para acceder a los datos almacenados de nuestro usuario.
```

```
public class DatabaseHandler extends SQLiteOpenHelper {


	private static final int DATABASE_VERSION = 1;
	//nombre BD
	private static final String DATABASE_NAME = "android_api";
    //nombre tabla
	private static final String TABLE_LOGIN = "login";

	//nombres de las columnas de la tabla
	private static final String KEY_ID = "id";
	private static final String KEY_NAME = "name";
	private static final String KEY_EMAIL = "email";
	private static final String KEY_UID = "uid";
	private static final String KEY_CREATED_AT = "created_at";

	public DatabaseHandler(Context context) {
		super(context, DATABASE_NAME, null, DATABASE_VERSION);
	}

	// En el onCreate incluimos la creación de la tabla de usuarios
	//porque todas deben de crearse a la vez
	@Override




public void onCreate(SQLiteDatabase db) {
		String CREATE_LOGIN_TABLE = "CREATE TABLE " + TABLE_LOGIN + "("
				+ KEY_ID + " INTEGER PRIMARY KEY,"
				+ KEY_NAME + " TEXT,"
				+ KEY_EMAIL + " TEXT UNIQUE,"
				+ KEY_UID + " TEXT,"
				+ KEY_CREATED_AT + " TEXT" + ")";
		db.execSQL(CREATE_LOGIN_TABLE);
	}

	// Upgrading database
	@Override
	public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
		// Drop older table if existed
		db.execSQL("DROP TABLE IF EXISTS " + TABLE_LOGIN);

		// Create tables again
		onCreate(db);
	}

    //insertará el usuario logueado en la base de datos

	public void addUser(String name, String email, String uid, String
created_at) {
		SQLiteDatabase db = this.getWritableDatabase();

		ContentValues values = new ContentValues();
		values.put(KEY_NAME, name);
		values.put(KEY_EMAIL, email);
		values.put(KEY_UID, uid);
		values.put(KEY_CREATED_AT, created_at);

		db.insert(TABLE_LOGIN, null, values);
		db.close(); // cerramos la conexion con la base de datos
	}

	 //para que nos devuelva los datos del usuario

	public HashMap<String, String> getUserDetails(){
```

```
            HashMap<String,String> user = new HashMap<String,String>();
            String selectQuery = "SELECT  * FROM " + TABLE_LOGIN;

         SQLiteDatabase db = this.getReadableDatabase();
         Cursor cursor = db.rawQuery(selectQuery, null);
         // vamos a la primera fila
         cursor.moveToFirst();
         if(cursor.getCount() > 0){
               user.put("name", cursor.getString(1));
               user.put("email", cursor.getString(2));
               user.put("uid", cursor.getString(3));
               user.put("created_at", cursor.getString(4));
         }
         cursor.close();
         db.close();
               // return user
               return user;
      }

      //cuenta el número de filas que hay en la tabla login para saber si hay
alguien conectado
      public int getRowCount() {
            String countQuery = "SELECT  * FROM " + TABLE_LOGIN;
            SQLiteDatabase db = this.getReadableDatabase();
            Cursor cursor = db.rawQuery(countQuery, null);
            int rowCount = cursor.getCount();
            db.close();
            cursor.close();

            // return row count
            return rowCount;
      }

       //elimina los datos del usuario previamente almacenados en esta tabla
      public void resetTables(){
            SQLiteDatabase db = this.getWritableDatabase();
            // Delete All Rows
            db.delete(TABLE_LOGIN, null, null);
            db.close();
      }

}
```

UserFunctions.java

```
package com.example.interbooks.library;

import java.util.ArrayList;
import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONObject;

import android.content.Context;

public class UserFunctions {

      private JSONParser jsonParser;

      private static String loginURL = "http://10.0.2.2/logreg/";
      private static String registerURL = "http://10.0.2.2/logreg/";

      private static String login_tag = "login";
      private static String register_tag = "register";

      // constructor
      public UserFunctions(){
```

```
            jsonParser = new JSONParser();
      }

      //solicitud de loqueo de usuario, parametros (email y password)

      public JSONObject loginUser(String email, String password){

            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair("tag", login_tag));
            params.add(new BasicNameValuePair("email", email));
            params.add(new BasicNameValuePair("password", password));

            JSONObject json = jsonParser.getJSONFromUrl(loginURL, params);
            // return json
            return json;
      }

