UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UNA APLICACIÓN PARA LA CONSULTA DE CONSUMO ELÉCTRICO A TRAVÉS DE DISPOSITIVOS MÓVILES

AUTOR: ALEMANY IBOR, SERGIO

TUTOR: GIL GÓMEZ, JOSE ANTONIO

Curso Académico: 2014-15

RESUMEN

Se ha desarrollado una aplicación para dispositivos móviles y tabletas Android que permite a los clientes de una determinada empresa de comercialización eléctrica poder acceder tanto a los detalles de su tarifa y consumos, así como a las últimas facturas emitidas. También se aportan consejos sobre cómo ahorrar en la factura eléctrica. Todo esto ha sido desarrollado con el programa informático Android Studio.

El programa ha sido desarrollado desde cero, desde su diseño hasta la implementación de cada uno de los módulos en las que se puede dividir el trabajo: Implementación del diseño mediante Layouts; conexión a la base de datos y tratamiento de mensajes obtenidos en formato JSON; hilos de ejecución para evitar cuelgues en las conexiones a la base de datos; simulación de tarifas eléctricas; creación de gráficas de consumo; y otras funciones secundarias como pueden ser implementar mapas de Google o añadir listas dinámicas.

Se ha desarrollado también una arquitectura WAMP con el que almacenar la base de datos necesaria para poder llevar a cabo el trabajo utilizando el programa WampServer. Se han utilizado unos datos de muestra proporcionados por la empresa colaboradora y se han implementado las tablas necesarias.

Las consultas a la base de datos se realizan a través de Internet accediendo a una serie de archivos PHP que realizan consultas predeterminadas sobre la base de datos. Estos archivos también han sido creados desde cero. Reciben unas variables mediante la codificación HTTP y devuelven los datos obtenidos por la consulta sobre la base de datos codificados en el formato JSON.

Palabras clave: Android, Studio, aplicación, móvil, consulta, consumo, eléctrico, factura, tarifa, WampServer, phpMyAdmin, base, datos, PHP, JSON, HTTP

RESUM

S'ha desenvolupat una aplicació per a dispositius mòbils i tabletes Android que permet als clients d'una determinada empresa de comercialització elèctrica poder accedir tant als detalls de la seua tarifa i consums, així com a les últimes factures emeses. També s'aporten consells sobre com estalviar en la factura elèctrica. Tot açò ha sigut desenvolupat amb el programa informàtic Android Studio.

El programa ha sigut desenvolupat des de zero, des del seu disseny fins a la implementació de cada un dels mòduls en què es pot dividir el treball: Implementació del disseny per mitjà de Layouts; connexió a la base de dades i tractament de missatges obtinguts en format JSON; fils d'execució per a evitar errors en les connexions a la base de dades; simulació de tarifes elèctriques; creació de gràfiques de consum; i altres funcions secundàries com poden ser el implementar mapes de Google o afegir llistes dinàmiques.

S'ha desenvolupat també una arquitectura WAMP amb el que emmagatzemar la base de dades necessàries per a poder dur a terme el treball utilitzant el programa WampServer. S'han utilitzat unes dades de mostra proporcionats per l'empresa col·laboradora i s'han implementat les taules necessàries.

Les consultes a la base de dades es realitzen a través d'Internet accedint a una sèrie d'arxius PHP que realitzen consultes predeterminades sobre la base de dades. Aqestos arxius també han sigut creats des de zero. Reben unes variables per mitjà de la codificació HTTP i tornen les dades obtingudes per la consulta sobre la base de dades codificades en el format JSON.

Paraules clau: Android, Studio, aplicació, consulta, consum, elèctric, factura, tarifa, WampServer, phpMyAdmin, base, datos, PHP, JSON, HTTP

ABSTRACT

There has been developed an application for mobile devices and Android tablets that allows customers of a certain electric marketing company access both the details of their tariff and electrical consumption, as well as their latest bills. It also provides some tips on how to save money on the electric bill. All of this has been developed with the Android Studio software.

The program has been developed from scratch, from the design to the implementation of each of the modules in which the work can be divided: implementation of the design by Layouts; connection to the database and processing of messages in JSON format; threads to avoid crashes in the connections to the database; simulation of electricity rates; creation of graphs of consumption; and other secondary functions such as implementing google maps or add dynamic lists.

A WAMP architecture with which store data base needed to carry out work using the WampServer has been also developed. A few sample data provided by the partner company have been used and necessary tables in the database has been implemented.

The queries to the database are made via Internet accessing a set of PHP files that perform default database queries. These files have also been created from scratch. They receive a few variables using HTTP encoding and return the data obtained by the query encoded in JSON format.

Keywords: Android, Studio, application, mobile, query, electric, bill, tariff, WampServer, phpMyAdmin, database, PHP, JSON, HTTP

DOCUMENTOS CONTENIDOS EN EL TFG

- Memoria
- Manual de usuario
- Presupuesto

ÍNDICE DE LA MEMORIA

ÍNDICE DEL MANUAL DE USUARIO

ÍNDICE DEL PRESUPUESTO

MEMORIA

1. ESTRUCTURA DEL DOCUMENTO

El documento introduce progresivamente las nociones necesarias para la comprensión del trabajo realizado.

