UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Diseño de una aplicación Android para la recogida de datos de tráfico y su comunicación a un servidor

Proyecto Final de Carrera

Ingeniería Técnica Informática de Gestión

Autor: Miguel Ángel Godoy Soro

Tutor: Juan Carlos Cano Escribá
Carlos Miguel Tavares Calafate

2014-2015

Resumen

Este proyecto consiste en el desarrollo de una aplicación Android para dispositivos móviles para almacenar y, posteriormente, utilizar los datos del tráfico proporcionados por los usuarios de la misma. De este modo, la aplicación desarrollada capta la localización y velocidad de los usuarios y la transmite a un servidor de bases de datos MySQL donde es almacenada para su posterior utilización a la hora de calcular la ruta más rápida entre dos o más puntos. Además, para hacer todo esto posible, se establece un modelo cliente-servidor que permite la creación de usuarios y su autentificación contra el servidor, de manera que los usuarios registrados podrán enviar sus datos al servidor de forma automática mientras estén utilizando la aplicación.

Esta funcionalidad se ha incluido en el desarrollo previo llevado a cabo por David Peris Martínez en su Trabajo de Fin de Grado titulado *Desarrollo de navegador para coches basado en Android y OpenStreetMap.*

Palabras clave: Aplicación, control de tráfico, Android, Java, Apache, MySQL, PHP.

Abstract

This project consists in the development of an Android application for mobile devices which allows storing and using the traffic data provided by the users of the application. This way, the developed application gets the user's location and speed and sends it to a MySQL database server, where it is stored for its further use when calculating the fastest route through two or several points. Besides, in order to make all of it possible, a server-client model has been established, which allows the register of users and their authentication with the server, thus allowing the registered users to send their traffic data to the server in an automatic way while they use the application.

All this development has been included in the application previously developed by David Peris Martínez in his Degree Final Paper named *Development of a car navigator based in Android and OpenStreetMap.*

Keywords: Application, traffic control, Android, Java, Apache, MySQL, PHP.

Tabla de contenidos

1. Introducción 5
1.1. Motivación 5
1.2. Objetivos 5
2. Aplicación cliente 6
2.1. Entorno de trabajo 6
2.1.1. Eclipse 6
2.1.2. Java 9
2.1.3. SDK Android 10
2.2. Librerías utilizadas 12
2.2.1. Android API 17 12
2.2.1.1. Activity 15
2.2.1.2. View 15
2.2.1.3. Intent 16
2.2.1.4. Bundle 16
2.2.1.5. Toast 16
2.2.1.6. SQLiteOpenHelper 17
2.2.1.7. Application 18
2.2.1.8. Geocoder 18
2.2.2. Volley 18
2.2.2.1. Request.Method 19
2.2.2.2. Response 19
2.2.2.3. Volley Error 19
2.2.2.4. toolbox.StringRequest 19
2.2.3. JSON 19
2.2.3.1. JSONException 20
2.2.3.2. JSONObject 20
2.3. Paquetes del proyecto 20
2.3.1. AndroidManifest.xml 21
2.3.2. Código fuente (src) 22
2.3.2.1. es.upv.disca.osmlcient.login 22
2.3.2.2. es.upv.disca.osmlcient.login.config 28
2.3.2.3. es.upv.disca.osmlcient.login.helper 28

2.3.2.4. es.upv.disca.osmclient.location 31
2.3.3. Carpeta de recursos (res) 33
2.3.3.1. Definición de vistas (layout) 33
2.3.3.2. Carpeta de imágenes e iconos (drawable) 36
2.3.3.3. Carpeta de animaciones (animator) 36
2.3.3.4. Definición de valores (values) 37
3. Servidor 39
3.1. WAMP 39
3.1.1. Funcionamiento 39
3.1.2. Estructura 40
3.2. MySQL 40
3.2.1. Users 42
3.2.2. Locations 43
3.3. PHP 44
3.3.1. Index.php 46
3.3.2. Include 47
3.3.2.1. Config.php 47
3.3.2.2. DB_Connect.php 48
3.3.2.3. DB_Functions.php 48
4. Pruebas y desarrollo 51
5. Conclusiones 53
6. Bibliografía 55
7. Anexos 57

1. Introducción

1.1. Motivación

Hoy en día, el mercado está lleno de aplicaciones que permiten el cálculo de rutas para moverse de manera eficiente por las calles de las ciudades con ayuda de mapas y la utilización del GPS. Sin embargo, la mayoría de estas no tienen en cuenta el tráfico y densidad de vehículos en las calles, por lo tanto, la ruta que proporcionan puede no ser la más rápida en un determinado momento del día.

Ejemplos de estas aplicaciones podrían ser por ejemplo Waze, Wisepilot, CoPILOT o NavFree.

Además, muchas de ellas requieren el acceso a internet, por ejemplo Google Maps, para cargar los mapas, por lo que los resultados ofrecidos pueden ser un poco lentos o tardar más de lo esperado en cargar.

Por todo ello, se ha decidido crear una aplicación con los mapas almacenados en local que permita el registro de usuarios que envíen sus datos de manera continua y en tiempo real para permitir una navegación más rápida y con menos coste en la tarifa de datos de los usuarios y que, a la vez, permita al servidor almacenar sus datos de navegación y velocidad para así utilizarlos para calcular rutas de manera eficiente teniendo en cuenta la densidad del tráfico.

1.2. Objetivos

Los objetivos principales que se pretenden alcanzar con el desarrollo de la aplicación son los siguientes:

- Crear una interfaz simple y sencilla, a la vez que vistosa que permita registrar usuarios en el servidor de base de datos aportando un nombre, un email y una contraseña válida. También deberá permitir iniciar sesión a los usuarios ya registrados e iniciar la navegación en el sistema de rutas.
- Implementar un sistema servidor de base de datos, basado en WAMP (Apache, MySQL y PHP en Windows), en el que se almacenen los usuarios creados desde la aplicación, así como sus datos de tráfico referentes a geoposición y velocidad en todo momento. La aplicación deberá enviar los datos referentes a su ubicación en la ciudad (población, calle, número), en qué fecha y hora han sido recogidos esos datos, y la velocidad de circulación en ese punto y en ese momento.
- Iniciar el sistema desarrollado previamente por el alumno David Peris Martínez en su TFG que permite el cálculo de rutas entre varios puntos, incorporando las nuevas funcionalidades de control de tráfico desarrolladas en este proyecto, que permitirán al servidor proporcionar rutas más rápidas dependiendo de los datos recibidos.

2. Aplicación cliente

En este apartado se presenta el proceso que he llevado a cabo para el desarrollo completo de la parte de la aplicación de Android de este proyecto. Para ello veremos el entorno de trabajo utilizado y las diferentes herramientas de programación utilizadas para su desarrollo, las librerías particulares que se han utilizado, así como sus clases más importantes y, finalmente, se repasará la estructura del proyecto que constituye la aplicación Android, viendo los recursos gráficos y el código fuente que permite el funcionamiento de la misma.

2.1. Entorno de trabajo

Para el desarrollo de la aplicación se ha utilizado el IDE (Integrated Development Environment o entorno de desarrollo integrado) Eclipse. Además, se ha utilizado el lenguaje de programación Java porque es el lenguaje necesario para desarrollar aplicaciones Android y se han utilizado una serie de librerías y clases adicionales a las nativas de java y Android que explicaré a continuación, con el fin de proveer a la aplicación de funcionalidades necesarias relativas a las conexiones con la bases de datos MySQL (volley), a la comunicación con el servidor php (JSON) y otras propias de la configuración y las bases de datos propias del sistema Android.

2.1.1. Eclipse

¿Por qué Eclipse y no Android Studio?

Esta pregunta se la realiza todo diseñador de aplicaciones nativas para Android tarde o temprano. Sin embargo, hay opiniones muy diversas y de todo tipo, aunque es realidad que últimamente la más predominante es la de realizar una transición definitiva a Android Studio, puesto que ya ha salido de su versión beta, superándola con éxito, y es un entorno muy agradable y que presenta ciertas ventajas frente a Eclipse. La primera de ellas es que es un entorno pensado y diseñado exclusivamente para el desarrollo de aplicaciones Android, y por eso, es mucho más sencillo e intuitivo de utilizar, sobre todo a la hora de importar y gestionar todo tipo de librerías, algo que en Eclipse puede provocar algún que otro quebradero de cabeza. Además, Eclipse ha dejado de recibir soporte y actualizaciones para Android, con lo cual dejará de ser útil y operativo tarde o temprano, como ya está sucediendo con el famoso y tan estable sistema operativo Windows XP.

Sin embargo, la transición de una aplicación realizada con Eclipse a Android Studio no es trivial, puesto que la estructura del sistema de ficheros del proyecto no es la misma ya que Android Studio incorpora los llamados fichero *gradle* que pueden suponer trabas a la hora de realizar la migración.

Así pues, como la aplicación realizada previamente fue diseñada en Eclipse, se va a seguir utilizando este sistema. No obstante, para futuros proyectos de aplicaciones Android que se inicien desde cero, recomendaría el uso de Android Studio, ya que es el IDE que va a recibir soporte a partir de ahora y permitirá aplicar el famoso y asombroso diseño "Material Design", que no es más que un patrón de diseño establecido por Google para dar a todas las aplicaciones Android desarrolladas bajo estos principios un toque moderno y sencillo, agradable para el usuario.

¿Qué es Eclipse?

Eclipse es un programa informático que está formado por distintas herramientas de programación de código abierto, multiplataforma, diseñado para crear aplicaciones de cliente enriquecido, opuestas a las llamadas aplicaciones de cliente liviano desarrolladas para navegadores. Esta plataforma se ha utilizado para desarrollar entornos de desarrollo integrados (del inglés, IDE), como el IDE Java Development Toolkit (JDT) o el compilador ECJ que se entrega como parte de Eclipse.

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

Originalmente, Eclipse fue liberado bajo la Common Public License, aunque poco después fue re-licenciado bajo la licencia Eclipse Public License. La Free Software Foundation ha dicho además que ambas licencias son licencias de software libre, pero incompatibles con la Licencia Pública General de GNU (GNU GPL).

De origen, Eclipse no proporciona grandes características ni funcionalidades al usuario final por sí mismo. Lo que hace de Eclipse un software tan recomendable son las plataformas que lo integran y que permiten un desarrollo rápido e integrado de características basado en un modelo de plug-ins.

Eclipse proporciona un modelo de interfaz (UI) común para trabajar con estas herramientas. Está diseñado para funcionar en múltiples sistemas operativos y provee una integración robusta con cada uno de ellos. Adicionalmente, los plug-ins se pueden programar y configurar para los API portables de Eclipse y correr también en todos los sistemas operativos.

En el núcleo de Eclipse, hay una arquitectura para el descubrimiento dinámico, carga y ejecución de plug-ins. La plataforma se encarga de manejar la lógica de encontrar y ejecutar el código correcto. La plataforma proporciona un modelo estándar de navegación. Por esta razón, cada plug-in puede centrarse en realizar un cierto número de pequeñas tareas (definición, pruebas, animación, publicación, compilación, depuración, realización de diagramas...) de manera eficiente.

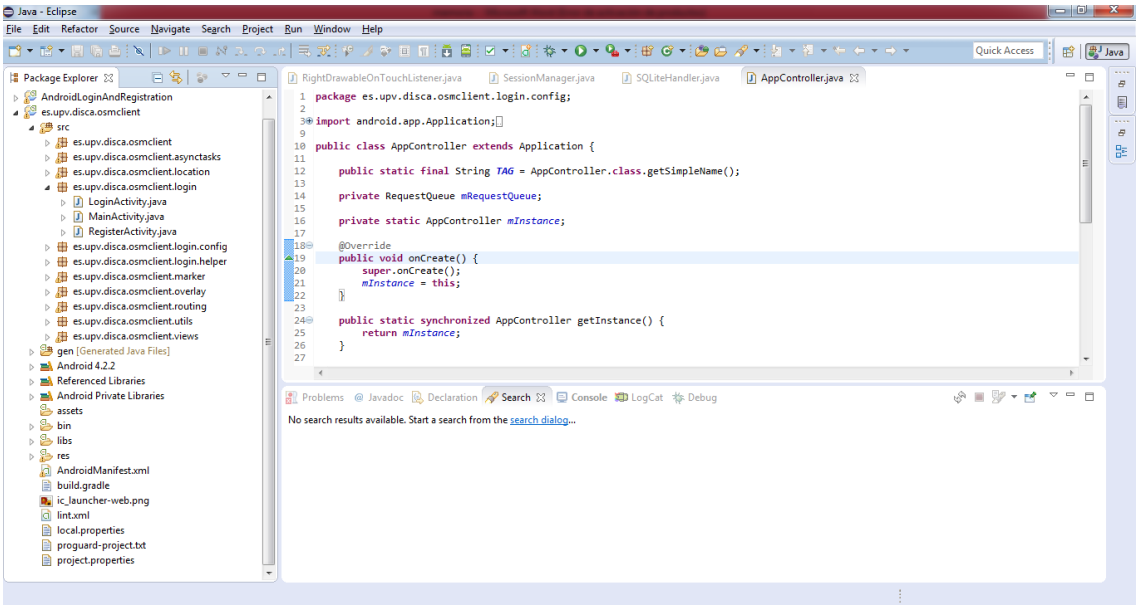

Imagen 1. Captura de pantalla IDE Eclipse

Especificaciones y funcionamiento

Para el desarrollo de la aplicación se ha utilizado la versión Luna (4.4) de Eclipse por ser la más estable en el momento de iniciar el proyecto. Cabe decir que actualmente ya se encuentra disponible la última versión (4.5), llamada Mars.

Dentro de Eclipse, los proyectos se estructuran en diferentes paquetes entre los que destacan el src, o paquete de código fuente, y el paquete res, donde se guardan los archivos xml que dan apariencia a las distintas pantallas. Sin embargo, profundizaré en este tema en un capítulo posterior.

Como se puede ver en la captura de pantalla anterior, las opciones son numerosas, y se pueden ampliar mediante la instalación de plug-ins. No obstante, las más importantes son el Android SDK Manager, para manejar los paquetes necesarios y las versiones Android para las que se puede programar, el AVD, para lanzar un emulador en el que probar las aplicaciones diseñadas, y las opciones para lanzar las aplicaciones. Para esto tenemos dos opciones, lanzar la aplicación en modo normal o en modo depuración. Además, mediante la opción Run -> Run Configurations... podemos establecer la configuración que deseamos utilizar cada vez que lancemos el proyecto.

Sin embargo, cabe destacar la gran utilidad de la opción de ejecutar en modo depuración (F11), puesto que permite depurar el código mediante la utilización de los llamados breakpoints y la ejecución del código línea por línea, permitiendo ver el estado de las variables y de la pila de ejecución en todo momento.