      //solicitud de registro de usuario, parametros (name,email y password)

      public JSONObject registerUser(String name, String email, String
password){
            // Building Parameters
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair("tag", register_tag));
            params.add(new BasicNameValuePair("name", name));
            params.add(new BasicNameValuePair("email", email));
            params.add(new BasicNameValuePair("password", password));

            // getting JSON Object
            JSONObject json = jsonParser.getJSONFromUrl(registerURL, params);
            // return json
            return json;
      }


      // comprueba en nuestra aplicación si ya hay un usuario registrado

      public boolean isUserLoggedIn(Context context){
            DatabaseHandler db = new DatabaseHandler(context);
            int count = db.getRowCount();
            if(count > 0){
                  // user logged in
                  return true;
            }
            return false;
      }

//elimina la información almacenada de un usuario en la BD del dispositivo
      public boolean logoutUser(Context context){
            DatabaseHandler db = new DatabaseHandler(context);
            db.resetTables();
            return true;
      }

}
```

PHP

Logreg

Config.php

```
<?php

define("DB_HOST", "localhost");
define("DB_USER", "root");
```

```
define("DB_PASSWORD", "");
define("DB_DATABASE", "logreg");
?>
```

DB_Connect.php

```
<?php

class DB_Connect {

    function __construct() {

    }

    function __destruct() {
        $this->close();
    }

    public function connect() {
        require_once 'config.php';
        $con = mysql_connect(DB_HOST, DB_USER, DB_PASSWORD);
        mysql_select_db(DB_DATABASE);

        return $con;
    }

    public function close() {
        mysql_close();
    }

}

?>
```

DB_Functions.php

```
<?php

class DB_Functions {

    private $db;


    function __construct() {
        require_once 'DB_Connect.php';
        $this->db = new DB_Connect();
        $this->db->connect();
    }

    function __destruct() {

    }

    /**
     * Guardamos nuevos usuarios
     * Devolvemos detalle del usuario
     */
    public function storeUser($name, $email, $password) {
        $uuid = uniqid('', true);
        $hash = $this->hashSSHA($password);
        $encrypted_password = $hash["encrypted"]; // password encriptada
        $salt = $hash["salt"]; // salt
        $result = mysql_query("INSERT INTO users(unique_id, name, email,
encrypted_password, salt, created_at) VALUES('$uuid', '$name', '$email',
'$encrypted_password', '$salt', NOW())");
```

```
        // comprobamos que la query se ejecutó correctamente
        if ($result) {
            // obtener detalle del usuario
            $uid = mysql_insert_id();
            $result = mysql_query("SELECT * FROM users WHERE uid = $uid");
            // devolvemos detalle del usuario
            return mysql_fetch_array($result);
        } else {
            return false;
        }
    }

    /**
     * Obtenemos usuario por email y contraseña
     */
    public function getUserByEmailAndPassword($email, $password) {
        $result = mysql_query("SELECT * FROM users WHERE email = '$email'") or
die(mysql_error());
        // verificamos el resultado de la query
        $no_of_rows = mysql_num_rows($result);
        if ($no_of_rows > 0) {
            $result = mysql_fetch_array($result);
            $salt = $result['salt'];
            $encrypted_password = $result['encrypted_password'];
            $hash = $this->checkhashSSHA($salt, $password);

            if ($encrypted_password == $hash) {
                // Se ha autenticado al usuario correctamente
                return $result;
            }
        } else {
            // usuario no encontrado
            return false;
        }
    }

    /**
     * Comprobamos si el usuario existe o no
     */
    public function isUserExisted($email) {
        $result = mysql_query("SELECT email from users WHERE email =
'$email'");
        $no_of_rows = mysql_num_rows($result);
        if ($no_of_rows > 0) {
            // usuario existe
            return true;
        } else {
            // usuario no existe
            return false;
        }
    }

    /**
     * Encriptamos la password
     * devolvemos el salt y la password encriptada
     */
    public function hashSSHA($password) {

        $salt = sha1(rand());
        $salt = substr($salt, 0, 10);
        $encrypted = base64_encode(sha1($password . $salt, true) . $salt);
        $hash = array("salt" => $salt, "encrypted" => $encrypted);
        return $hash;
    }

    /**
     * Desencriptamos la password
     * Devolvemos una hash string
```

```
     */
    public function checkhashSSHA($salt, $password) {

        $hash = base64_encode(sha1($password . $salt, true) . $salt);

        return $hash;
    }

}

?>
```

Index.php

```
<?php
  /**
 * Comprobamos la petición POST
 */
if (isset($_POST['tag']) && $_POST['tag'] != '') {
    // obtenemos tag
    $tag = $_POST['tag'];

    require_once 'DB_Functions.php';
    $db = new DB_Functions();