En el apartado 2 se introducen los objetivos del trabajo y la motivación para su realización. El apartado 3 describe el ámbito de aplicación que cubrirá el trabajo. El apartado 4 analiza brevemente la existencia de aplicaciones similares. El apartado 5 repasa la normativa aplicable al trabajo.

El apartado 6 describe todo el trabajo realizado para llevar a cabo el trabajo. Este apartado contiene también una introducción teórica con todas las definiciones necesarias para la comprensión del trabajo. El apartado 7 enumera los trabajos que se podrían aplicar para mejorar la aplicación creada.

2. MOTIVACIÓN Y OBJETIVOS DEL TRABAJO

El consumo energético global ha estado creciendo exponencialmente desde la revolución industrial. Gran parte de esta energía proviene del consumo de electricidad, y contribuye en gran medida, al aumento del cambio climático, puesto que su producción emite una gran cantidad de gases de efecto invernadero.

Es necesario pues, una reducción de la emisión de gases de efecto invernadero, y una de las posible formas de conseguirlo es reduciendo el consumo eléctrico en la medida de lo posible.

Adicionalmente, este aumento del consumo energético global está provocando el inicio de una crisis energética global. Esto supone que cada vez es más difícil satisfacer la demanda energética, con el consecuente aumento del precio de esta.

Por tanto, el ahorro y la eficiencia energética adquieren una gran importancia en cuanto es necesario reducir la crisis energética que se está produciendo. Para ello, es de vital importancia que los consumidores de energía eléctrica tengan mayor conocimiento de sus consumos eléctricos.

El objetivo del trabajo es el de crear una aplicación para dispositivos móviles y tabletas Android que permita a los clientes de una empresa de comercialización eléctrica poder acceder tanto a los detalles de su tarifa y consumos, así como a las ultima facturas emitidas. También se aportarán consejos sobre cómo ahorrar en la factura eléctrica. Con esto, el usuario de la empresa tendrá un mayor conocimiento de su consumo eléctrico y le permitirá tanto el ahorro en su factura eléctrica como el consecuente ahorro energético correspondiente.

Todo el proceso se hará con la colaboración de la empresa Suministros Especiales Alginetenses. Se trata de una empresa comercializadora y distribuidora eléctrica que opera principalmente en la localidad de Alginet, situado en la provincia de Valencia. Esta empresa proporcionará los datos de consumo necesarios para que la aplicación pueda desarrollarse y probarse correctamente.

3. ÁMBITO DE APLICACIÓN

La aplicación será desarrollada para los dispositivos móviles con sistema operativo Android y les será de utilidad para los clientes que tengan contratada una tarifa de baja tensión menor a 450KW con la empresa Suministros Especiales Alginetenses.

Esta selección engloba a las tarifas 2.0A, 2.0DHA, 2.1A, 2.1DHA y 3.1DHA, quedando fuera de ella las tarifas 3.1A y 6.1. Estas 2 tarifas son minoritarias, es decir, la proporción de contratos con alguna de estas tarifas es muy pequeña puesto que están orientadas a grandes empresas con grandes consumos.

Por tanto, esta aplicación le será de utilidad a más del 90% del total de clientes de dicha empresa.

4. ESTADO DEL ARTE Y ANTECEDENTES

Algunas grandes empresas comercializadoras de energía eléctrica como Iberdrola ya poseen aplicaciones con funciones similares a las de la aplicación objetivo en este trabajo. Esta aplicación permite ver las facturas y los consumos de los clientes, pero no incluye otras funcionalidades como la simulación de tarifas para comprobar cuál es la más recomendable para el usuario, o la vista de consumos en un periodo personalizable.

A pesar de ello, la empresa colaboradora no posee ningún medio por el cual los usuarios puedan mantener un control sobre sus consumos, por lo que el añadido de esta aplicación supone una gran mejora del servicio para sus clientes.

5. NORMATIVA

Para poder desarrollar una aplicación Android hay que tener en cuenta diversos aspectos legales para evitar cualquier tipo de sanción. Dependiendo de las funcionalidades de la aplicación, esta deberá cumplir unas normativas específicas. Por ejemplo, si la aplicación ha de publicarse en la Google Play, esta deberá cumplir con la normativa pertinente. A continuación se muestra una descripción de los aspectos que se han tenido en cuenta a la hora de desarrollar la aplicación objeto del trabajo:

5.1. Derechos de los recursos utilizados

Es necesario contar con las licencias de todos los recursos utilizados en la aplicación, ya sean librerías de programación, bases de datos, elementos gráficos, melodías, textos, etc.

Todos los recursos utilizados en la aplicación desarrollada son de distribución gratuita, incluyendo los SDK de Android, la arquitectura WampServer para crear bases de datos, y la librería de gráficas GraphView.

También se han utilizado datos de clientes cedidos por la empresa Suministros Especiales Alginetenses, para poder desarrollar la aplicación de forma eficiente. Estos datos también han sido totalmente cedidos por la propia empresa.

5.2. Google Play

Para poder publicar aplicaciones en la tienda de aplicaciones Google Play, es necesario cumplir con una normativa obligada:

- Acuerdo de distribución para desarrolladores de Google Play. Disponible en https://play.google.com/intl/ALL_es/about/developer-distribution-agreement.html
- Políticas del Programa para Desarrolladores de Google Play. Disponible en https://play.google.com/intl/ALL_es/about/developer-content-policy.html

5.3. Licencia y condiciones de uso

Es imprescindible que el usuario acepte con antelación las licencias y condiciones de uso necesarias para poder hacer uso de la aplicación. Estas condiciones deberán ajustarse a la legislación vigente y eximirán al desarrollador de tantas responsabilidades como sean posible. Esto será de gran utilidad para el desarrollador como parte de una defensa ante cualquier reclamación que se puedan realizar por un mal uso de la aplicación.