2.1.2. Java

¿Por qué Java?

Java es el único lenguaje de programación utilizado en el diseño de aplicaciones Android, puesto que C++ se utiliza a un nivel inferior para programar en código nativo. En este caso he utilizado el update 75 de la versión 7 de Java Runtime Environment (JRE), aunque ya está disponible la versión 8.

El proyecto Java

El proyecto del lenguaje de programación Java fue iniciado en junio de 1991 por James Gosling, Mike Sheridan y Patrick Naughton, de Sun Microsytems, Inc, ahora convertida en Oracle Corporation. Originalmente fue pensado para la televisión interactiva, pero era demasiado avanzado para la tecnología que utilizaba la televisión por cable en ese momento. Gosling diseñó el lenguaje con un estilo y sintaxis similiar al de C/C++, para que los programadores de aplicaciones lo encontrasen familiar.

¿Qué es Java?

Java es un lenguaje de programación de propósito general que es concurrente, basado en clases, orientado a objetos y específicamente diseñado para tener tan pocas dependencias de implementación como sea posible. Sigue la filosofía WORA (write once, run anywhere), lo que significa que está pensado para poder ser ejecutado en cualquier plataforma y/o dispositivo que soporte Java, sin necesidad de ser recompilado. Esto es posible gracias a que las aplicaciones Java suelen ser compiladas a un bytecode y no a un ejecutable, por lo que puede ejecutarse en cualquier máquina virtual Java (Java Virtual Machine, JVM), sin importar la arquitectura de computación.

En la actualidad es uno de los lenguajes de programación más utilizados, gracias particularmente a su uso en aplicaciones web cliente-servidor, con un total estimado de 9 millones de desarrolladores.

Inicialmente, en 1995, Sun lanzó las referencias de implementación, compiladores, máquinas virtuales y librerías de clase bajo licencias propietarias. Sin embargo, en mayo de 2007, de acuerdo con las especificaciones de la Java Community Process, Sun relicenció la mayoría de tecnología Java bajo la licencia GNU General Public License (GNU GPL).

Los principios bajo los que se desarrolló el lenguaje fueron los siguientes:

- Debe ser simple, orientado a objetos y familiar.
- Debe ser robusto y seguro.
- Debe tener una arquitectura neutra y portable.
- Debe ejecutarse con alto rendimiento.
- Debe ser interpretado, dinámico y permitir hilos.

Portabilidad

La portabilidad es una de los principales objetivos de Java, lo cual significa que los programas diseñados en Java deben ejecutarse de manera similar en cualquier combinación de hardware y sistema operativo. Como ya se ha dicho anteriormente, esto se consigue compilando el código a una representación intermedia llamada Java bytecode, en lugar de a un código máquina específico de la arquitectura. Las instrucciones del bytecode son análogas a las del código máquina, pero pensadas para ser ejecutadas en una máquina virtual, la cual sí que se ha escrito específicamente para el hardware destino. Los usuarios finales, normalmente utilizan una herramienta conocida como Java Runtime Environment (JRE) instalada en su propia máquina o en los navegadores web.

2.1.3. SDK Android

¿Qué es el SDK de Android?

El SDK (Software Development Kit) es un kit de desarrollo de software que nos permitirá el desarrollo de aplicaciones Android y ejecutar un emulador del sistema Android de cualquier versión desde el ordenador.

Cada versión de Android diseñada por Google viene acompañada de su propio SDK, que incluye el depurador de código, una biblioteca de librerías, un emulador de la versión, ejemplos de código, documentación y tutoriales. Para poder utilizarlos desde Eclipse será necesario utilizar el plug-in conocido como ADT (Android Development Tools).

Como ya se ha comentado, las actualizaciones del SDK están coordinadas con el desarrollo general de Android. De esta manera, el SDK soporta también versiones antiguas de Android, en caso de que se necesite testear o instalar aplicaciones en dispositivos obsoletos. Las herramientas de desarrollo son componentes descargables, de modo que aunque se instale la última versión, es compatible con instalaciones de versiones anteriores.

Eclipse dispone de una opción para abrir lo que se conoce como Android SDK Manager, un asistente para la descarga e instalación de los diferentes SDK de Android y otras herramientas útiles para el desarrollo de aplicaciones.

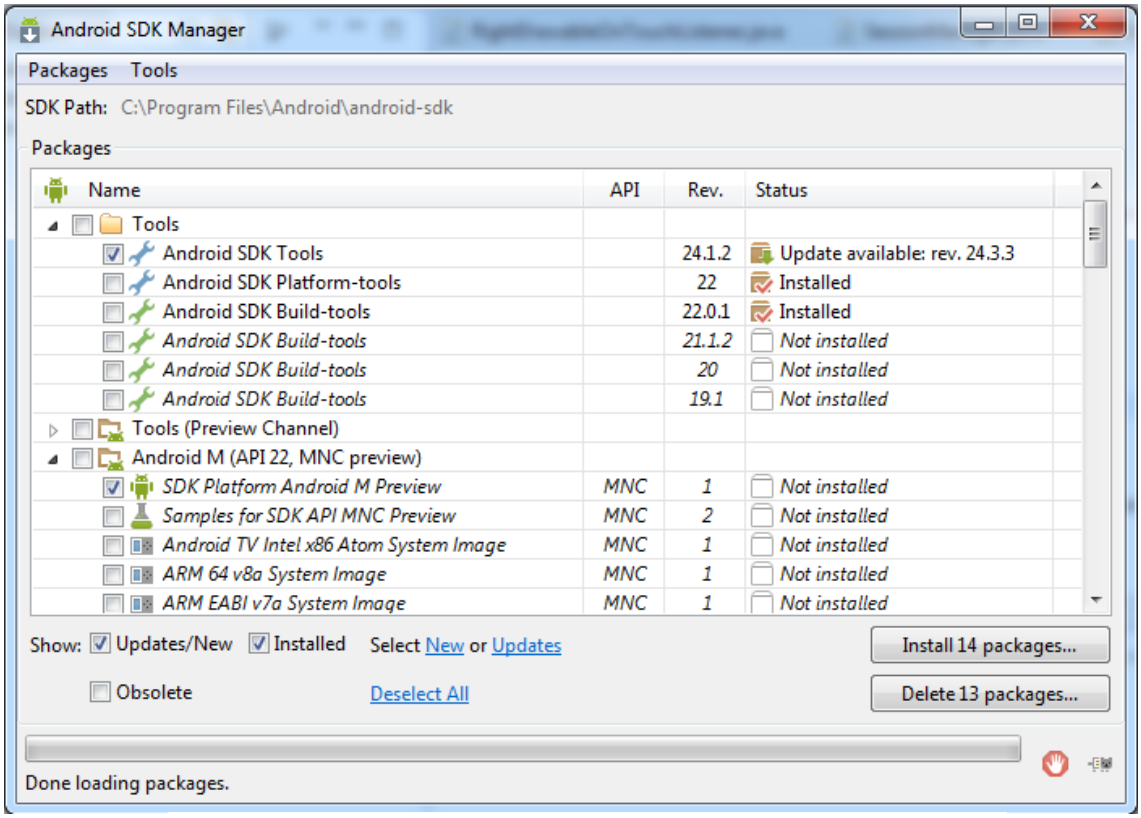

Imagen 2. Captura de pantalla de Android SDK Manager

Android Virtual Device

El emulador incluido en el SDK, también conocido como AVD (Android Virtual Device), es capaz de emular tanto dispositivos reales (seleccionándolos de una lista) como dispositivos con las características que el usuario indique (RAM, procesador, memoria interna, resolución de pantalla, tamaño...).

El usuario puede crear tantos dispositivos como quiera y puede lanzar su aplicación en varios a la vez para testear diferencias de comportamiento. Sin embargo, también se puede depurar y probar las aplicaciones sobre dispositivos reales, lo cual es más recomendable a la hora de desarrollar código puesto que la respuesta del terminal será inmediata y los emuladores suelen ser un poco lentos. No obstante, los emuladores son ideales cuando la aplicación es estable y se quiere testear su funcionamiento en dispositivos con diferentes tamaños y resoluciones de pantalla.

Imagen 3. Captura de pantalla de Android Virtual Device

2.2. Librerías utilizadas

Además de las librerías propias de Android, de las cuales se ha utilizado la versión 17, se han utilizado también las librerías *volley* para la utilización de llamadas al servidor de base de datos MySQL desde Android y las librerías JSON para el intercambio de información entre el servidor y la aplicación cliente. Todas ellas se desarrollan a continuación.

2.2.1. Android API 17

El nivel 17 del API de Android supone una actualización de la versión Jelly Bean que ofrece nuevas características y herramientas para usuarios y desarrolladores.

Google esperaba lanzar esta versión en octubre de 2012, sin embargo, el acto fue cancelado debido al huracán Sandy. No hubo acto después de eso, sino que Google decidió publicar esta versión con una nota de prensa posterior. Este nivel del API supondría la versión 4.2 de Android, conocida como Jelly Bean, que está basada en el kernel 3.4.0 de Linux. Los primeros terminales en

incorporar esta versión de manera nativa fueron el Nexus 4 y el Nexus 10, en noviembre de 2012.

Antes del cambio a la versión 4.3, el último desarrollo de la 4.2 supuso la versión 4.2.2, publicada en febrero de 2013.

Conceptos

Cualquier aplicación diseñada para Android está constituida visualmente por actividades, fragments, vistas y servicios, entre otros. Las actividades son el motor principal de las aplicaciones, representan cada una de las pantallas de la aplicación, lo que sería equivalente a las ventanas de cualquier otro sistema operativo. Las actividades están siempre asociadas a un layout o contenedor. Los layouts son ficheros xml que representan la parte visual de la pantalla, mientras que la funcionalidad de la misma se determina en una clase java.

Los fragments fueron introducidos en la versión 3.0 de Android, junto con el lanzamiento del sistema operativo para tablets. Su principal función era permitir la reutilización de código y la adaptación de las aplicaciones a los nuevos tamaños de pantalla que venían de la mano con los nuevos dispositivos Tablet, más grandes y con nuevas resoluciones.

Las vistas equivalen a los controles de usuario de, por ejemplo, Windows Forms. Es decir, representan todos los elementos visuales de la aplicación (botones, labels, cajas de texto, listas, tablas, imágenes…).

Los servicios son clases java sin representación gráfica cuya finalidad es llevar a cabo tareas, normalmente en segundo plano, que deben estar continuamente en ejecución, como por ejemplo, estar atentos a la recepción de notificaciones.

El ciclo de vida de las actividades y las aplicaciones en Android

Como ya se ha dicho, una aplicación está formada principalmente por un conjunto de actividades. Las actividades son las que controlan el ciclo de vida de la aplicación, dado que los usuarios cambian entre actividades, no entre aplicaciones, mientras navegan por la aplicación.

Una aplicación Android se ejecuta dentro de su propio proceso Linux, que es creado por la aplicación y seguirá vivo hasta que ya no sea requerido y el sistema necesite su memoria para alguna otra tarea. Una característica que se deriva de esto es que no es la aplicación la que controla cuando se destruye el proceso, sino el propio sistema, basándose en el conocimiento que tiene del estado de la aplicación. Si el sistema destruye el proceso, el usuario podrá volver a la aplicación, pero habrá perdido el estado en el que la había dejado la última vez. En estos casos, es responsabilidad del programador almacenar el estado de las actividades si quiere conservar el estado.

De todo esto se deriva la importancia de manejar correctamente el ciclo de vida de las actividades en Android. Por tanto, es necesario comprender y manejar los eventos relacionados con el ciclo de vida para construir aplicaciones estables.

- Activa (running): La aplicación está encima de la pila, es decir, se está ejecutando, es visible y tiene el foco del sistema.
- Visible (paused): La actividad es visible pero no tiene el foco. Este estado se alcanza cuando otra actividad no visible o que no ocupa toda la pantalla pasa a tener el foco. Si pasase a estar tapada por completo pasaría a…
- Parada (stopped): La actividad no es visible. Se deberá guarder el estado de la interfaz de usuario y de las variables, preferencias…
- Destruida (destroyed): La aplicación llama a su método finish() o el sistema la destruye para liberar memoria para otras tareas.

Cuando la aplicación pasa de un estado a otros se lanzan una serie de eventos que el programador debe controlar para manejar correctamente el ciclo de vida de las actividades.

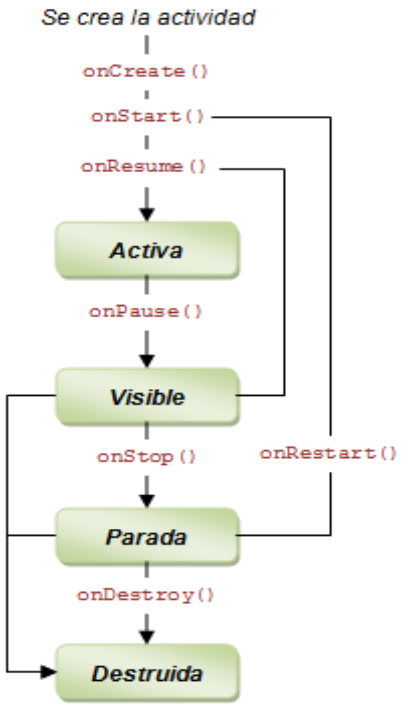

Imagen 4. Ciclo de vida de las activiades en Android

- onCreate(Bundle): Se llama en la creación de la actividad y se utiliza para realizar la inicialización de todas las variables, la interfaz de usuario y la estructura de datos. Puede recibir información a través de un objeto de tipo Bundle, por si se llama desde otra actividad o por si ha sido destruida y vuelta a crear.
- onStart(): Se lanza cuando la actividad es mostrada al usuario.

- onResume(): Se llama cuando la aplicación va a empezar a interaccionar con el usuario. Es un buen momento para inicializar las animaciones, la música, etc.
- onPause(): Es llamado cuando la actividad está a punto de ser movida a segundo plano, ya sea porque se ha lanzado otra actividad o porque el terminal se ha suspendido o bloqueado.
- onStop(): La aplicación deja de ser visible para el usuario y es posible que sea destruida si el sistema necesita la memoria para otras tareas.
- onRestart(): Se llama cuando se vuelve a la actividad después de haber estado detenida.
- onDestroy(): Se llama antes de que la actividad sea totalmente destruida. Por ejemplo, si el usuario pulsa el botón de vovler, se llama al método finish() o el sistema decide destruirla para liberar memoria.

2.2.1.1. Activity

Una activity o actividad es un componente que proporciona una pantalla con la que el usuario puede interaccionar para hacer algo, como llamar por teléfono, sacar una foto, enviar un mail o ver un mapa. A cada actividad se le proporciona una ventana en la que dibujar su interfaz de usuario. Normalmente esta ventana llena la pantalla por completo, pero puede darse el caso de que se más pequeña que la pantalla y que "flote" sobre otras ventanas.