    // Empezamos a preparar la respuesta en forma de Array que luego
convertiremos en un json
    $response = array("tag" => $tag, "success" => 0, "error" => 0);

    // evaluamos tag
    if ($tag == 'login') {
        // chequeamos el login
        $email = $_POST['email'];
        $password = $_POST['password'];

        // obtenemos y comprobamos el usuario por email y password
        $user = $db->getUserByEmailAndPassword($email, $password);
        if ($user != false) {
            // usuario encontrado
            // marcamos el json como correcto con success = 1
            $response["success"] = 1;
            $response["uid"] = $user["unique_id"];
            $response["user"]["name"] = $user["name"];
            $response["user"]["email"] = $user["email"];
            $response["user"]["created_at"] = $user["created_at"];
            $response["user"]["updated_at"] = $user["updated_at"];
            echo json_encode($response);
        } else {
            // usuario no encontrado
            // marcamos el json con error = 1
            $response["error"] = 1;
            $response["error_msg"] = "Email o password incorrecto!";
            echo json_encode($response);
        }
    } else if ($tag == 'register') {
        // Registrar nuevo usuario
        $name = $_POST['name'];
        $email = $_POST['email'];
        $password = $_POST['password'];

        // comprobamos si ya existe el usuario
        if ($db->isUserExisted($email)) {
            // usuario ya existe - marcamos error 2
            $response["error"] = 2;
            $response["error_msg"] = "Usuario ya existe";
```

```
            echo json_encode($response);
        } else {
            // guardamos el nuevo usuario
            $user = $db->storeUser($name, $email, $password);
            if ($user) {
                // usuario guardado correctamente
                $response["success"] = 1;
                $response["uid"] = $user["unique_id"];
                $response["user"]["name"] = $user["name"];
                $response["user"]["email"] = $user["email"];
                $response["user"]["created_at"] = $user["created_at"];
                $response["user"]["updated_at"] = $user["updated_at"];
                echo json_encode($response);
            } else {
                // fallo en la inserción del usuario
                $response["error"] = 1;
                $response["error_msg"] = "Ocurrió un error durante el
registro";
                echo json_encode($response);
            }
        }
    } else {
        echo "Petición no válida";
    }
} else {
    echo "Acceso denegado";
}
?>
```

android_connect

actualizar_noticia.php

```
<?php


$response = array();

if (isset($_POST['pid']) && isset($_POST['name']) && isset($_POST['curso']) &&
isset($_POST['asignaturas']) && isset($_POST['accion']) &&
isset($_POST['contacto']) && isset($_POST['description'])) {

    $pid = $_POST['pid'];
    $name = $_POST['name'];
    $curso = $_POST['curso'];
       $asignaturas = $_POST['asignaturas'];
       $accion = $_POST['accion'];
       $contacto = $_POST['contacto'];
    $description = $_POST['description'];

    require_once __DIR__ . '/db_connect.php';

    $db = new DB_CONNECT();

    $result = mysql_query("UPDATE products SET name = '$name', curso =
'$curso', asignaturas = '$asignaturas', accion = '$accion', contacto =
'$contacto', description = '$description' WHERE pid = $pid");

    if ($result) {
        $response["success"] = 1;
        $response["message"] = "noticia actualizada correctamente.";

        echo json_encode($response);
    } else {

    }
```

```
} else {

    $response["success"] = 0;
    $response["message"] = " Campo(s) necesarios perdido(s)";

    echo json_encode($response);
}
?>
```

borrar_noticia.php

```
<?php


$response = array();

if (isset($_POST['pid'])) {
    $pid = $_POST['pid'];

    require_once __DIR__ . '/db_connect.php';

    $db = new DB_CONNECT();

    $result = mysql_query("DELETE FROM products WHERE pid = $pid");

    if (mysql_affected_rows() > 0) {

        $response["success"] = 1;
        $response["message"] = "noticia borrada correctamente";

        echo json_encode($response);

    } else {

        $response["success"] = 0;
        $response["message"] = "noticia no encontrada";

        echo json_encode($response);
    }
} else {

    $response["success"] = 0;
    $response["message"] = "Campo(s) necesarios perdido(s)";

    echo json_encode($response);
}
?>
```

Crear_noticia.php

```
<?php
/*
 * creara una nueva noticia en la base de datos con todos sus campos
(name,curso,asignaturas,accion,contacto y descripcion)
 */
$response = array();

if (isset($_POST['name']) && isset($_POST['curso']) &&
isset($_POST['asignaturas']) && isset($_POST['accion']) &&
isset($_POST['contacto']) && isset($_POST['description'])) {