En el caso de la aplicación a desarrollar, en el momento de la instalación de la aplicación se incluirán los permisos de todos los servicios que vaya a utilizar la aplicación, como puede ser el caso del acceso a internet, por ejemplo.

5.4. Informar al usuario

La aplicación objeto de trabajo se podría considerar como "servicio de la sociedad de la información, por lo que hay que cumplir con la normativa de estos servicios que correspondería a la "Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico". Por tanto, la principal obligación de estos servicios es la de informar a los usuarios sobre los aspectos marcados por la ley. Esto se puede indicar en las condiciones legales o en secciones de la aplicación como pueden ser las pantallas de "acerca de" o "quienes somos".

Estos apartados aportan información al usuario respecto a quien esa detrás de la aplicación móvil. En la aplicación que se va a desarrollar, se añadirá una pantalla “como encontrarnos” en la que se añadirá la dirección de las oficinas centrales y teléfono y e-mail de contacto.

5.5. Ley de protección de datos

Según la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter personal, la aplicación creada dispondrá de una pantalla de acceso con el que los usuarios se identificaran mediante un usuario y contraseña.

Si la aplicación llegara a publicarse, también se debería implementar un cifrado de los datos enviados y recibidos por el servidor utilizado.

6. DESARROLLO DEL TRABAJO

6.1. Introducción:

6.1.1. ¿Qué es Java?

Java es un lenguaje de programación orientado a objetos y de propósito general. La sintaxis de Java está muy inspirada en la del lenguaje de programación C/C++. Es un lenguaje interpretado y compilado, es decir, JAVA se pasa primero a un bytecode que luego es interpretado. El bytecode sería un intermedio entre la programación en lenguaje Java y el código máquina final.

Los programas escritos en Java se ejecutan en una Máquina Virtual de Java (JVM) desarrollada con el objetivo de dar una abstracción de la CPU de la máquina que ejecuta el programa. De esta forma, todo programa escrito en Java es interpretado por una JVM que realiza una conversión de dicho código a un código particular de la CPU utilizada, lo que permita a los programas escritos en Java ser independientes del sistema operativo usado.

Android incorpora una máquina virtual de Java para la ejecución de las aplicaciones, pero no utiliza la JVM oficial para interpretar el código generado. En su lugar, Android usa una versión de la JVM modificada, adaptada y optimizada para dispositivos móviles, esta máquina virtual se llama Dalvik.

Además, Android ofrece al desarrollador un Kit de Desarrollo de Software (SDK) escrito en Java que le permite acceder a todas las librerías para desarrollar programas que se ejecutan en el dispositivo móvil.

6.1.2. ¿Qué es Android?

Android es un sistema operativo basado en el núcleo Linux diseñado principalmente para smartphones y tablets. Es un sistema operativo libre, gratuito y multiplataforma. Se puede observar la arquitectura de Android en la siguiente imagen:

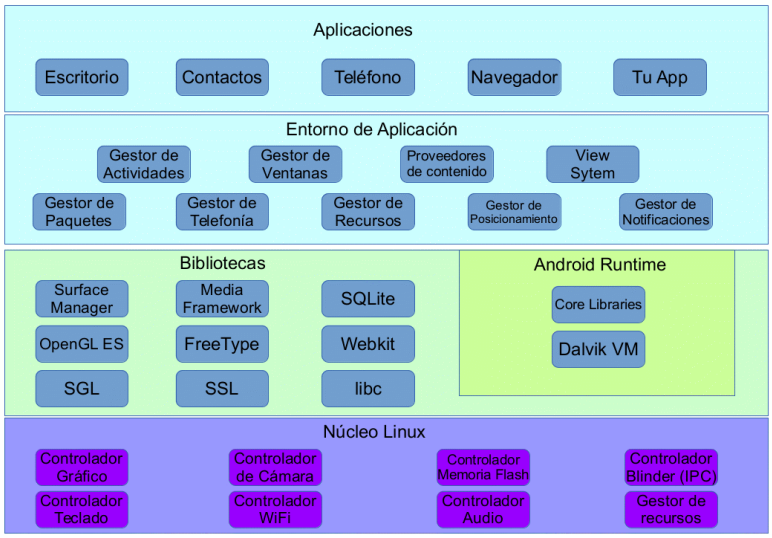


Ilustración 1: Arquitectura de Android. Fuente: persefoneblog.wordpress.com

Como se puede observar, el sistema Android puede ser dividido en cuatro capas o partes relacionadas entre sí:

Aplicaciones: La capa de aplicación es la capa más alta del sistema y es donde se encuentran las aplicaciones nativas del sistema operativo (como el teléfono, los contactos, etc) u otras aplicaciones desarrolladas por terceros y escritas en Android.

Entorno de aplicación: Las aplicaciones que se encuentran en el Entorno de Aplicación han sido programadas haciendo uso de un framework o SDK proporcionado por Android y escrito en Java. Este SDK provee al desarrollador de todo lo necesario para acceder a la funcionalidad del dispositivo móvil.