Normalmente, como ya se ha dicho anteriormente, las aplicaciones están formadas por un conjunto de actividades que están ligeramente vinculadas entre ellas. Normalmente, una de ellas es considerada la aplicación principal, definida en el proyecto como lauhing activity, la cual se mostrará al usuario al iniciar la aplicación. Las actividades pueden llamarse entre ellas y, cada vez que una es creada, la anterior es detenida, y el sistema la mantiene en una pila con un mecanismo LIFO (last in, first out), es decir, que se irán recuperando a medida que el usuario pulse el botón "atrás" en orden inverso al que fueron introducidas en dicha pila.

Los métodos principales para controlar el ciclo de vida de las actividades se han descrito en el apartado anterior.

2.2.1.2. View

Esta clase representa un control de usuario básico, a partir del cual heredarán los widgets que darán lugar a los botones, carteles, cajas de texto... que compondrán el interfaz de usuario de la actividad.

Una vista ocupa un área determinada de la pantalla, y proporciona varios métodos manejadores de eventos para controlar su comportamiento.

La clase ViewGroup, por su parte, es la raíz a partir de la cual se generan los layouts o controles de tipo contenedor dentro de los cuales se enmarcan las

vistas. Estos elementos definen la apariencia de la pantalla en general, permitiendo alinear las vistas dentro de ellas de diversas maneras y ofreciendo una gran cantidad de posibilidades.

2.2.1.3. Intent

Un intent no es más que la intención de llevar a cabo una tarea. Es el método utilizado por Android para indicar, por ejemplo, que se debe lanzar una nueva actividad. Un intent es un objeto de mensaje que se puede utilizar para solicitar una acción a otro componente de la aplicación. Aunque los intents facilitan la comunicación entre distintos componentes de la apliación, hay tres usos fundamentales de ellos:

- Empezar una actividad: Puedes lanzar una nueva actividad pasando un intent al método startActivity(). El intent define qué actividad se lanzará y la información que recibirá.
- Lanzar un servicio: Un servicio, como se ha indicado antes, es un componente que realiza tareas en segundo plano sin interfaz de usuario. Para lanzar un servicio se utiliza el método startService(), al que se le pasa un intent que porta la información necesaria para iniciarlo.
- Entregar un mensaje o una retransmisión: Una retransmisión no es más que un mensaje que cualquier aplicación puede recibir. El sistema los utiliza para notificar eventos del sistema a las aplicaciones. Se pueden lanzar pasando un intent a los métodos sendBroadcast(), sendOrderedBroadcast() o sendStickyBroadcast().

Existen dos tipos de intents: los explícitos y los implícitos. Los explicitos se usan para especificar mediante el nombre el componente y se suelen usar para lanzar componente de tu propia aplicación, puesto que conoces el nombre de la clase que representa dicho componente. Por su parte, los intents implícitos, no especifican el nombre del componente, sino que declaran una acción que debe realizarse, lo cual permite a un componente de otra aplicación manejar esa acción.

2.2.1.4. Bundle

Los bundles son una clase utilizada para pasar información y datos entre distintas actividades, o para almacenarla cuando la aplicación se detiene, permitiendo recuperar el estado anterior de las variables y los views que componen la actividad.

2.2.1.5. Toast

Los toast son un mecanismo que proporciona un feedback simple y sencillo sobre una operación en una pequeña ventana emergente. Este mecanismo solo ocupa el tamaño necesario en pantalla, y la actividad que tiene el foco sigue

siendo visible y se sigue pudiendo interactuar con ella. Los toast desaparecen automáticamente después del tiempo de espera que se le indique, el cual puede ser largo (TOAST.LONG) o corto (TOAST.SHORT).

Se pueden crear de manera estática sin crear propiamente el objeto con la llamada a su método makeText() y se muestran mediante su método show().

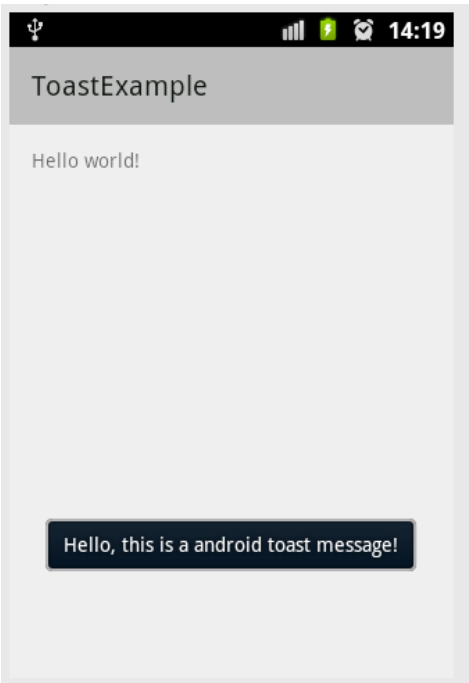

Imagen 5. Ejemplo de mensaje Toast en Android.

Los desarrolladores suelen utilizar los Toast para depurar el código, cuando la ejecución en modo depuración no es posible, para notificar algún mensaje importante al usuario o para mostrar algún tipo de error.

2.2.1.6. SQLiteOpenHelper

SQLite es un motor de bases de datos muy popular hoy en día por ofrecer características tan importantes como su pequeño tamaño, no necesitar servidor, necesitar poca configuración, ser transaccional y, por supuesto, ser de código libre.

Por defecto, Android incorpora todas las herramientas necesarias para la creación y gestión de bases de datos SQLite, entre las que se incluye una API completa para llevar a cabo todas las tareas necesarias de manera sencilla.

La clase auxiliar SQLiteOpenHelper es la encargada de crear, actualizar y conectar con una base de datos SQLite. Concretamente, se deberá crear una clase que heredará de ésta (en el proyecto se ha utilizado la clase SQLiteHandler) y que se deberá personalizar para adaptar a las necesidades propias de la aplicación.

La clase SQLiteOpenHelper tiene un constructor, que normalmente no se sobreescribirá, y dos métodos abstractos, onCreate() y onUpdate(), que se deberán personalizar con el código necesario para crear y actualizar la estructura de la base de datos de la aplicación.

Para utilizar la base de datos se deberá crear el objeto de la clase que hemos creado y llamar a su método getReadableDatabase() o getWritableDatabase(), para obtener una referencia a la base de datos, en función de si se necesita consultar los datos o también se necesita realizar modificaciones sobre los mismos.

2.2.1.7. Application

La clase Application es una clase base útil a la hora de mantener el estado global de la aplicación. Normalmente el programador puede crear su propia instancia de esta clase (en nuestro caso AppControlle.java), especificando un nombre el fichero AndroidManifest.xml, lo que causará que el sistema cree la clase para que sea instanciada cuando la aplicación es creada.

Esta clase también es muy útil a la hora de almacenar valores entre diferentes actividades sin necesidad de pasarlos de unas a otras.

2.2.1.8. Geocoder

Esta clase se encarga de manejar la geocodificación y la geocodificación inversa, es decir, la "traducción" de una dirección de calle en coordenadas latitud y longitud y viceversa. La cantidad de información que contiene una localización de estas características puede variar. Por ejemplo, se puede obtener la dirección concreta del edificio más cercano o puede que solo se obtenga una población y un código postal.

Esta clase precisa de un servicio de backend que no está incluido en el núcleo de Android. Las consultas de geocodificación pueden devolver resultados vacíos si el servicio no está disponible en la plataforma. Se puede usar el método isPresent() para comprobar si el servicio está disponible.

2.2.2. Volley

Volley es una librería HTTP que proporciona a las aplicaciones Android una manera de manejar las redes más sencilla y más rápida. Esta librería está disponible a través del repositorio AOSP.

Las principales ventajas de esta librería son que permite una esquematización automática de las peticiones de red, múltiples conexiones de red concurrentes, manejo de memoria y de disco transparente, priorización de peticiones, un api para cancelar peticiones, facilidad de personalización y herramientas de depuración y trazado, entre otras.

Volley destaca en la realización de operaciones de tipo RPC (Remote Procedure Call), usadas para poblar la vista con los resultados de una búsqueda, por ejemplo. Se integra fácilmente con cualquier protocolo de comunicaciones y trae, por defecto, soporte para strings planos, imágenes y ficheros JSON.

Sin embargo, Volley no está pensada para grandes descargas de datos u operaciones de tipo streaming, ya que la librería mantiene todas las respuestas en memoria mientras se manejan.

2.2.2.1. Request.Method

Esta sencilla clase nos permitirá codificar qué tipo de método (POST, GET...) se va a utilizar en la petición HTTP que se realice al servidor.

2.2.2.2. Response

Esta clase contiene los objetos Listener y ErrorListener que serán los encargados de manejar las peticiones HTTP realizadas al servidor a la hora de registrar usuarios e iniciar sesión.

2.2.2.3. Volley Error

Como su propio nombre indica, esta clase ayudará a manejar los errores que se den durante las conexiones HTTP que se realicen a la hora de registrar usuarios o cuando éstos inicien sesión.

2.2.2.4. toolbox.StringRequest

Esta es la clase utilizada para lanzar la llamada a la petición HTTP que se encargará de iniciar sesión o de registrar a los usuarios en el servidor de bases de datos MySQL. Para construir la petición hará falta indicar que tipo de petición será (POST/GET), la url a la que va dirigida y dos listeners. Estos listeners serán los encargados de manejar los resultados y los errores de dicha petición, sobreescribiendo los métodos onResponse() y onErrorResponse(), estrechamente ligados con la librería JSON explicada en el siguiente apartado.

2.2.3. JSON

JSON es el acrónimo de JavaScript Object Notation y es un formato de fichero para el intercambio de datos, ligeramente similar a XML. Sin embargo, es un subconjunto de la notación literal de objetos JavaScript que no requiere XML, que son ficheros más pesados.

Además, su simplicidad ha propiciado la generalización de este lenguaje de marcado, especialmente como alternativa al ya nombrado XML. La principal ventaja de JSON frente a XML es que es mucho más fácil escribir un analizador o parser para JSON que para XML. En JavaScript, por ejemplo, basta con

utilizar la función eval(), lo cual ha sido fundamental para ser aceptado por la comunidad de desarrolladores AJAX, ya que cualquier navegador es capaz de utilizarlo.

JSON se emplea sobretodo en entornos donde el tamaño del flujo de datos entre cliente y servidor es muy importante, como por ejemplo en aplicaciones Android, y sobre todo cuando la fuente del origen de datos es conocida y fiable.

2.2.3.1. JSONException

Como cualquier excepción, es una clase que extiende la clase Exception y que se lanza para indicar algún tipo de error con la API de JSON. Algunos de estos errores pueden ser los siguientes:

- Intentar leer o construir documentos mal formados.
- Utilizar null como nombre.
- Uso de tipos numéricos no disponibles en JSON como NaN o infinite.
- Búsquedas utilizando índices fuera del rango o nombres inexistentes.
- Contradicciones de tipos durante las búsquedas.

2.2.3.2. JSONObject

La clase JSONObject es la responsable de crear los documentos JSON necesarios para el funcionamiento de la aplicación. En este caso, será la encargada de recoger las respuestas del servidor cuando se intente realizar un registro o inicio de sesión de los usuarios en el mismo.

2.3. Paquetes del proyecto

Un proyecto en Eclipse se estructura de la forma que se puede apreciar en la imagen 6. Cabe destacar entre todas las carpetas la carpeta /src/ que contendrá todo el código fuente de la aplicación, la carpeta /res/, que contendrá todos los ficheros de recursos necesarios para el proyecto (imágenes, vídeos, cadenas de texto, ficheros xml...) y la carpeta /gen/, que contiene una serie de elementos generados automáticamente al desarrollar y compilar el proyecto.

Otras carpetas también importantes son las carpetas:

- /assets/, que contiene el resto de ficheros auxiliares necesarios para el funcionamiento de la aplicación.
- /bin/, contiene los elementos compilados de la aplicación y otros ficheros auxiliares, en principio, el desarrollador no necesita tocar nada dentro de esta carpeta.
- /libs/, contiene las librerías auxiliares necesarias para el correcto funcionamiento de la aplicación. Normalmente las librerías tienen formato .jar.

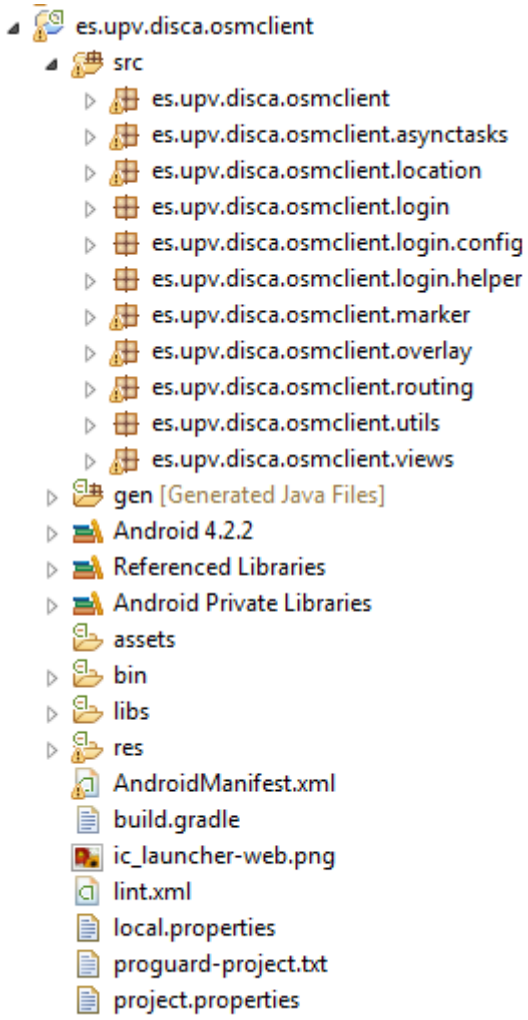

Imagen 6. Estructura de un proyecto en Eclipse

En los capítulos que siguen se profundiza en las carpetas y ficheros más importantes que conforman el proyecto de aplicación Android.

2.3.1. AndroidManifest.xml

Todas las aplicaciones Android creadas en Eclipse tienen este fichero. Como su propio nombre indica, es un manifiesto en el que se indican todos los componentes que formarán parte de la aplicación, ya sean actividades, pantallas, mensajes..., las librerías auxiliares necesarias, los permisos necesarios para el funcionamiento de la aplicación y los datos de identificación de la misma (nombre, versión, icono...).

Concretamente, la aplicación tendrá los permisos de localización, acceso a red wifi y a internet, permiso de escritura en el almacenamiento externo o tarjeta sd y permiso para poder depurar la aplicación.

Además, se declaran las actividades necesarias para el funcionamiento de la aplicación: LoginActivity, RegisterActivity y MainActivity, que se explicarán a continuación.