    $name = $_POST['name'];
    $curso = $_POST['curso'];
       $asignaturas = $_POST['asignaturas'];
       $accion = $_POST['accion'];
       $contacto = $_POST['contacto'];
```

```
    $description = $_POST['description'];

    require_once __DIR__ . '/db_connect.php';

    $db = new DB_CONNECT();

    $result = mysql_query("INSERT INTO products(name, curso, asignaturas,
accion, contacto, description) VALUES('$name', '$curso', '$asignaturas',
'$accion', '$contacto', '$description')");

    if ($result) {

        $response["success"] = 1;
        $response["message"] = "Noticia creada correctamente.";

        echo json_encode($response);
    } else {

        $response["success"] = 0;
        $response["message"] = "ERROR.";

        echo json_encode($response);
    }
} else {

    $response["success"] = 0;
    $response["message"] = " Campo(s) necesarios perdido(s)";

    echo json_encode($response);
}
?>
```

Get_detalles_noticia.php

```
<?php

$response = array();

require_once __DIR__ . '/db_connect.php';

$db = new DB_CONNECT();

if (isset($_GET["pid"])) {
    $pid = $_GET['pid'];

    $result = mysql_query("SELECT *FROM products WHERE pid = $pid");

    if (!empty($result)) {
        if (mysql_num_rows($result) > 0) {

            $result = mysql_fetch_array($result);

            $notice = array();
            $notice["pid"] = $result["pid"];
            $notice["name"] = $result["name"];
            $notice["curso"] = $result["curso"];
             $notice["asignaturas"] = $result["asignaturas"];
             $notice["accion"] = $result["accion"];
             $notice["contacto"] = $result["contacto"];
            $notice["description"] = $result["description"];
            $notice["created_at"] = $result["created_at"];
            $notice["updated_at"] = $result["updated_at"];

            $response["success"] = 1;
            $response["noticia"] = array();

            array_push($response["noticia"], $notice);
```

```
            echo json_encode($response);
        } else {
            $response["success"] = 0;
            $response["message"] = "Noticia no encontrada";
            echo json_encode($response);
        }
    } else {
        $response["success"] = 0;
        $response["message"] = "Noticia no encontrada";
        echo json_encode($response);
    }
} else {
    $response["success"] = 0;
    $response["message"] = " Campo(s) necesarios perdido(s)";
    echo json_encode($response);
}
?>
```

Get_notices.php

```
<?php

// permite formar una lista con todas las noticias

$response = array();

require_once __DIR__ . '/db_connect.php';

// connecting to db
$db = new DB_CONNECT();

//optiene todas las noticias de la tabla product
$result = mysql_query("SELECT *FROM products") or die(mysql_error());

// check for empty result
if (mysql_num_rows($result) > 0) {


    $response["noticias"] = array();

    while ($row = mysql_fetch_array($result)) {

        $noticia = array();
        $noticia["pid"] = $row["pid"];
        $noticia["name"] = $row["name"];
        $noticia["curso"] = $row["curso"];
         $noticia["asignaturas"] = $row["asignaturas"];
         $noticia["accion"] = $row["accion"];
        $noticia["description"] = $row["description"];
        $noticia["created_at"] = $row["created_at"];
        $noticia["updated_at"] = $row["updated_at"];

        array_push($response["noticias"], $noticia);
    }
    $response["success"] = 1;

    echo json_encode($response);
} else {

    $response["success"] = 0;
    $response["message"] = "No notices found";

    // echo no users JSON
    echo json_encode($response);
}
?>
```

Bases de datos

```
CREATE TABLE products (
pid int(11) primary key auto_increment,
nombre varchar(100) not null,
curso varchar(100) not null,
asignatura varchar(100) not null,
accion varchar(10) not null,
contacto varchar(30) not null,
descripcion text,
created_at timestamp default now(),
updated_at timestamp
);

create table users(
   uid int(11) primary key auto_increment,
   unique_id varchar(30) not null unique,
   name varchar(60) not null,
   email varchar(100) not null unique,
   encrypted_password varchar(90) not null,
   salt varchar(10) not null,
   created_at datetime,
   updated_at datetime null
);
```