Bibliotecas/Android Runtime: Las funciones del framework o SDK de Android pueden hacer uso de las librerías de más bajo nivel como SQLite, OpenGL o SSL. El código creado mediante el uso de las funciones del SDK es compilado, optimizado y posteriormente interpretado por la máquina virtual Dalvik. Posteriormente, dicha máquina virtual se encarga de convertir el código a instrucciones entendibles por el procesador del dispositivo móvil.

Núcleo Linux: La última capa del sistema se encarga del tratamiento de los recursos propios del sistema operativo. Ejemplos de esto pueden ser la gestión de memoria o la administración y monitorización de procesos.

6.1.3. XML

Para poder desarrollar en Android se necesitan unos ciertos conocimientos sobre la estructura del lenguaje de marcado eXtensible Markup Language (XML). Android utiliza el lenguaje XML para definir toda la parte del diseño de la interfaz que contiene una aplicación. Toda la apariencia gráfica de la interfaz, imágenes, audios, videos, botones, controles, etc, es indicada en un fichero XML y cargada posteriormente en una clase (.java) donde se le dota de funcionalidad.

Con eso, se puede deducir que la estructura que sigue Android a la hora de disponer sus herramientas para desarrollar interfaces es la del Modelo Vista Controlador (MVC). El principal objetivo del MVC es separar completamente los datos de la aplicación de la apariencia visual de ésta. El MVC se divide en tres partes básicas:

Modelo: Son las representaciones sobre los distintos modelos de almacenamiento de la información que utiliza una aplicación Android, como bases de datos, web services, etc.

Vista: Es la apariencia básica que tiene una aplicación Android. Como se ha dicho anteriormente, esta apariencia es programada mediante el lenguaje XML.

Controlador: Son las clases (.java) que se encargan de relacionar la información con la apariencia gráfica. El controlador se encarga por tanto de integrar el modelo con la vista.

6.1.4. Componentes principales de una aplicación Android

Una vez descrito el funcionamiento del sistema Android, se describen, a modo de introducción, los componentes o clases principales que aparecen con mayor frecuencia en una aplicación Android.

Activity: La clase Activity (actividad) representa el componente principal de la interfaz gráfica de una aplicación Android. Las Activities pueden verse como las distintas pantallas o ventanas de las que se compone una aplicación.

Service: La clase Service (servicios) son componentes que no presentan interfaz gráfica y que funcionan en segundo plano. Los objetos Service son utilizados para crear tareas específicas como el lanzamiento de notificaciones, la actualización de datos de la interfaz principal, etc.

View: La clase View representa el componente básico en la que se apoyan todos los elementos que construyen una interfaz. En Android existen multitud de clases para generar interfaces. Todas las clases que generan una interfaces, como botones o etiquetas de texto, heredan directa o indirectamente de la clase View.

Content Provider: La clase Content Provider representa el mecanismo básico para compartir datos entre aplicaciones Android. Mediante los Content Provider, es posible utilizar datos determinados en diversas aplicaciones sin necesidad de mostrar detalles sobre la estructura o sobe su almacenamiento interno.

Broadcast Receiver: La clase Broadcast Receiver representa el componente destinado a detectar y reaccionar ante determinados eventos que genera el sistema operativo Android. Algunos de esos eventos pueden ser la conexión a una red Wi-Fi, la entrada de una llamada, la llegada de un SMS, etc.

Intent: La clase Intent representa el componente de comunicación entre objetos de clase Activity u objetos de clase Service. Esta clase permite la transferencia de datos entre Activities o Services. También permite iniciar nuevas Activities o incluso iniciar aplicaciones externas.

6.1.5. Estructura de un proyecto Android

Para entender la estructura de un proyecto en Android, lo primero que se debe distinguir son los conceptos de *proyecto* y *módulo*. Un proyecto es único, y engloba a todos los demás elementos. Dentro de un proyecto se pueden incluir varios *módulos*, que pueden representar aplicaciones distintas, versiones diferentes de una misma aplicación, o distintos componentes de un sistema. En nuestro caso se trabajará con un solo módulo dentro de nuestro proyecto.

A continuación se describen los contenidos principales de nuestro proyecto:

En el primer nivel se pueden observar dos carpetas: la que contiene todos los archivos y carpetas correspondientes al proyecto (Industriales en este caso), y la carpeta "External Libraries" que hace referencia a la librería de Android seleccionada para hacer la compilación (API 22 en nuestro caso), y a las librerías de Java (JDK).