2.3.2. Código fuente (src)

Como ya se ha dicho anteriormente, en esta carpeta se incluyen los archivos de código fuente de la aplicación. Concretamente se han estructurado en tres paquetes dentro de esta carpeta que se definen a continuación.

2.3.2.1. es.upv.disca.osmlcient.login

En este paquete encontramos las tres actividades principales de la aplicación LoginActivity, RegisterActivity y MainActivity.

LoginActivity

Esta es la clase encargada de realizar las peticiones al servidor para comprobar que el usuario que inicia sesión existe y es válido. Esta funcionalidad es accedida desde el evento de pulsación del botón de inicio de sesión, el cual llama al método checkLogin() de la propia clase. Este último método será el encargado de verificar los detalles del usuario en el servidor realizando la petición http gracias a la librería volley. Cabe destacar que antes de llamar a este método se hacen las comprobaciones oportunas para comprobar que el email tenga un formato válido.

A continuación, veremos este método en profundidad.

```
private void checkLogin(final String email, final String password) {
    // Tag used to cancel the request
    String tag_string_req = "req_login";

    pDialog.setMessage("Iniciando sesión ...");
    showDialog();

    remember = checkRemember.isChecked();

    StringRequest strReq = new StringRequest(Method.POST, AppConfig.URL_REGISTER,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            Log.d(TAG, "Login Response: " + response.toString());
            hideDialog();

            try {
                //Workaround para evitar problemas con el php
                //devuelve los errores antes del json y no se puede leer
                response = response.substring(response.indexOf("{") + 1, response.length());
                response = response.substring(response.indexOf("{") , response.length());

                JSONObject jObj = new JSONObject(response);
                boolean error = jObj.getBoolean("error");

                // Check for error node in json
                if (!error) {
                    // user successfully logged in
                    // Create login session
                    if(remember) session.setLogin(true);

                    // Launch main activity
                    Intent intent = new Intent(LoginActivity.this, MainActivity.class);
                    startActivity(intent);
                    overridePendingTransition(R.animator.login_to_register_in,
R.animator.login_to_register_out);
                    finish();
                } else {
```

```
                // Error in login. Get the error message
                String errorMsg = jObj.getString("error_msg");
                Toast.makeText(getApplicationContext(), errorMsg, Toast.LENGTH_LONG).show();
            }
        } catch (JSONException e) {
            // JSON error
            e.printStackTrace();
        }
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Log.e(TAG, "Login Error: " + error.getMessage());
        Toast.makeText(getApplicationContext(), error.getMessage(), Toast.LENGTH_LONG).show();
        hideDialog();
    }
}) {
    @Override
    protected Map<String, String> getParams() {
        // Posting parameters to login url
        Map<String, String> params = new HashMap<String, String>();
        params.put("tag", "login");
        params.put("email", email);
        params.put("password", password);

        return params;
    }
};
// Adding request to request queue
AppController.getInstance().addToRequestQueue(strReq, tag_string_req);
}
```

Vayamos por partes. En primer lugar se crea un dialog, un tipo de vista que solo ocupa una parte de la pantalla que suele utilizarse para indicar que el sistema está llevando a cabo una actividad que puede durar más tiempo de lo normal. Inmediatamente después encontramos la petición http, dentro de la cual encontramos todos los métodos que serán utilizados para controlarla.

Como ya se vio en un punto anterior de la memoria, esta petición necesitará cuatro parámetros:

- El tipo de petición que será, en nuestro caso `Method.POST`
- la url a la que va dirigida, en nuestro caso definida en la clase appconfig, `AppConfig.URL_REGISTER`
- Un listener en caso de que la petición sea correcta
    - En caso de que la petición funcione, lo primero que encontramos es un workaround para evitar un error ocasionado por las sentencias MySQL utilizadas que no tiene mayor importancia y no impide el correcto funcionamiento de la aplicación.
    - A continuación, si no ha habido otros errores, si el usuario ha elegido recordar su sesión mediante el checkBox de la pantalla inicial, se lo indicaremos a la aplicación mediante la llamada a `session.setLogin(true)`.
    - Finalmente, lanzaremos la pantalla de bienvenida, sobreescribiendo la transición por defecto de Android por una más sofisticada que añade más detalle a la aplicación.

        - No obstante, si ha habido un error posterior a la petición, este se mostrará al usuario mediante las funcionalidades proporcionadas por la clase Toast de la que hemos hablado anteriormente.
- Un listener en caso de que la petición devuelva un error
    - Si la petición solicitada al servidor ha devuelto algún tipo de error, ya sea de tiempo de espera o de otro tipo, este se mostrará al usuario, permitiéndole volver a introducir sus credenciales para un nuevo intento.

A continuación, encontramos el fragmento que establece los parámetros para la petición, siendo el más importante `params.put("tag", "login");`, que servirá al servidor para encauzar la petición, como veremos más alante en el apartado del servidor, como una solicitud de inicio de sesión.

Finalmente, encontramos la última línea de código que simplemente agrega la petición a la clase AppController para almacenar el estado de la misma.

Además, gracias al mecanismo incorporado para recordar los datos de inicio de sesión de los usuarios, se incluye el siguiente fragmento de código para pasar directamente a la pantalla de bienvenida si el usuario había decidido recordar su inicio de sesión. Para ello utilizamos la referencia creada a la clase SessionManager.

```
// Check whether the user is already logged in or not
if (session.isLoggedIn()) {
   // User is already logged in. Take him to main activity
   Intent intent = new Intent(LoginActivity.this, MainActivity.class);
   startActivity(intent);
   finish();
}
```

Por otra parte, también destaca una pequeña clase creada dentro de la actividad llamada MyGestureListenerToLeft, que utiliza una animación que describiremos a continuación, e inicializada mediante la siguiente línea de código, que servirá para cambiar a la pantalla de registro de usuarios mediante el gesto de desplazamiento en cualquier parte de la pantalla.

```
gestureDetectorCompat = new GestureDetectorCompat(
          this, new MyGestureListenerToLeft());
```

No obstante, también cabe destacar que al inicializar la actividad se crearán referencias a todos los controles de la pantalla y se vinculará la actividad a su layout xml para establecer los vínculos y referencias necesarias para el correcto funcionamiento de la misma.

```
setContentView(R.layout.activity_login);
```

RegisterActivity

Esta clase será la encargada de registrar nuevos usuarios en el servidor, así como de almacenarlos en la base de datos local SQLite. No obstante será muy similar a la actividad de inicio de sesión, cambiando la petición y añadiendo métodos para la comprobación de que las credenciales proporcionadas por el usuario sean válidas.

Lo primero será inicializar las variables y referencias necesarias para el correcto funcionamiento de la actividad y el establecimiento del vínculo con el layout de la pantalla de registro.

```
setContentView(R.layout.activity_register);
```

En esta clase también se utiliza la referencia a RightDrawableOnTouchListener para modificar las cajas de texto donde se escriben las contraseñas para cambiar entre ver el texto u ocultarlo mostrando asteriscos o puntos.

Como antes, la funcionalidad del registro de usuarios será accedida desde el evento que controla el clic del botón de registro, el cual llamará al método checkParameters() para comprobar las credenciales del usuario y luego, si estas son válidas, llamará al método registerUser().

El método de comprobación de parámetros comprobará primero que los campos no estén vacíos. También comprobará que la dirección de email sea una dirección válida y que los valores proporcionados en los campos de contraseña y de repetir contraseña coincidan. En caso de que alguna comprobación no sea satisfactoria se mostrará un mensaje al usuario con la ayuda del Toast y se devolverá false. En caso de que todas las comprobaciones sean correctas se devolverá true.

```
private boolean checkParameters(String name, String email,
      String password, String repeatPassword){
   if (!name.isEmpty() && !email.isEmpty() && !password.isEmpty() && !repeatPassword.isEmpty())
{
      if(isValidEmail(email)){
         if(password.equals(repeatPassword)){
            return true;
         } else {
            mostrarError("¡Las contraseñas no coinciden!");
         }
      }
      else{
         mostrarError("¡La dirección de email no es válida!");
      }
   }
   else{
      mostrarError("¡Sí claro! Te faltan datos... :P");
   }
   return false;
}
```

Por su parte, el método registerUser(), realizará la petición necesaria para el registro de usuarios en el servidor y almacenará el usuario en la base de datos SQLite.

```
private void registerUser(final String name, final String email, final String password) {
   // Tag used to cancel the request
```

```
    String tag_string_req = "req_register";

    pDialog.setMessage("Registrando ...");
    showDialog();

    StringRequest strReq = new StringRequest(Method.POST, AppConfig.URL_REGISTER, new
Response.Listener<String>() {

        @Override
        public void onResponse(String response) {
            Log.d(TAG, "Register Response: " + response.toString());
            hideDialog();

            try {
                //Workaround para evitar problemas con el php
                //devuelve los errores antes del json y no se puede leer
                response = response.substring(response.indexOf("{") + 1, response.length());
                response = response.substring(response.indexOf("{") , response.length());

                JSONObject jObj = new JSONObject(response);
                boolean error = jObj.getBoolean("error");
                if (!error) {
                    // User successfully stored in MySQL
                    // Now store the user in sqlite
                    String uid = jObj.getString("uid");

                    JSONObject user = jObj.getJSONObject("user");
                    String name = user.getString("name");
                    String email = user.getString("email");
                    String created_at = user.getString("created_at");

                    // Inserting row in users table
                    db.addUser(name, email, uid, created_at);

                    // Launch login activity
                    Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);
                    startActivity(intent);
                    finish();
                } else {
                    // Error occurred in registration. Get the error
                    // message
                    String errorMsg = jObj.getString("error_msg");
                    Toast.makeText(getApplicationContext(), errorMsg, Toast.LENGTH_LONG).show();
                }
            } catch (JSONException e) {
                e.printStackTrace();
                Toast.makeText(getApplicationContext(), e.toString(), Toast.LENGTH_LONG).show();
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.e(TAG, "Registration Error: " + error.getMessage());
                Toast.makeText(getApplicationContext(), error.getMessage(),
Toast.LENGTH_LONG).show();
                hideDialog();
            }
        }) {
            @Override
            protected Map<String, String> getParams() {
                // Posting params to register url
                Map<String, String> params = new HashMap<String, String>();
                params.put("tag", "register");
                params.put("name", name);
                params.put("email", email);
                params.put("password", password);

                return params;
            }
        };

        // Adding request to request queue
        AppController.getInstance().addToRequestQueue(strReq, tag_string_req);
    }
```

La principal diferencia con la actividad de inicio de sesión es que en este caso el parámetro que se utilizará para encauzar la petición indica que es una petición de registro `params.put("tag", "register");`. El resto de parámetros se utilizarán para crear el usuario en la base de datos del servidor. La otra gran diferencia respecto al código de inicio de sesión es que, al crear el usuario, este también se creará en la base de datos del terminal llamando al método addUser() de la clase que se encarga de gestionar la base de datos, SQLiteHandler, `db.addUser(name, email, uid, created_at);`. Además, si el registro del usuario se lleva a cabo con éxito, se devolverá la aplicación a la pantalla de inicio de sesión para que proceda a iniciar sesión y empezar con la actividad.

Por otro lado, en este caso la pantalla también dispone de un gesto para devolver al usuario a la pantalla de registro. Sin embargo, a diferencia del anterior, en este caso la clase se llama `MyGestureListenerToRight`. Como su propio nombre indica, en este caso la animación, que veremos más adelante, mueve la pantalla hacia la derecha, deslizando el dedo de izquierda a derecha.

MainActivity

En este caso se trata de una pantalla de bienvenida que mostrará el usuario que ha iniciado sesión y un botón central principal que permitirá el acceso al navegador para que el usuario pueda comenzar a buscar rutas y enviar su información de tráfico al servidor. Para ello se llamará al método iniciarNavegacion() que creará un intent hacia la clase OSMClientActivity pasándole el usuario que ha iniciado sesión a través de un string dentro de un bundle, clase auxiliar para compartir información entre actividades Android que se ha explicado anteriormente.

```
private void iniciarNavegacion(){
   // Launching the login activity
   Intent intent = new Intent(MainActivity.this, OSMClientActivity.class);
   Bundle extras = new Bundle();
   extras.putString("User", txtName.getText().toString());
   intent.putExtras(extras);
   startActivity(intent);
   overridePendingTransition(R.animator.login_to_register_in,
         R.animator.login_to_register_out);
   finish();
}
```

No obstante, otra función imprescindible de esta pantalla es la de desconectar al usuario que ha iniciado sesión, de manera que la aplicación olvidará sus datos de inicio de sesión y no evitará esa pantalla la próxima vez que inicie la aplicación. Para ello, se llamará al método logoutUser() desde el evento que gestiona la presión del botón “desconectar”. Este método llamará al método setLogin() de la clase SessionManager pasando el parámetro booleano false para que se olviden las credenciales del usuario y creará un intent que devolverá al usuario a la pantalla de inicio de sesión. Además, se borrarán los datos del usuario de la base de datos SQLite para evitar problemas futuros llamando al

método deleteUsers de la clase SQLiteHandler, que se verá un poco más adelante.

```
private void logoutUser() {
   session.setLogin(false);

   db.deleteUsers();

   // Launching the login activity
   Intent intent = new Intent(MainActivity.this, LoginActivity.class);
   startActivity(intent);
   overridePendingTransition(R.animator.register_to_login_in,
         R.animator.register_to_login_out);
   finish();
}
```

2.3.2.2. es.upv.disca.osmlcient.login.config

En este paquete se almacenan los ficheros Java necesarios para la correcta configuración de los parámetros de la aplicación.

AppConfig

Es una clase pequeña con dos variables estáticas, ya que se pretende que sean accesibles desde cualquier otra clase de la aplicación. En estas dos variables se guardan las urls del servidor que se van a utilizar para dar de alta nuevos usuarios y para que éstos inicien sesión en el mismo.

```
public static String URL_LOGIN = "http://192.168.43.48/android_login_api/"
public static String URL_REGISTER = "http://192.168.43.48/android_login_api/"
```

AppController

Esta es una clase que hereda de la clase Application que, como ya se ha dicho en un punto anterior de esta memoria, sirve para almacenar el estado de la aplicación y recuperarlo cuando vuelva a ejecutarse o cuando vuelva a tener el foco principal de ejecución cuando el usuario vuelva a ella. Se caracteriza por tener dos variables privadas: mRequestQueue de tipo RequestQueue y mInstance de tipo AppController, que servirá para el propósito definido anteriormente, almacenando la instancia de la aplicación.

2.3.2.3. es.upv.disca.osmlcient.login.helper

En este paquete se han incluido los ficheros necesarios para la gestión de la base de datos de Android SQLite, una clase Java para recordar las credenciales del usuario que inicia sesión a lo largo del resto de pantallas de la aplicación y una última clase que proporciona la utilidad necesaria para modificar las cajas de texto donde se escriben las contraseñas al iniciar sesión o registrarse, que permite modificarlas para que muestren la cadena de texto que está escrito o muestren los asteriscos típicos de los campo de contraseña, por si se necesitase validar lo que se ha escrito.

SQLiteHandler

Esta es la clase encargada de crear la base de datos SQLite de la aplicación. También se encargará de almacenar la información de los usuarios creados mediante su método addUser(). Además, dispone de un método llamado getUserDetails() que nos devolverá la información del usuario para no tener que obtenerla siempre desde el servidor.

Destacan en ella las variables DATABASE_NAME y TABLE_LOGIN, que representan el nombre de la base de datos y de la tabla de usuarios respectivamente. Además, se declaran las variables KEY_ID, KEY_NAME, KEY_EMAIL, KEY_UID y KEY_CREATED_AT para almacenar los nombres de las columnas de la tabla de usuarios para facilitar la inserción y recuperación de estos campos.

La clase constará de un constructor y el método onCreate heredado de SQLiteOpenHelper que permitirán la creación de la base de datos.

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

// Creating Tables
@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);
	Log.d(TAG, "Database tables created");
}
```

Así pues, una vez creada la base de datos mediante los métodos anteriores, destacan los métodos que proporcionan la verdadera funcionalidad a la clase. Entre ellos, el método addUser se encargará de insertar los usuarios creados en la base de datos, siempre y cuando sean usuarios válidos para el servidor y creándose previamente en el mismo.

```
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); // Name
   values.put(KEY_EMAIL, email); // Email
   values.put(KEY_UID, uid); // Email
   values.put(KEY_CREATED_AT, created_at); // Created At
   // Inserting Row
   long id = db.insert(TABLE_LOGIN, null, values);
   db.close(); // Closing database connection
   Log.d(TAG, "New user inserted into sqlite: " + id);
}
```

A su vez, el método getUserDetails será el encargado de obtener la información del usuario (sobre todo en la pantalla de bienvenida) para mostrarla en pantalla. Para ello devolverá una variable HashMapz<String, String> en la que almacenará el campo y su valor de cada columna que se necesite de la base de datos, concretamente el nombre, el email, el uid y la fecha de creación.

```
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);
   // Move to first row
   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
   Log.d(TAG, "Fetching user from Sqlite: " + user.toString());

   return user;
}
```

SessionManager

Como ya se ha dicho anteriormente, esta clase será la encargada de mantener la información del usuario que inicia sesión disponible para el sistema en cualquier parte de la aplicación. Para ello utilizará las SharedPreferences, que no son más que una clase propia del SDK de Android que proporciona un framework general que permite almacenar y recuperar pares clave-valor persistentes de tipos de datos primitivos (booleans, floats, ints, longs y strings). Esta información permanecerá disponible entre sesiones, aunque la aplicación sea matada por el sistema para liberar memoria.

RightDrawableOnTouchListener

Como ya se ha dicho anteriormente, esta clase sirve para ser aplicada a los textboxes de contraseñas para ser aplicada su método onTouchListener y modificar el texto para ser mostrado u ocultado dependiendo de las necesidades del usuario. Para ello se utiliza el siguiente fragmento de código.

```
inputPassword.setOnTouchListener(new
RightDrawableOnTouchListener(inputPassword){
       @Override
   public boolean onDrawableTouch(final MotionEvent event) {
      int type = inputPassword.getInputType();
      if(type == 129)
         inputPassword.setInputType(InputType.TYPE_TEXT_VARIATION_NORMAL);
      else
         inputPassword.setInputType(InputType.TYPE_CLASS_TEXT |
            InputType.TYPE_TEXT_VARIATION_PASSWORD);
      return true;
        }
});
```

2.3.2.4. es.upv.disca.osmclient.location

Este paquete contiene clases creadas en el proyecto realizado por David, y que he tenido que modificar para añadir las funcionalidades necesarias para cumplir los objetivos previstos para este nuevo proyecto.

GpsMyLocationProvider

Esta es la clase principal que se ha visto modificada para proporcionar las nuevas funcionalidades a la aplicación. Concretamente se ha tenido que modificar el método onLocationChanged del LocationListener.

El LocationListener es la clase encargada de controlar el gps en Android. Para ello se implementa una variable de este tipo y se suelen sobreescribir sus cuatro métodos principales: onLocationChanged, onProviderDisabled, onProviderEnabled y onStatusChanged.

Así pues, se ha añadido el código necesario para recoger los datos de posición del usuario y enviarlos al servidor. Para ello se ha utilizado la clase Geocoder de Android que utiliza los servicios de GoogleMaps para realizar el proceso de geocodificación reversa, que consiste en obtener una dirección concreta a partir de unas coordenadas de latitud y longitud dadas.

```
geo = new Geocoder(mContext, Locale.getDefault());
List<Address> list =
   geo.getFromLocation(Double.valueOf(location.getLatitude()),
   Double.valueOf(location.getLongitude()), 1);
if (list != null && list.size() > 0) {
    Address address = list.get(0);
    direccion = address.getAddressLine(0);
    calle = direccion.split(",")[0];
    if(direccion.split(",").length == 2)
    {
        numero = direccion.split(",")[1];
    }
    poblacion = address.getLocality();
    if (poblacion==null)
        poblacion = "";
    velocidad = Float.toString(location.getSpeed());

    enviarLocalizacion(poblacion, calle, numero, velocidad);
}
```

Como vemos, se utiliza la clase geocoder pasándole la latitud y la longitud proporcionadas por el método onLocationChanged, que es llamado cada vez que el gps del dispositivo recibe nueva información respecto a sus coordenadas. El método getFromLocation() de esta clase nos proporciona unos resultados que se interpretan como una lista, a partir de los cuales se puede obtener un objeto de tipo Address del cual extraer la información referente a la dirección de la ubicación. Una vez obtenida toda la información, se le pasa la método enviarLocalizacion() que se encargará de enviar dicha información al servidor, a través de una nueva petición http de la librería volley.

```
private void enviarLocalizacion(final String poblacion, final String calle,
        final String numero, final String velocidad) {
   // Tag used to cancel the request
   String tag_string_req = "req_location";

   StringRequest strReq = new StringRequest(Method.POST,
           AppConfig.URL_REGISTER, new Response.Listener<String>() {
      @Override
      public void onResponse(String response) {
         try {
            //Workaround para evitar problemas con el php
            //devuelve los errores antes del json y no se puede leer
            response = response.substring(response.indexOf("{") + 1,
response.length());
            response = response.substring(response.indexOf("{") ,
response.length());

            JSONObject jObj = new JSONObject(response);
            boolean error = jObj.getBoolean("error");
            if (!error) {

            } else {
               // Error occurred while sending location. Get the error message
               String errorMsg = jObj.getString("error_msg");
               Toast.makeText(mContext, errorMsg, Toast.LENGTH_LONG).show();
            }
         } catch (JSONException e) {
            e.printStackTrace();
            Toast.makeText(mContext, e.toString(), Toast.LENGTH_LONG).show();
         }
      }
   }, new Response.ErrorListener() {
      @Override
      public void onErrorResponse(VolleyError error) {
         Toast.makeText(mContext, error.getMessage(),
Toast.LENGTH_LONG).show();
      }
   }) {
      @Override
      protected Map<String, String> getParams() {
         // Posting params to register url
         Map<String, String> params = new HashMap<String, String>();
         params.put("tag", "location");
         params.put("name", User);
         params.put("poblacion", poblacion);
         params.put("calle", calle);
         params.put("numero", numero);
         params.put("fechaHora", now());
         params.put("velocidad", velocidad);

         return params;
      }
   };

   // Adding request to request queue
   AppController.getInstance().addToRequestQueue(strReq, tag_string_req);
}
```

Como se puede observar, el método sigue la misma estructura de petición http que hacían los encargados de registrar usuarios e iniciar sesión. La principal diferencia es que ahora enviaremos el tag de "location" para que el servidor encauce la petición de manera adecuada y también se enviarán los datos necesarios para crear un nuevo registro en la tabla locations del servidor (nombre del usuario, población, calle, número, fecha y hora y velocidad).

2.3.3. Carpeta de recursos (res)

Esta otra carpeta, propia de todo proyecto Android, es utilizada para contener los archivos que tienen que ver con la parte visual de la aplicación, ya sean los layouts, es decir, los ficheros que definen las diferentes pantallas de la aplicación, las imágenes, las animaciones que definen las transiciones entre pantallas, los menus de cada pantalla o diferentes valores y entradas de texto que tomarán las variables de las diferentes actividades y vistas de la aplicación.

2.3.3.1. Definición de vistas (layout)

Esta carpeta contendrá los ficheros xml que definirán las pantallas de la aplicación. Para ello, se anidarán dentro de un contenedor o layout las distintas vistas de todo tipo que componen cada pantalla (EditText, CheckBox, Button...)

activity_login.xml

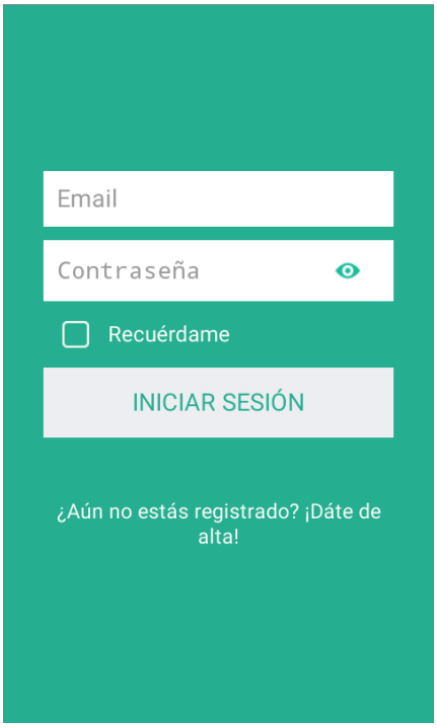

Imagen 7. Vista de la pantalla de inicio de sesión

Como podemos observar, esta pantalla cuenta con dos cajas de texto o EditText para introducir el email y la contraseña con los que el usuario iniciará sesión.

Tienen un “hint” o pista inicializado que recuerda al usuario para que sirve cada uno de los campos. Además, el campo de la contraseña se ha inicializado con una imagen situada a la derecha del mismo (mediante la propiedad DrawableRight) que será el que se una al evento definido anteriormente con la clase RightDrawableOnTouchListerener para mostrar u ocultar la contraseña.

A continuación encontramos el CheckBox que permitirá al usuario recordar sus credenciales para no tener que introducirlas cada vea que inicie sesión. De este CheckBox cabe destacar que se ha personalizado utilizando un recurso definido como cb_selector, que veremos a continuación, que se le pasa al CheckBox a través de su propiedad Button. Esto hace que en vez de utilizar los iconos predefinidos por Android para los estados de marcado y desmarcados utilice unos personalizados que están incluidos como recursos de tipo imagen en la aplicación llamados checked.png y unchecked.png.

Finalmente nos encontramos con dos botones, el botón de inicio de sesión, mediante la pulsación del cual se intentará iniciar sesión en el servidor y se pasará a la pantalla de bienvenida y el botón de ir a la pantalla de registro (del mismo color del fono, por lo que no se aprecia como botón en la imagen), útil cuando el usuario todavía no se ha registrado. No obstante, la funcionalidad de este último también puede ser accedida desplazando el dedo de derecha a izquierda de la pantalla, puesto que así se ha definido en el código.

activity_register.xml

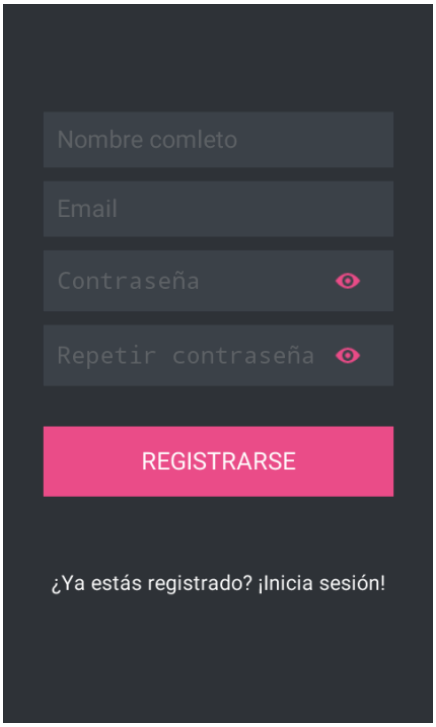

Imagen 8. Vista de la pantalla de registro de nuevos usuarios

En esta ocasión la pantalla presenta cuatro cajas de texto, para escribir los datos que en ellas se indica. Los dos campos de contraseña se han definido de igual manera que en la pantalla de inicio de sesión, con una imagen a la derecha del mismo que se puede clicar para cambiar entre ver el texto u ocultarlo, utilizando de nuevo la clase RightDrawableOnTouchListiner.

De nuevo, cuenta con dos botones. El primero de ellos de color rosa permite realizar la petición al servidor una vez se han rellenado todos los datos para crear el usuario en la base de datos. El segundo, del mismo color que el fondo, permite volver a la pantalla de inicio de sesión. En este caso también se ha incorporado código para gestionar el gesto de desplazar el dedo de izquierda a derecha para volver a dicha pantalla.

activity_main.xml

Imagen 9. Vista de la pantalla de bienvenida

A diferencia de las dos pantallas anteriores, esta cuenta con tres TextViews y no EditTexts, puesto que no se va a solicitar ninguna información adicional al usuario y solo se pretende mostrarle información. El primero de ellos es visible a con el texto “Bienvenido”. Los otros, sin embargo, no es que estén oculton, es que se rellenarán en tiempo de ejecución con el nombre y el email del usuario que haya iniciado sesión. Para ello se utiliza el siguiente código, habiendo obtenido previamente referencias a los TextViews nombradods.