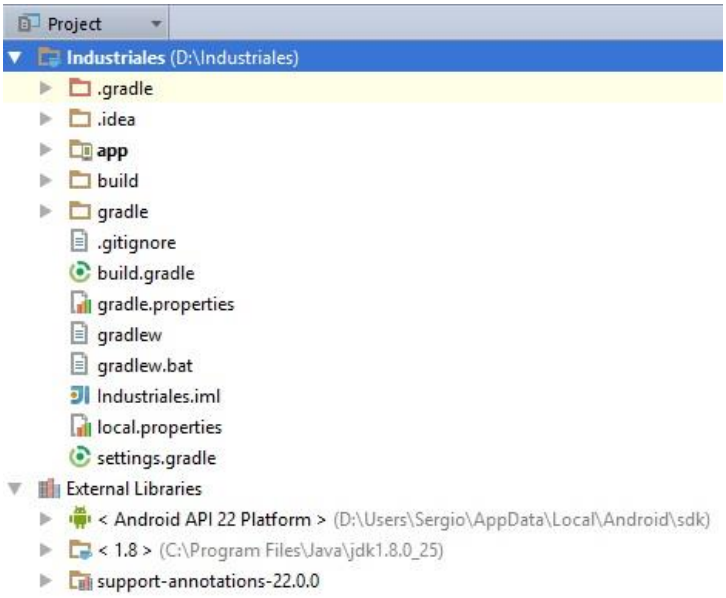


Ilustración 2: Estructura de un proyecto Android

La carpeta principal dentro del proyecto será la que posee el nombre que se le ha asignado al proyecto, y dentro de ella, la carpeta "**src**" es la que contiene todos los archivos del código fuente de la aplicación, así como sus recursos. El resto de archivos y carpetas corresponden a archivos de configuración del proyecto, archivos de la herramienta Gradle (que realiza la compilación automatizada del proyecto), y archivos correspondientes al sistema de control de versiones.

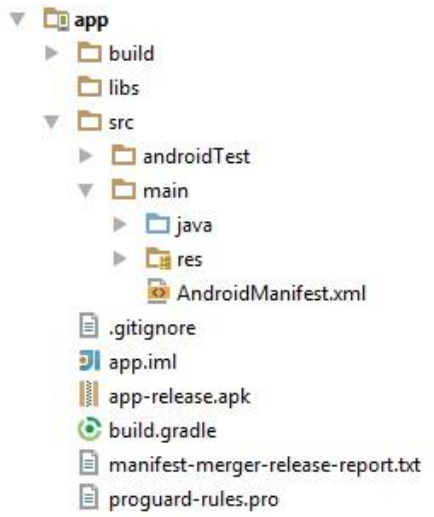


Ilustración 3: Estructura de la carpeta "app"

La carpeta **"src"** contiene 3 elementos principales:

java: Contiene los archivos de código fuente (.java) que debes ir creando para dirigir el funcionamiento de la aplicación. Se organiza en paquetes de igual manera que las aplicaciones Java, e inicialmente puedes encontrar en ella el archivo correspondiente al código fuente de la Activity que se ha creado al generar el proyecto.

res: Es la carpeta de recursos de la aplicación. En ella se mantendrán una serie de archivos en formato XML con los datos referentes a los recursos usados por la aplicación. Además almacenará las imágenes que usará la aplicación. Esta carpeta se organiza en subcarpetas en función del contenido que almacena cada una, y a su vez se crearán dentro de ellas otras subcarpetas según el idioma, tamaño de la pantalla, y otras características. De esa manera se podrán utilizar distintos valores y elementos en función de las características del dispositivo en el que se esté ejecutando la aplicación

AndroidManifest.xml: Es un archivo de control que contiene información sobre las características generales de la aplicación y sus componentes. Por ejemplo, describe algunas características sobre las Activities, services, Intent receivers, y content providers que va a utilizar la aplicación; los permisos que requiere la aplicación; las librerías externas que va a necesitar; las características requeridas para los dispositivos; los niveles de la API que se soportan o son requeridos; y otros.

En la carpeta **"res"** se pueden encontrar las siguientes subcarpetas:

drawable: Contiene las imágenes y otros elementos gráficos usados en la aplicación. Para poder definir diferentes recursos dependiendo de la resolución y densidad de la pantalla del dispositivo se suele dividir en varias subcarpetas:

/drawable (recursos independientes de la densidad)

/drawable-ldpi (densidad baja)

/drawable-mdpi (densidad media)

/drawable-hdpi (densidad alta)

/drawable-xhdpi (densidad muy alta)

/drawable-xxhdpi (densidad muy muy alta :)

layout: Contiene los ficheros de definición XML del aspecto de las diferentes pantallas de la interfaz. Para definir distintos Layouts dependiendo de la orientación del dispositivo se puede dividir también en subcarpetas:

/layout (vertical)

/layout-land (horizontal)

anim/animator: Contienen la definición de las animaciones utilizadas por la aplicación.

color: Contiene los ficheros XML de definición de colores.

menú: Contiene la definición XML de los menús de la aplicación.

xml: Contiene otros ficheros XML utilizados por la aplicación.

raw: Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos. Las imágenes utilizadas estarían dentro de esta carpeta.

values: Contiene otros ficheros XML de recursos de la aplicación, como por ejemplo cadenas de texto (strings.xml), estilos (styles.xml), colores (colors.xml), arrays de valores (arrays.xml), tamaños (dimens.xml), etc.

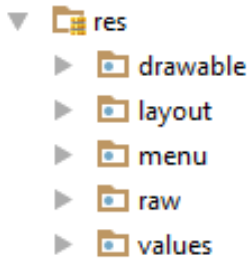


Ilustración 4: Estructura de la carpeta "res"

No todas estas carpetas tienen por qué aparecer en cada proyecto Android, tan sólo las que se necesiten. En este trabajo tan solo se utilizarán 5 de ellas como se puede observar en la imagen: drawable, layout, menú, raw y values.

Fichero "build.gradle":

Contiene información necesaria para la compilación del proyecto, por ejemplo la versión del SDK de Android utilizada para compilar, la mínima versión de Android que soportará la aplicación, referencias a las librerías externas utilizadas, etc.