```
String name = user.get("name");
String email = user.get("email");

// Displaying the user details on the screen
```

```
txtName.setText(name);
txtEmail.setText(email);
```

Además, esta pantalla vuelve a contar con dos botones. El primero de ellos permitirá acceder a la pantalla del mapa para iniciar la navegación, la búsqueda de rutas y el envío de datos al servidor. El segundo, por su parte, permite cerrar sesión al usuario conectado de forma que, además, se olvidarán sus credenciales si se estaban recordando.

2.3.3.2. Carpeta de imágenes e iconos (drawable)

En esta carpeta se organizan las imágenes que se añaden al proyecto para ser utilizadas en los ImageButton, Images u otros elementos utilizados en la aplicación. Además, una vez se añade una imagen a la aplicación, se realizan varias copias de diversos tamaños que serán utilizados en función de la resolución y tamaño de pantalla del dispositivo en el que se instale la aplicación. Estas copias se guardan en distintas subcarpetas dentro de la carpeta drawable dependiendo de su tamaño.

En nuestro caso destacan las imágenes checked.png y unchecked.png utilizadas para sobreescribir las que incluye pro defecto el CheckBox del SDK de Android. Por otra parte, también destacan las imágenes eye_pink.png y ojo_green.png, que serán las utilizadas en las cajas de texto de tipo contraseña para alternar el modo en que se visualiza el texto introducida en ellas. Finalmente, destaca una última imagen llamada ic_launcher.png que he introducido sustituyendo a la que proporciona Android por defecto y que actuará como icono de la aplicación en el menú de aplicaciones del sistema operativo.

cb_selector.xml

Es en esta carpeta donde se incluye este pequeño fichero del que hemos hablado anteriormente y que será utilizado para modificar el CheckBox de Android.

2.3.3.3. Carpeta de animaciones (animator)

Esta carpeta no está incluida por defecto cuando se crea un nuevo proyecto de Android, si no que ha sido añadida después de la creación del proyecto. No obstante, los ficheros aquí almacenados podrían haberse guardado en otra carpeta.

Las animaciones son ficheros xml que pueden resultar tan complejos como la animación que definen. Los utilizados por nuestra aplicación sin embargo, no son muy complicados, a pesar de que los efectos que se consiguen con ellos funcionan de manera fluida y actúan de manera consistente.

Cuando estos ficheros sustituyen a las animaciones por defecto que incluye Android para pasar de una pantalla a otra, funcionan en parejas, puesto que se define uno para la pantalla saliente y otro para la entrante. Es por esta razón

que nuestra aplicación incluye cuatro de estos ficheros, a pesar de tener solo dos animaciones reales. Estos se organizan en dos parejas: login_to_register y register_to_login. Cada pareja tiene un fichero para la pantalla entrante (in) y otro para la pantalla saliente (out).

Su utilización resulta sencilla a pesar de lo que pueda parecer y simplemente hay que reemplazar la animación por defecto de Android.

```
overridePendingTransition(R.animator.login_to_register_in,
   R.animator.login_to_register_out);
```

Veamos el ejemplo de **login_to_register_in**

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <translate android:fromXDelta="100%p" android:toXDelta="0" android:duration="300"/>
   <alpha android:fromAlpha="0.0" android:toAlpha="1.0" android:duration="300" />
</set>
```

El elemento translate controla la posición y localización del layout o cualquier vista sobre la que se aplica la animación. Puede mover un objeto en cualquier dirección X o Y. En este caso mueve el objeto desde el margen derecho (100% de los píxeles) hasta la posición inicial (0). Además, también se puede indicar cuanto debe durar la animación, en milisegundos. En nuestro caso la animación durará 0,3 segundos.

Por su parte, el elemento alpha controla el nivel alpha de cualquier vista, es decir, su opacidad, en una escala de 0 a 1. En esta caso, la pantalla entrante pasará de estar totalmente oculta (nivel alpha 0) a estar visible al 100% (nivel alpha 1). También en este caso se indica que la animación debe tardar 300 milisegundos en terminar este proceso.

2.3.3.4. Definición de valores (values)

En esta carpeta se nos permite definir una serie de valores que serán accesibles desde cualquier punto de la aplicación. Normalmente es útil para definir todas las cadenas de texto que se van a utilizar en una aplicación, lo cual servirá para modificarlas rápidamente sin tener que buscarlas en cada punto de la aplicación donde aparezca la cadena. Además, también permitirá traducir la aplicación de manera rápida y sencilla, simplemente sustituyendo el fichero donde se definen las cadenas de texto de la aplicación (strings.xml), por el nuevo fichero en el nuevo idioma.

No obstante, a pesar de permitir la definición de las cadenas de texto incluidas en la aplicación, también se pueden definir muchos otros valores.

strings.xml

Este fichero, como hemos dicho, permite definir las cadenas de texto para ser utilizada en la aplicación, a las cuales se accede de dos formas. Desde los

ficheros xml de definición de vistas se accederá mediante "@string/nombre" y desde los ficheros de código fuente se accederá utilizando la referencia R.string.nombre.

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:android="http://schemas.android.com/apk/res/android">

<string name="hint_email">Email</string>
    <string name="hint_password">Contraseña</string>
    <string name="hint_repeat_password">Repetir contraseña</string>
    <string name="hint_name">Nombre comleto</string>
    <string name="btn_login">INICIAR SESIÓN</string>
    <string name="btn_register">REGISTRARSE</string>
    <string name="btn_link_to_register">¿Aún no estás registrado? ¡Dáte de
alta!</string>
    <string name="btn_link_to_login">¿Ya estás registrado? ¡Inicia
sesión!</string>
    <string name="welcome">Bienvenido</string>
    <string name="btn_logout">DESCONECTAR</string>
    <string name="name">Nombre completo</string>
    <string name="remember_me">Recuérdame</string>
    <string name="empezar">¡Iniciar la navegación!</string>
    <string name="title_activity_map">MapActivity</string>
</resources>
```

colors.xml

Este fichero nos permitirá definir una serie de colores y asignarles un nombre, ya sea el nombre del color o una referencia al control para el que lo hemos creado.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="bg_login">#26ae90</color>
    <color name="bg_register">#2e3237</color>
    <color name="bg_main">#428bca</color>
    <color name="white">#ffffff</color>
    <color name="input_login">#222222</color>
    <color name="input_login_hint">#999999</color>
    <color name="input_register">#888888</color>
    <color name="input_register_bg">#3b4148</color>
    <color name="input_register_hint">#5e6266</color>
    <color name="btn_login">#26ae90</color>
    <color name="btn_login_bg">#eceef1</color>
    <color name="lbl_name">#333333</color>
    <color name="btn_logut_bg">#ff6861</color>
</resources>
```

Estos son los colores utilizados por la aplicación. De la misma manera que las cadenas de texto, para utilizarlos utilizaremos "@colors/nombre" o R.color.nombre.

Así pues, se podrían definir gran cantidad de ficheros para definir arrays, estilos o cualquier otra funcionalidad que pueda ser almacenada en un fichero xml.

3. Servidor

En este apartado se presenta el proceso llevado a cabo para el desarrollo completo de la parte del servidor encargado de almacenar los datos de tráfico así como los usuarios de este proyecto. Para ello veremos el sistema utilizado (WAMP) y las diferentes herramientas de programación utilizadas para el desarrollo, la creación de la base de datos MySQL, y la creación del servidor con PHP.

3.1. WAMP

WAMP no es más que el acrónimo utilizado para definir un sistema de infraestructura de internet utilizado para crear rápidamente un servidor con las siguientes herramientas:

- **W**indows, como sistema operativo.
- **A**pache, como servidor web.
- **M**ySQL, como herramienta gestora de las bases de datos
- **P**HP, como lenguaje de programación, aunque a veces también se utiliza **P**ython o **P**erl.

Por todo ello, el uso de WAMP permite servir páginas html a internet y gestionar datos en línea. Además, con el uso de los lenguajes de programación, permite el desarrollo de aplicaciones Web y Web Services.

El proyecto WampServer es un proyecto de código libre que permite a los desarrolladores crear servidores web rápidamente. Además, la herramienta PhpMyAdmin, puesta a disposición por el servidor, permite un rápido y sencillo manejo de las bases de datos creadas para el sistema.

Este conjunto de herramientas también está disponible para Linux (LAMP) y Macintosh (MAMP).

3.1.1. Funcionamiento

Para poner un servidor web en marcha mediante WAMP, no hace falta más que descargar la herramienta desde la página web del proyecto (www.wampserver.com/en) e instalarla siguiendo, generalmente, las opciones por defecto durante el proceso de instalación.

Una vez instalado, basta con ejecutar la aplicación y se ponen en marcha todos los servicios.

Cuando se instala, se crea la carpeta wamp en el directorio indicado durante el proceso de instalación, por defecto, C:/wamp. Dentro de esta carpeta se crearán de manera automática un grupo de carpetas entre las que cabe destacar la

carpeta www. Dentro de esta carpeta, podremos crear tantas subcarpetas como queramos para poner dentro nuestros ficheros PHP, que formarán nuestra aplicación web.

Una vez ya se ha instalado, podemos acceder al servidor mediante la url http://localhost o haciendo clic en el botón que aparecerá en el menú de Windows y seleccionando la opción localhost. Además, accediendo a la opción PhpMyAdmin, o la dirección http://localhost/phpmyadmin, se puede acceder a un sitio web para manejar las bases de datos de la aplicación.

3.1.2. Estructura

Para cubrir las necesidades del servidor para la aplicación, es decir, para permitir la creación e inicio sesión de usuarios y el envío de los datos de tráfico, se ha creado dentro del directorio www de wamp un subdirectorio llamado android_login_api, dentro del cual encontraremos los ficheros index.php y otro subdirectorio llamado include, dentro del cual encontraremos los ficheros config.php, DB_Connect.php y DB_Functions.php, que se explicarán a continuación.

Además, se ha creado la base de datos android_api con dos tablas, Users y Locations, donde se almacenarán los usuarios registrados en la aplicación y los datos de tráfico que estos envíen al servidor. Veremos la estructura de estas tablas y su creación en un capítulo posterior de esta misma sección.

3.2. MySQL

¿Qué es MySQL?

MySQL es el segundo sistema de manejo de bases de datos relacionales (RDBMS, Relational DataBase Management System) más utilizado en todo el mundo, y el primero entre los open-source. El proyecto MySQL ha ofrecido su código de manera abierta a través de la licencia GNU-GPL, así como bajo una serie de acuerdos de propiedad. Originalmente, el proyecto era propiedad de la compañía sueca MySQL AB (fundada por Davis Axmark, Allan Larsson y Michael Widenius), que ahora ha pasado a formar parte de Oracle. Además, para su uso propietario, hay varias ediciones de pago disponibles que ofrecen funcionalidades adicionales.

La primera versión de MySQL fue lanzada en mayo de 1995. Inicialmente tenía la intención de ser de uso personal de mSQL, basada en el lenguaje de bajo nivel ISAM, el cual los creadores consideraron después muy lento e inflexible. Crearon entonces una nueva interfaz SQL, manteniendo el API de mSQL. De esta manera, muchos desarrolladores que lo conocían podrían utilizarlo, evitando el pago de licencias que requería mSQL.

De la misma manera que PHP, su uso dentro de los sistemas 'AMP', le ha permitido ser una opción popular entre las bases de datos disponibles para su uso en aplicaciones web. MySQL también es usado en muchas webs de perfil alto y gran escala como Google, Facebook, Twitter…

Excepto para Windows, MySQL se distribuye sin herramientas gráficas para su administración. Los usuarios utilizan las herramientas incluidas de línea de comando o algún workbench de forma externa.

MySQL está escrito en C y C++, y su parser está escrito en yacc, aunque utiliza un analizador léxico propio. Además funciona en gran cantidad de plataformas, incluyendo Windows, Linux, OS X, AIX, Solaris, Symbian…

El software de servidor MySQL y las librerías cliente utilizan una distribución de licenciamiento dual. Son ofrecidas bajo la versión GPL o bajo una licencia propietaria. Ofrecen soporte a través del manual oficial. Además, diferentes canales de IRC y foros ofrecen soporte de manera gratuita. Por otra parte, Oracle ofrece soporte de pago a través de sus productos Enterprise, entre los cuales hay diferencias de precio y características. No obstante, también existen terceros que ofrecen soporte y servicios de diferentes maneras, entre los que destacan Sky SQL Ab y Percona.

Características

MySQL se ofrece bajo dos ediciones diferentes: la open source MySQL Community Server y la propietaria Enterprise Server. La versión Enterprise se diferencia en una serie de extensiones con licencia que se instalan como plug-ins del servidor, pero por todo lo demás comparte características y numerado de versiones con la versión libre.

- Escrito en C y en C++
- Probado con un amplio rango de compiladores diferentes
- Funciona en diferentes plataformas
- Proporciona sistemas de almacenamiento transaccionales y no transaccionales
- Un sistema de reserva de memoria muy rápido basado en threads
- Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verficación basada en el host
- Soporte a un amplio subconjunto de ANSI SQL 99
- Procedimientos almacenados
- Triggers y cursores
- Vistas actualizables
- Soporte para SSL

- Sub-SELECTs o SELECTs anidados
- Soporte para Unicode
- Soporte para múltiples motores de almacenamiento, permitiendo seleccionar el más efectivo para cada tabla en la aplicación.

No obstante, como otras bases de datos SQL, MySQL tiene una serie de limitaciones. Por ejemplo, no cumple el estándar completo para algunas de las funcionalidades implementadas, incluyendo referencias de clave ajena algunos motores de almacenamiento que no sean los que se ofrecen por defecto. Hasta la versión 5.7, los triggers están limitados a uno por acción, lo que significa que solo se puede definir un trigger para ejecutarse después de una operación INSERT y otro antes para una misma tabla. Además, no se pueden definir triggers para las vistas.

Sin embargo, MySQL puede ejecutarse en entornos de computación en la nube. Algunos modelos comunes de implantación para MySQL en la nube son Virtaul machine image, MySQL as a service y Managed MySQL cloud hosting.

3.2.1. Users

Esta será la tabla donde se almacenen los usuarios registrados en la aplicación y donde se comprobará que el usuario que está iniciando sesión existe y es válido. Para crearla se debe entrar a la dirección http://localhost/phpmyadmin con el servicio WAMP iniciado y seleccionar la pestaña SQL, que nos ofrecerá un editor para introducir sentencias.

En primer lugar ejecutaremos la siguiente consulta para crear la base de datos principal de nuestra aplicación:

```
create database android_api
```

De esta manera se creará la base de datos en la carpeta correspondiente del servidor y podremos empezar a definir las tablas.

Como ya se ha dicho, la tabla Users se encargará de almacenar los usuarios de la aplicación y contendrá 8 campos. Veamos primero la sentencia utilizada para crear la base de datos:

```
create table users(
        uid int(11) primary key auto_increment,
        unique_id varchar(23) not null unique,
        name varchar(50) not null,
        email varchar(100) not null unique,
        encrypted_password varchar(80) not null,
        salt varchar(10) not null,
```

```
    created_at datetime,
    updated_at datetime null);
```

Como se puede observar, uid será el identificador interno del usuario, y la clave primaria de la tabla. Será un campo autonumérico que aumentará cada vez que se cree un nuevo usuario.

El campo unique_id será el utilizado por la aplicación para diferenciar los usuarios, se define como not null y unique precisamente con este fin. Este campo será creado automáticamente por el servidor mediante la función MySQL uniqid(“”,true), la cual devuelve un resultado de tipo string, por eso el campo se declara como varchar(23). De esta manera, cada usuario tendrá un identificador único del tipo 4f074eca601fb8.88015924.

De la misma manera el nombre y el email se definen también como not null y el email además como unique, puesto que no se aceptarán dos usuarios con el mismo email, ya que es una herramienta personal. Estos dos últimos campos se definen como varchar, el tipo de datos SQL para cadenas de texto, con una longitud máxima de 50 y 100 caracteres respectivamente.

Cabe destacar que la contraseña se guardará en dos campos separados: encrypted_password y salt. Las contraseñas se almacenarán utilizando el método base64_encode. Es por esto que cada contraseña necesita dos columnas para almacenarse. La primera servirá para almacenar la contraseña encriptada y la otra para almacenar el salt utilizado para encriptar la contraseña.

Los dos últimos campos se utilizan para almacenar la fecha de registro de los usuarios y posteriores modificaciones a algunos de sus datos.

3.2.2. Locations

Esta es la tabla donde se almacenan los datos de tráfico enviados por los usuarios de la aplicación al servidor.

Para crearla se utiliza la siguiente instrucción SQL:

```
create table locations(
  User varchar(50),
    Poblacion varchar(50),
    Calle varchar(50) not null,
    Numero varchar(10),
    fechaHora datetime,
    Velocidad varchar(80) not null);
```

En esta tabla, entonces, se guardará una referencia al usuario que envía los datos, así como los parámetros de localización proporcionados por la geocodificación reversa realizada por su terminal, es decir, la población y la calle

y el número donde se encuentra, en caso de estar disponibles, puesto que el proceso de geocodificación puede devolver resultados muy variados.

También se almacenará la fecha y la hora en la que el usuario envía los datos para, a la hora de calcular rutas para el resto de usuarios, se tengan en cuenta únicamente los datos más recientes proporcionados.

Finalmente, se almacena la velocidad para que el servidor pueda hacerse una idea de la congestión de tráfico en la zona indicada.

Con todos estos datos, el servidor debería ser capaz de utilizarlos para calcular la ruta más rápida de manera más eficiente, teniendo en cuenta el tráfico en distintos lugares y en cada momento.

3.3. PHP

¿Qué es PHP?

PHP es un lenguaje de programación de scripts y server-side creado en 1995, pensado y diseñado para el desarrollo web pero también usado como lenguaje de programación de propósito general. La implementación referencia de Php fue originalmente creada por Ramsus Lerdorf en 1994, y hoy en día es producida por el PHP Group.

Originalmente las siglas significaban Personal Home Page, sin embago, hoy en día hacen referencia a Hypertext Preprocessor, una especie de acrónimo recursivo.

En enero de 2013, Php estaba instalado en más de 240 millones de sitios web, y 2,1 millones de servidores.

Una de las ventajas más grandes del código PHP es su facilidad para mezclarse con código HTML (HyperText Markup Language), y puede integrarse fácilmente con varios motores o frameworks web. El código normalmente es procesado por un intérprete de PHP, que normalmente está implementado como modulo nativo de los servidores web o como ejecutable CGI (Common Gateway Interface). Después de ser interpretado y ejecutado, el web server envía los resultados al cliente, normalmente como parte de la página web generada.

Además, PHP también ha evolucionado para incluir una interfaz de línea de comando o CLI (Command-line interface), por lo que también puede ser utilizado en aplicaciones gráficas independientes.

El intérprete de PHP canónico, impulsado por el Zend Engine, es un código libre licenciado con la PHP License. PHP ha sido ampliamente exportado y puede ser implantado en la mayoría de servidores web sobre cualquier sistema operativo y plataforma, siempre libre de cargos. Sin embargo, y a pesar de su popularidad,

hasta 2014 no había ningún estándar o especificación escrito para PHP, pero desde el pasado 2014 hay una iniciativa formal en marcha para escribir una serie de especificaciones php.

Versión

Actualmente, desde 2014 se está trabajando en una nueva versión mayor de lanzamiento llamada PHP7, cuya publicación se espera para noviembre de 2015. Sin embargo, y como la versión 6 nunca fue publicada debido a que el proyecto de integración de PHP con Unicode que iba a ser la versión 6 nunca se hizo público, actualmente las versiones que se utilizan son las versiones 5.4, 5.5 y 5.6.

Instalación y configuración

Existen dos grandes vías de añadir soporte PHP a un servidor: como un módulo nativo del servidor o como un CGI ejecutable, como ya hemos dicho. PHP tiene un módulo de interfaz directo llamado SAPI (Server Application Programming Interface), el cual es soportado por la mayoría de servidores web como Apache HTTP Server, Miscrosoft IIS, Netscpace o iPlanet. En caso de que PHP no disponga de módulo de soporte para un servidor, siempre se puede usar un procesador CGI o FastCGI, en cuyo caso el servidor se configura para utilizar este procesador para todos los ficheros PHP.

PHP también se puede utilizar para desarrollar aplicaciones de usuario de interfaz gráfica (GUI), utilizando la extensión PHP-GTK. Además, cuando se utiliza PHP en entornos de nube, son necesarios una serie de SDK (Software Development Kits), que son proporcionados para utilizar características específicas de la nube.

Uso

PHP, como ya se ha dicho con anterioridad, es un lenguaje de scripts de propósito general que encaja específicamente en el desarrollo web del lado del servidor. Todo código PHP en un fichero pedido es ejecutado por el runtime PHP, normalmente para crear contenido web dinámico o imágenes utilizadas en la web o en cualquier otro sitio. También puede ser utilizado para el scripting de línea de comando o para diseñar aplicaciones de usuario de interfaz gráfica.

PHP puede ser instalado en la mayoría de servidores web, en gran cantidad de sistemas operativos y plataformas, y puede ser utilizado con una enorme cantidad de sistemas de control de bases de datos relacionales. Además, la mayoría de proveedores de hosting web soportan el uso de PHP para sus clientes. Está disponible de manera gratuita, y el PHP Group proporciona el código fuente completo para que sus usuarios puedan construir, personalizar y extenderlo para su propio uso.

PHP actúa principalmente como filtro, recibiendo información de un fichero o un flujo de datos que contengan texto y/o instrucciones PHP y devolviendo otro flujo de datos saliente. Normalmente la respuesta consistirá en código HTML, aunque también se podría tratar de ficheros JSON (como es el caso del proyecto que se presenta en esta memoria), XML o ficheros de datos binarios, tales como imágenes o diferentes formatos de audio. Desde la versión PHP4, el parser de PHP, de manera similar a Java, produce un bytecode que, al ser interpretado por el Zend Engine, proporciona una mejora significativa de rendimiento sobre su predecesor.

Aunque originalmente fue diseñado para crear páginas web dinámicas, hoy en día PHP se centra principalmente en el scripting del lado del servidor, y es similar a otros lenguajes de este tipo tales como ASP.NET de Microsoft, JavaServer Pages de Sun... PHP también ha atraído el desarrollo de muchos frameworks de software que proporcionan los bloques de construcción y estructuras de diseño para promocionar el desarrollo ágil de aplicaciones (RAD).

La arquitectura LAMP se ha vuelto muy popular en la industria del desarrollo web como una manera de desarrollar aplicaciones. Paquetes similares como WAMP (que nos ocupa en este proyecto) o MAMP también están disponibles para cada sistema operativo, puesto que la primera letra de estos términos define el sistema operativo que soportan (Linux, Windows o Macintosh), y el resto viene de Apache, MySQL y PHP, aunque la P también puede significar Python o Perl.

3.3.1. Index.php

Este será el fichero de encauzar las diferentes peticiones que llegan al servidor dependiendo del parámetro tag con el que lleguen.

```
// get tag
$tag = $_POST['tag'];
```

Además, se incluirá una referencia al fichero DB_Functions.php que se explicará más adelante. Con la siguiente estructura se encauzan las diferentes peticiones tageadas como login, register o location, y se descartan las que lleguen con cualquier otro tag.

```
// check for tag type
if ($tag == 'login') {

}else if ($tag == 'register') {

}else if($tag == 'location') {

}else {

}
```

Así, en caso de recibir una petición de tipo 'login' se recuperará el usuario y contraseña enviados por el usuario junto a la petición y se enviarán al método getUserByEmailAndPassword de la clase DB_Functions para comprobar si existe el usuario. En caso de encontrar el usuario se devolverán sus datos indicando que no se ha encontrado ningún error en un fichero Json, típico formato para el intercambio de ficheros entre plataformas a través de internet. Si no se encuentra el usuario, en cambio, se devolverá un fichero Json con el campo error a True.

Por otro lado, si la petición llega al servidor con el tag 'register', el servidor recuperará los datos de nombre, email y contraseña enviados junto a la petición. Primeramente comprobará si el usuario ya existe, llamando al método isUserExisted() de DB_Functions, en cuyo caso devolverá un error indicándolo. En caso contrario, se llamará al método storeUser(), de nuevo de la clase DB_Functions, pasándole estos tres parámetros nuevamente. Si el usuario es almacenado de manera satisfactoria, se devolverá un fichero Json libre de errores con los datos con los que se ha creado el usuario, para ser interpretados por la aplicación "cliente". Sin embargo, si el usuario no se pudiese crear, se devolvería un nuevo fichero Json con un error de creación de usuarios.

Finalmente, si la petición viene marcada como 'location', se recuperarán los datos enviados por la aplicación (población, calle, número, fecha, hora y velocidad) y se almacenarán llamando al método storeUserLocation() para ser utilizados en un futuro para la creación de rutas más rápidas y mejor optimizadas. Si la localización es almacenada con éxito se le contestará a la aplicación con un fichero Json y si no, se enviará un fichero conteniendo un error indicando que no se ha podido almacenar la información, para que el usuario compruebe que puede estar fallando.

Sin embargo, si la petición llega con cualquier otro tag o con ausencia de este campo, será descartada y se notificará al cliente a través de un nuevo fichero Json.

Los errores que responde el servidor mediante los ficheros Json son utilizados por la aplicación para ser mostrados al usuario a través de un Toast.

3.3.2. Include

En esta carpeta encontramos ficheros adicionales del servidor, tanto de configuración,como de establecimiento de la conexión con la base de datos, como la clase que contiene los métodos que serán llamados por index.php y que proporcionan la funcionalidad real a la aplicación.

3.3.2.1. Config.php

Esta clase contiene los parámetros principales de configuración del servidor. Contendrá el nombre del host, que para las pruebas ha sido la máquina local o localhost, puesto que se han realizado siempre en red de área local. Además, se guardará aquí el usuario y su contrsaseña de la base de datos, en nuestro caso se ha optado por la opción “root”-“root”. Además, guardaremos el nombre de la base de datos sobre la que va a trabajar el servidor, que debe haber sido creada en el servidor desde PhpMyAdmin.

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

3.3.2.2. DB_Connect.php

Esta es la clase encargada de crear la conexión del servidor PHP con la base de datos MySQL. Para ello bastará con incluir una referencia a la clase Config.php, crear la conexión (utilizando los parámetros de nombre de host, usuario y contraseña de la base de datos definidos en el fichero de configuración), seleccionar la base de datos que se utilizará (definido también en el fichero de configuración) y devolver la conexión, puesto que el método que la devuelve habrá sido llamado desde DB_Functinos.php.

```
<?php
    // Connecting to database
    public function connect() {
        require_once 'include/Config.php';
        // connecting to mysql
        $con = mysql_connect(DB_HOST, DB_USER, DB_PASSWORD) or
die(mysql_error());
        // selecting database
        mysql_select_db(DB_DATABASE) or die(mysql_error());
        // return database handler
        return $con;
    }
}
?>
```

3.3.2.3. DB_Functions.php

Se puede considerar que esta es la clase principal de los ficheros del servidor. En ella se incluyen todos los métodos que se encargarán de interactuar con la base de datos, ya sea para recuperar datos o para almacenar nuevos registros, ya sean de usuarios o de localizaciones. Además, también se podrá comprobar si el usuario existe cuando se reciban peticiones de inicio de sesión.

Para ello contiene una serie de métodos vitales para proporcionar estas funcionalidades.

```
public function storeUser($name, $email, $password) {
    $uuid = uniqid('', true);
```

```
        $hash = $this->hashSSHA($password);
        $encrypted_password = $hash["encrypted"]; // encrypted password
        $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())");
        // check for successful store
        if ($result) {
            // get user details
            $uid = mysql_insert_id(); // last inserted id
            $result = mysql_query("SELECT * FROM users WHERE uid = $uid");
            // return user details
            return mysql_fetch_array($result);
        } else {
            return false;
        }
    }
```

El método storeUser(), como su propio nombre indica, se encarga de almacenar nuevos usuarios registrados en la base de datos mediante una sentencia de tipo INSERT. Devolverá el resultado de la sentencia en caso de funcionar y falso en caso de que se obtenga algún error.

```
    public function getUserByEmailAndPassword($email, $password) {
        $result = mysql_query("SELECT * FROM users WHERE email =
'$email'") or die(mysql_error());
        // check for result
        $no_of_rows = mysql_num_rows($result);
        if ($no_of_rows > 0) {
            $result = mysql_fetch_array($result);
            $salt = $result['salt'];
            $encrypted_password = $result['encrypted_password'];
            $hash = $this->checkhashSSHA($salt, $password);
            // check for password equality
            if ($encrypted_password == $hash) {
                // user authentication details are correct
                return $result;
            }
        } else {
            // user not found
            return false;
        }
    }
```

Este nuevo método se utilizará para recuperar los datos del usuario y devolverlos a la aplicación.

```
    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) {
            // user exists
            return true;
        } else {
            // user not exists
            return false;
```

```
    }
}
```

Este será el método encargado de comprobar que un usuario existe cuando se recibe una petición de inicio de sesión. Para ello, se realiza una sentencia SQL que busca el email entre todos los usuarios. Si se encuentra algún registro coincidente se considera que el usuario existe y se devuelve True y en caso contrario se devuelve False.

```
public function storeUserLocation($name, $poblacion, $calle, $numero,
$fechaHora, $velocidad){
    $result = mysql_query("INSERT INTO Locations(User, Poblacion,
Calle, Numero, fechaHora, Velocidad) VALUES('$name', '$poblacion',
'$calle', '$numero', '$fechaHora', '$velocidad')");
    if($result){
        return true;
    }
    else{
        return false;
    }
}
```

Este último método será el encargado de introducir los datos de tráfico recibidos de la aplicación en la base de datos del servidor gracias a una sentencia INSERT que incorpora los datos recibidos de usuario, población, calle, número, fecha, hora y velocidad.

4. Pruebas y desarrollo

Pruebas y terminales

Cabe destacar que las pruebas, a pesar de haber sido completamente satisfactorias, no han podido ser completamente exhaustivas, al tener el servidor en la misma red local que el dispositivo para facilitar la comunicación entre ambos dispositivos, puesto que no se ha dispuesto de terminales Android con tarifa de conectividad de datos 3G/4G.