En un proyecto pueden existir varios ficheros "build.gradle", para definir determinados parámetros a distintos niveles. En este trabajo se observa que solamente existe un fichero "build.gradle" a nivel de proyecto, y otro a nivel de módulo dentro de la carpeta /app. El primero define parámetros globales a todos los módulos del proyecto, y el segundo sólo tiene efecto para el módulo correspondiente.

Carpeta /app/build/:

Contiene una serie de elementos generados automáticamente al compilar el proyecto. Cada vez que se compila el proyecto, la maquinaria de compilación de Android genera una serie de ficheros automáticamente y que no se deberán modificar.

A destacar sobre todo el fichero que aparece desplegado en la imagen anterior, llamado "R.java", donde se define la clase R. Esta clase R contiene en todo momento una serie de constantes con los identificadores (ID) de todos los recursos de la aplicación, de forma que se puede acceder fácilmente a estos recursos desde el código a través de dichos datos.

6.1.6. Layouts

Entre los recursos creados por defecto cabe destacar los Layouts, que contienen la definición de la interfaz gráfica de la pantalla principal de la aplicación.

Los Layouts son ficheros XML que se encargan de establecer el diseño de la interfaz de usuario de una Activity. Controla la distribución, la dimensión y la posición de todos los elementos y controles que componen la interfaz, además de actuar como contenedores de objetos de la misma. Un Layout puede contener otros Layouts en su interior para así formar estructuras jerárquicas con los que organizar la vista de pantalla.

La siguiente lista describe los *Layouts* más utilizados en Android:

- **FrameLayout:** Este Layout es el más simple de todos. Posiciona todos los elementos que contiene en la esquina superior izquierda, de forma que solo será visible el último objeto que se ha introducido. Por este motivo, se suele utilizar para contener un solo elemento en su interior.
- **LinearLayout:** Se encarga de apilar todos los elementos que contiene uno detrás de otro de manera lineal. Un LinearLayout puede apilar sus elementos de forma vertical u horizontal, dependiendo del valor que se le aplique a su atributo "Android: orientation".
- **TableLayout:** Permite distribuir los elementos de su interior en forma de tabla. Dentro de un TableLayout se pueden definir filas y columnas y la posición de cada componente de interfaz dentro de la propia tabla. En realidad, TableLayout es una especialización de LinearLayout. En concreto, de un LinearLayout vertical. Aunque con un comportamiento particular.
- **TableRow:** Este componente es otra especialización de LinearLayout. Esta vez de un LinearLayout horizontal. Así que en realidad es un grupo de LinearLayout horizontales dentro de un LinearLayout vertical.
- **GridLayout:** Este elemento es una versión más potente de TableLoyout, pero tan solo está disponible desde la versión 14 de la API de Android, por lo que no funciona en versiones anteriores.
- **RelativeLayout:** Permite posicionar cada elemento de forma relativa a su elemento padre o a cualquier otro elemento contenido dentro del propio Layouts. Se puede indicar que un elemento esté por debajo de otro, por ejemplo.

- **AbsoluteLayout:** Posiciona los elementos que van en su interior especificando sus coordenadas “x” e “y”. Esta clase y su funcionalidad han sido marcadas como obsoletas, puesto que no se adecua a los diferentes tamaños de pantalla que puede tener un dispositivo Android.

Ilustración 5: Layouts principales

Estos Layouts son los más comunes, aunque también se suelen utilizar otra clase de Layouts, de complejidad mayor, que son descritas a continuación:

- **ScrollView:** Visualiza una columna de elementos y cuando estos no caben en la pantalla se permite un deslizamiento vertical.
- **ListView:** Visualiza una lista verticalmente en varios elementos. La diferencia con ScrollView es que los elementos mostrados no están predefinidos, sino que se crean dinámicamente según se necesiten a partir de un elemento base. Su utilización es algo compleja y se detalla su implementación en un apartado de la memoria.
- **GridView:** Visualiza una cuadricula deslizable de varias filas y varias columnas.
- **TabHost:** Proporciona una lista de ventanas seleccionables por medio de etiquetas que pueden ser pulsadas por el usuario para seleccionar la ventana que desea visualizar.
- **ViewFlipper:** Permite visualizar una lista de elementos de forma que se visualice uno cada vez. Puede ser utilizado para intercambiar los elementos cada cierto intervalo de tiempo.

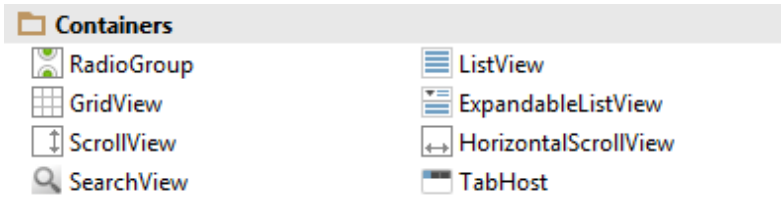


Ilustración 6: Layouts secundarios

6.1.6.1. Propiedades Layouts

El aspecto y el comportamiento de las vistas, tanto diseños como widgets, se pueden personalizar mediante propiedades. En el archivo XML las propiedades se representan mediante atributos de las etiquetas que se corresponden con las vistas. En la API, las propiedades son atributos privados o protegidos de las clases que implementan las vistas.