Las pruebas han sido realizadas con los siguientes terminales:

- Samsung Galaxy S4. Versión de Android 5.1.1 y API 21.
- Samsung Galaxy Tab3 Lite con versión de Android 4.2.2 y API 17.
- Dispositivo personalizado con versión de Android 5.0.1 y API 19, emulado por el AVD Manager del que hemos halado anteriormente.

Proceso de desarrollo

Para el desarrollo de la aplicación se ha utilizado un modelado de prototipos, es decir, un modelo conocido dentro de los modelos de desarrollo evolutivo. Se ha desarrollado en varias etapas, proporcionando cada ellas una retroalimentación y mejoras para la siguiente, avanzando y corrigiendo los errores de código de forma gradual. Este tipo de diseño permite al usuario y al desarrollador (en este caso la misma persona) entender mejor las funcionalidades de la aplicación y cómo se debe comportar así como ver resultados a corto plazo.

En primer lugar se desarrollaron las pantallas y vistas para el registro de usuarios y el inicio de sesión, así como la pantalla de bienvenida desde la que se accede al navegador. Una vez realizadas estas pantallas, se comprobó que funcionasen correctamente en distintas resoluciones y también sobre distintas versiones de Android, puesto que el desarrollo se inició pensando en la última versión de Android pero luego se adaptó para que funcionase también a partir de la versión 4.2.2.

A continuación, se procedió a la instalación y configuración del servidor. Se creó la base de datos y la tabla de usuarios. Además, se comenzó el desarrollo de las funciones principales del servidor, las que aceptarían las peticiones de los terminales para registrar usuarios e iniciar sesión. El desarrollo de estas funciones no resultó sencillo, puesto que nunca antes había utilizado PHP ni MySQL, lo cual me hizo perder más tiempo del esperado. Una vez me familiaricé con PHP, se empezó a desarrollar las funciones propiamente dichas.

Inicialmente, se separaron las peticiones de registro e inicio sesión utilizando un TAG: ‘login’ para el inicio de sesión de los usuarios y ‘register’ para el registro de nuevos usuarios. Posteriormente se añadió el TAG ‘location’ para encauzar

correctamente las peticiones de inserción de datos de localización y velocidad de los usuarios.

Las peticiones se encauzan desde el fichero index.php del servidor, mientras que las funciones que realmente proporcionan al servidor la funcionalidad necesaria se encuentran dentro del fichero DB_Functions.php. Además, se crearon los ficheros Config.php y DB_Connect.php para configurar los parámetros necesarios del servidor (usuario y contraseña para acceder a las bases de datos), así como para establecer la conexión con la base de datos.

Creación de usuarios e inicio de sesión

Una vez creados los ficheros del servidor, se realizaron las primeras pruebas de creación de usuarios, por lo que se puso en marcha el servidor y se configuró una red local a la que conectar tanto el servidor como el terminal desde el que iba a realizar las peticiones. Para ello se utilizó un método un tanto rudimentario, puesto que no había mejores opciones disponibles. Se utilizó un terminal móvil como router para conectar tanto el servidor como el terminal a través de una red local. Para iniciar las pruebas, lo único que hacía falta era configurar las direcciones IP a las que se conectaría el terminal, las cuales están configuradas en el fichero AppConfig.java del paquete es.upv.disca.osclient.login.config.

El principal problema que apareció aquí fue un conflicto con el acceso a la base de datos MySQL desde php, de manera que la aplicación funcionaba y los usuarios eran creados en la base de datos pero el servidor devolvía un error de la versión de MySQL utilizada, pidiendo utilizar eMySQL. Sin embargo, al actualizar todos los ficheros del servidor a esta versión o adaptación de MySQL el error se propagó y se hizo más grande, por lo que, ase optó por la opción de establecer un *workaround* desde la aplicación que evitara entender el código devuelto como error y seguir la ejecución normal.

Una vez realizadas las pruebas y comprobado que el registro de usuarios y el inicio de sesión funcionaban correctamente, se procedió a dedicar un poco de tiempo a mejorar el aspecto visual de la aplicación, añadiendo algunas funcionalidades. La primera de ellas fue la de añadir las animaciones de cambio de pantalla entre la de inicio de sesión y la de registro de usuarios. Posteriormente se añadió el efecto de deslizar para cambiar entre las dos pantallas y finalmente la opción que permite visualizar la contraseña escrita pulsando en el ojo que aparece en su cuadro de texto, y volver a ocultarla volviendo a pulsar. Finalmente se agregó la opción de recordar el usuario que había iniciado sesión para pasar directamente a la pantalla de bienvenida, así como personalizar el efecto de pulsación en el CheckBox que indica si se quiere recordar o no el último inicio de sesión.

Envío de información de tráfico al servidor

Una vez concluidas las pruebas de creación de usuarios y su inicio de sesión, se empezaron a añadir a la aplicación las funcionalidades necesarias para el envío de datos al servidor.

Para ello, se importó el proyecto previo del alumno David Peris Martínez en Eclipse y se procedió a hacer las modificaciones oportunas sobre el método onLocationChanged(), que es el evento que se lanza cuando el gps del terminal recibe una actualización de su posición (latitud y longitud).

Una vez finalizado el desarrollo en Android, se modificaron los ficheros del servidor. Como ya se ha dicho anteriormente, se añadió la opción del TAG 'location' al fichero index.php del servidor, así como las funciones necesarias para guardar los datos recibidos en el servidor en el fichero DB_Functions.php.

Para realizar las pruebas se utilizó el mismo sistema que con la creación y registro de usuarios, utilizar un terminal como punto de acceso para el servidor y el terminal con la aplicación. Por esta razón, los resultados de las pruebas no son totalmente concluyentes porque no se ha podido testear en "campo abierto", pero al no disponer de terminal Android con acceso a la red, se tuvo que hacer uso de este sistema.

Sin embargo, cabe destacar que esta funcionalidad funcionó correctamente desde el principio y fue testeada en diversas ubicaciones para comprobar que el proceso de geocodificación reversa funcionaba como era de esperar, por lo que los resultados fueron satisfactorios.

5. Conclusiones

Me gustaría destacar que los objetivos del proyecto se han cumplido con éxito y que todo ha funcionado correctamente desde las primeras etapas del proyecto, aunque aun así ha sufrido modificaciones para corregir pequeños errores y para proporcionar mejor funcionalidad y mejoras de rendimiento.

Además, la realización de este proyecto me ha servido para ampliar mis horizontes informáticos, puesto que no había cursado ninguna asignatura de programación web y mis conocimientos de bases de datos eran limitados. Así pues, haber trabajado con PHP y MySQL me ha ayudado en estos campos, lo cual creo que me ayudará ampliamente a lo largo de mi carrera profesional.

Por otra parte, aunque ya conocía la metodología de programación y el lenguaje Java, así como el SDK de Android, nunca había dedicado tiempo a aprender un poco de diseñó ni a realizar interfaces modernas y con un aspecto visual trabajado. Esto me llevó a fijarme en los estándares publicados por Google en su manual de "Material Design", con lo que pude aprender muchas cosas sobre el aspecto visual de las aplicaciones y patrones para utilizar en mis aplicaciones.

Con trabajo futuro, habría que ampliar la aplicación para trabajar contra un servidor real que aceptase una mayor cantidad de peticiones simultáneas de manera fluida y finalmente habría que diseñar el servidor que utilizase la información creada por la aplicación para calcular rutas de manera más eficiente.

Finalmente, me gustaría destacar que he aprendido mucho con la realización de este proyecto y que he disfrutado aprendiendo, puesto que me encanta la programación de aplicaciones y, sobretodo, me ha gustado aprender un poco de diseño e implementación en Android.

6. Bibliografía

Desarrollar apps Android en Eclipse o en Android Studio. BetaBeers. 26 de febrero de 2015. Disponible en http://betabeers.com/forum/desarrollar-apps-android-eclipse-o-android-studio-1284/

Eclipse. Wikipedia. 30 de junio de 2015. Disponible en https://es.wikipedia.org/wiki/Eclipse_(software)

Eclipse. Eclipse. Disponible en http://www.eclipse.org/home/index.php

Eclipse help. Eclipse. Disponible en http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2FFint_eclipse.htm

GNU General Public License. Wikipedia. 11 de julio de 2015. Disponible en http://es.wikipedia.org/wiki/GNU_General_Public_License

Material design. Google. Disponible en https://www.google.com/design/spec/material-design/introduction.html#

Java. Wikipedia. 10 de julio de 2015. Disponible en https://en.wikipedia.org/wiki/Java_(programming_language)

Android SDK. AndroidPit. Android para Principiantes. Disponible en http://www.androidpit.es/sdk-android

Android SDK. Wikipedia. 12 de julio de 2015. Disponible en http://es.wikipedia.org/Desarrollo_de_programas_para_Android#Android_SDK

Android APIs. Android developers. Disponible en http://developer.android.com/about/versions/android-4.2.html

Android versions. Wikipedia. 13 de julio de 2015. Disponible en http://en.wikipedia.org/wiki/Android_version_history

Ciclo de vida de una actividad en Android. Androidcurso. Disponible en http://www.androidcurso.com/index.php/tutoriales-android/37-unidad-6-multimedia-y-ciclo-de-vida/158-ciclo-de-vida-de-una-actividad

Managing the activity lifecycle. Android developers. Disponible en http://developer.android.com/training/basics/activity-lifecycle/index.html

Activity. Android developers. Disponnible en http://developer.android.com/reference/Android/app/Activity.html

Activities. Android developers. Disponible en http://android.developer.com/guide/components/activities.html

El concepto de Android Intent. Arquitectura Java. Disponible en http://www.arquitecturajava.com/el-concepto-de-android-intent/

Intents. Android developers. Disponible en http://android.developer.com/guide/components/intents-filters.html

Toasts. Android developers. Disponible en http://android.developer.com/guide/topics/ui/notifiers/toasts.html

SQLiteOpenHelper. Android developers. Disponible en http://android.developer.com/reference/android/database/sqlite/SQLiteOpenHelper.html

Bases de datos en android. SGOliver. Disponible en http://www.sgoliver.net/blog/bases-de-datos-en-android-i-primeros-pasos/

Application. Android developers. Disponible en http://android.developer.com/reference/android/app/Application.html

Extending the Android Application class. Devahead. Disponibl en http://www.devahead.com/blog/2011/06/extending-the-android-application-class-and-dealing-with-singleton/

Android Application Class. Intertech. Disponible en http://www.intertech.com/Blog/androids-application-class/

Volley. Android googlesource. Disponible en https://android/googlesource.com/platform/frameworks/volley

Geocoder. Android developers. Disponible en http://android.developer.com/reference/android/location/Geocoder.html

JSON. Wikipedia. 5 de mayo de 2015. Disponible en https://es.wikipedia.org/wiki/JSON

JSONException. Android developers. Disponible en http://android.developer.com/reference/org/json/JSONException.html

JSONObject. Android developers. Disponible en http://android.developer.com/reference/org/json/JSONObject.html

RequestQueue. Android developers. Disponible en http://android.developer.com/training/volley/requestqueue.html

Curso de programacion Android. SGOliver. Disponible en http://www.sgoliver.net/blog/curso-de-programacion-android/indice-de-contenidos/

WAMP. Wikipedia. 20 de octubre de 2014. Disponible en https://es.wikipedia.org/wiki/WAMP

LAMP. Wikipedia. 8 de junio de 2015. Disponible en https://en.wikipedia.org/wiki/LAMP_(software_bundle)#Variants_and_equivalents_on_other_platforms

PHP. Wikipedia. 12 de julio de 2015. Disponible en https://en.wikipedia.org/wiki/PHP

PHP5 Tutorial. W3Schools. Disponible en http://www.w3schools.com/php/

WampServer. Wampserver. Disponible en http://www.wampserver.com/en/

MySQL. Wikipedia. 26 de junio de 2015. Disponible en https://en.wikipedia.org/wiki/MySQL

Modelo de prototipos. Wikipedia. 26 de mayo de 2015. Disponible en https://es.wikipedia.org/wiki/Modelo_de_prototipos

7. Anexos

Anexo 1. Configuración del servidor

Para configurar de manera rápida y eficaz el servidor tendremos que seguir los siguientes pasos.

1. Descargar WampServer desde http://wampserver.com/en/
2. Lanzar el ejecutable descargado y realizar la instalación. Bastará con hacer clic en siguiente aceptando las opciones que se ofrecen por defecto durante la instalación.
3. A continuación deberemos dar permiso de acceso a terminales externos al servidor. Para ello vamos a C:/wamp/bin/apache/apache2.4.9/conf y editamos el fichero httpd.conf añadiendo las siguientes filas en el sub-apartado <directory>.
   ```
   Order deny,allow
   Allow from all
   ```
   Además, también añadiremos las siguientes líneas en el sub-apartado <directory c:/wamp/www>
   ```
   Require all granted
   Order deny,allow
   Allow from all
   ```
4. A continuación, deberemos indicar que la contraseña para el usuario 'root'@'localhost' es 'root'. Para ello, accedemos a localhost/phpmyadmin y seleccionamos la pestaña de usuarios. Buscamos el usuario root de localhost y lo editamos para establecer su contraseña como 'root'.
5. Ahora, sin embargo, al volver a acceder a localhost/phpmyadmin nos darña un error `#1045 - Access denied for user 'root'@'localhost' (using password: NO).` Puesto que hemos cambiado la configuración. Para solucionar este error deberemos editar el fichero config.inc.php

localizado en C:/wamp/apps/phpmyadmin4.1.14 indicando que la contraseña es 'root'.

```
$cfg['Servers'][$i]['password'] = 'root';
```

6. Ahora, lo único que queda por hacer en el servidor es crear la base de datos Android_api y las tablas users y locations como ya hemos visto en un apartado anterior de esta memoria.

Anexo 2. Configuración de los parámetros de la aplicación

Para configurar la aplicación para que apunte correctamente al servidor donde están alojadas las bases de datos deberemos dirigirnos a la clase **AppConfig** dentro del paquete **es.upv.disca.osmclient.login.config** e indicar cuáles serán las URLs de inicio de sesión y de registro de usuarios.

```
public static String URL_LOGIN = "http://192.168.43.178/android_login_api/";

public static String URL_REGISTER = "http://192.168.43.178/android_login_api/";
```

Para saber a qué dirección debemos apuntar, en caso de estar en la misma red local que el servidor, deberemos lanzar una consola en el servidor y ejecutar el comando ipconfig que nos proporcionará la dirección IPv4 del mismo, la cual utilizaremos en las líneas especificadas anteriormente.

En caso de que debamos apuntar a un servidor en la nube, podremos consultar nuestra IP con nuestro ISP o utilizando la web http://www.cualesmiip.com por ejemplo.