Para modificar una propiedad sólo tenemos que darle un valor al atributo correspondiente, incluyéndolo dentro de la etiqueta XML que representa la vista. Sin embargo, por código no se puede hacer lo mismo, dado que los atributos privados o protegidos no son accesibles y el lenguaje java no permite definir propiedades en las clases como hacen otros lenguajes (C#, por ejemplo). Por este motivo, cada atributo XML se suele corresponder con un método de la clase cuyo nombre coincide con el del atributo, pero con la cadena "set" como prefijo. Este prefijo es una convención habitual para indicar que el método sirve para establecer el valor de un atributo de la clase. Si la clase permite también recuperar el valor del atributo, existirá un método similar con prefijo "get".

La API de Android aprovecha la herencia de java para homogeneizar las propiedades disponibles para las vistas. Dado que las propiedades son atributos de las clases, pueden ser heredadas por las clases hijas (si se definen como protegidas y no como privadas). De esta forma, como todas las vistas heredan directa o indirectamente de la clase View, las propiedades que se definen en esta clase estarán disponibles en todas las clases hijas.

6.1.6.2. Identificadores

Una de las propiedades que proporciona la clase View es la que se utiliza en el ejemplo como "android: id". Se trata de un identificador para las vistas, aunque no es obligatorio. En el archivo XML, asignar un identificador a una vista sirve para poder hacer referencia a ella tanto en el propio archivo XML como en la actividad que lo utilice. Si no vamos a referenciar una vista no es necesario asignarle un identificador.

Para poder usar el identificador XML desde el código Java, el compilador genera una clase llamada R en la que se incluye una constante por cada atributo Id del archivo XML. El método findViewById de la clase Activity permite obtener la vista que se necesitan a partir de estas constantes de la clase R.

Por otro lado, cuando la interfaz de usuario se crea por entero desde el código se tienen objetos definidos para cada vista y se pueden usar para acceder a sus propiedades o como referencia en otras vistas, sin necesidad de asignar identificadores.

La clase View proporciona otras muchas propiedades a las vistas. Algunas de las más habituales son:

android: background (setBackgroundResource, setBackgroundDrawable, setBackgroundColor): permite reemplazar el fondo predeterminado de la vista por un color uniforme o una imagen.

android: clickable (setClickable): determina si la vista responderá a pulsaciones sobre ella o no. Se puede hacer que cualquier vista responda a pulsaciones aunque no sea su cometido habitual.

android: minWidth (setMinimumHeight) y android:minHeight (setMaximumHeight): permiten especificar un tamaño mínimo (ancho y alto) para la vista. Como los tamaños de pantalla de los dispositivos Android son muy variados y los diseños tienden a adaptarse a ellos, no se suelen fijar los tamaños de las vistas. Pero sí se puede establecer un mínimo que garantice la correcta presentación de su contenido.

android: padding (setPadding): el padding es un espacio alrededor de los límites de la vista, pero dentro de ella, que el contenido no puede ocupar. El padding sirve para alejar el contenido de los bordes de la vista, evitando que quede demasiado cerca o pegado a los contenidos de las vistas circundantes. Además del atributo android:padding, que permite definir el padding para los cuatro bordes de la vista simultáneamente, hay atributos particulares para cada borde: paddingLeft, paddingRight, paddingTop y paddingBottom.

android: visibility (setVisibility): permite controlar la visibilidad de la vista. Tiene tres posibles valores:

- visible: la vista se muestra en la actividad. Es el estado predeterminado.
- invisible: la vista no se muestra en la actividad, pero el diseño la tiene en cuenta. El resultado es que se deja sin ocupar el espacio que ocuparía si fuera visible.
- gone: la vista ni se muestra ni la tiene en cuenta el diseño. Es como si no se hubiera incluido nunca en la actividad. Sin embargo, la vista existe y sus propiedades son accesibles.

Luego, cada vista concreta añade sus propias propiedades. Por ejemplo, <TextView> y <Button> disponen de un atributo “android: text”, que permite establecer el texto que mostrarán.

6.1.6.3. Componentes básicos

Dentro de cada Layout se pueden insertar componentes, que definirán el aspecto de la Activity. Los componentes más utilizados son los siguientes:

- **TextView:** permite mostrar una o más líneas de texto en la pantalla.
- **EditText:** versión especializada del anterior que permite editar el texto.
- **Button:** widget que muestra una imagen con aspecto de botón presionable y con un texto en su interior. Hay un widget muy similar, ImageButton, que muestra una imagen en lugar de un texto.
- **CheckBox:** la típica casilla de verificación, con un texto y una imagen que cambia para indicar si está seleccionada o no.
- **RadioButton:** los botones de radio permiten escoger una entre múltiples opciones mutuamente excluyentes. Al igual que el CheckBox, el widget incluye un texto y una imagen que cambia de apariencia en función de si está seleccionado o no. Para agruparlos se utiliza RadioGroup, una versión especializada de LinearLayout.
- **ImageView:** permite mostrar una imagen. Hay otro widget, Gallery, que permite mostrar varias imágenes en forma de lista horizontal con desplazamiento.
- **ProgressBar:** permite mostrar una barra de progreso. Tiene una especialización que añade un elemento deslizable a lo largo de la barra que el usuario puede utilizar para establecer un valor: SeekBar.
- **Spinner:** es el equivalente Android de las listas de opciones desplegables, aunque aquí las opciones se muestran en una actividad que se superpone a la que contiene el widget.
- **DatePicker y TimePicker:** widgets que permiten seleccionar una fecha o una hora, respectivamente. Para seleccionar fechas se ha añadido recientemente a la API el widget CalendarView, que muestra el típico calendario y permite seleccionar una fecha sobre él.

6.1.6.4. Fragments

Se puede definir a un fragmento como una porción de interface de usuario o vista que se integra en una activity, es decir, trozos de la interfaz de usuario con su propio comportamiento y que pueden ser re-utilizados en las pantallas que deseemos.

Por lo tanto existe la posibilidad de combinar múltiples fragmentos en una sola actividad o incluso reutilizar fragmentos en otras actividades. De esta manera cada fragmento tendrá su propio ciclo de vida, recibirá sus propios eventos de entrada y se podrá agregar o eliminar mientras la activity de acogida este en marcha

Para poder crear un Fragment es necesario crear una subclase de Fragment. La clase Fragment, al igual que la clase Activity, contiene funciones de tipo callback, (cuando una función es pasada como argumento a otra función). Al menos, en todo ciclo de vida de un Fragment, se recomienda utilizar las siguientes funciones:

- **onCreate()** : El sistema llama a esta función cuando se crea el Fragment.
- **onCreateView()** : El sistema llama a esta función cuando se dibuja por primera vez el Fragment en la interfaz de usuario.
- **onPause()** : El sistema llama a esta función cuando el usuario deja de utilizar el Fragment (no implica que éste sea destruido).

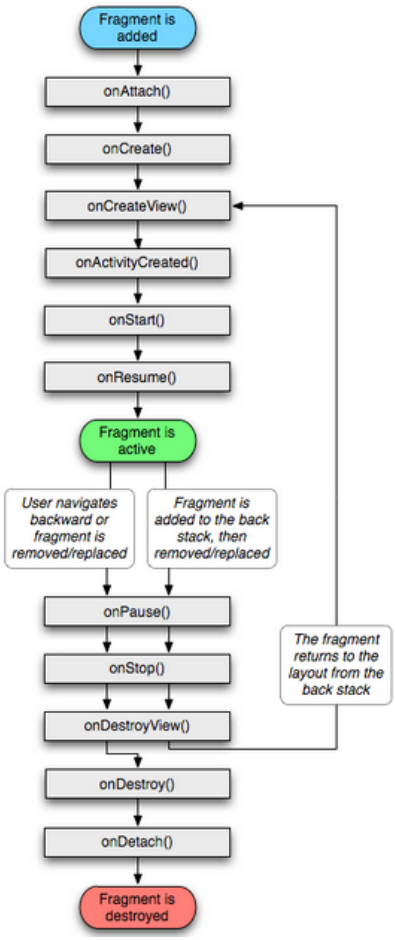


Ilustración 7: Ciclo de vida de un Fragment. Fuente: academiaandroid.com

6.1.7. Ciclo de vida de una Activity

Una vez implementados todos los Layouts es necesario implementar la lógica de la aplicación. Esto se hará mediante los Activities.

Una Activity es el medio por el cual un usuario interactúa con la aplicación. Es un tipo de clases java que heredan de Activity. Es decir, es una clase java que hereda los métodos de una más genérica. Siempre que se cree una nueva Activity, tiene que ser declarada en el fichero AndroidManifest.xml.

El sistema operativo Android trata a las Activities mediante una pila haciendo que cada Activity pase por multitud de estados mientras una aplicación se está ejecutando. El usuario final siempre estará visualizando la Activity que se encuentre en la cima de la pila. Cuando una Activity arranca, siempre se posiciona encima de la pila de Activities. El sistema va a mantener una pila con las actividades previamente visualizadas, de forma que el usuario va a poder regresar a la actividad anterior pulsando la tecla “atrás”.

Una aplicación Android corre dentro de su propio proceso Linux. Este proceso es creado para la aplicación y continuará vivo hasta que ya no sea requerido y el sistema reclame su memoria para asignársela a otra aplicación.

Una característica importante, y poco usual de Android, es que la destrucción de un proceso no es controlado directamente por la aplicación. En lugar de esto, es el sistema quien determina cuando destruir el proceso, basándose en el conocimiento que tiene el sistema de las partes de la aplicación que están corriendo (actividades y servicios), qué tan importante son para el usuario y cuanta memoria disponible hay en un determinado momento.

Si tras eliminar el proceso de una aplicación, el usuario vuelve a ella, se crea de nuevo el proceso, pero se habrá perdido el estado que tenía esta aplicación.

Como vemos, Android es sensible al ciclo de vida de una Activity, que puede estar en uno de estos cuatro estados:

- **Activa (Running):** La Activity está encima de la pila, lo que quiere decir que es visible y tiene el foco.
- **Visible (Paused):** La Activity es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra actividad con alguna parte transparente o que no ocupa toda la pantalla. Cuando una Activity está tapada por completo, pasa a estar parada.
- **Parada (Stopped):** Cuando la Activity no es visible, se recomienda guardar el estado de la interfaz de usuario, preferencias, etc.
- **Destruida (Destroyed):** Cuando la Activity termina, o es matada por el sistema Android, sale de la pila de actividades.

Cada vez que una actividad cambia de estado se van a producir eventos que podrán ser capturados por ciertos métodos de la actividad. A continuación se muestra un esquema que ilustra los métodos que capturan esos eventos:

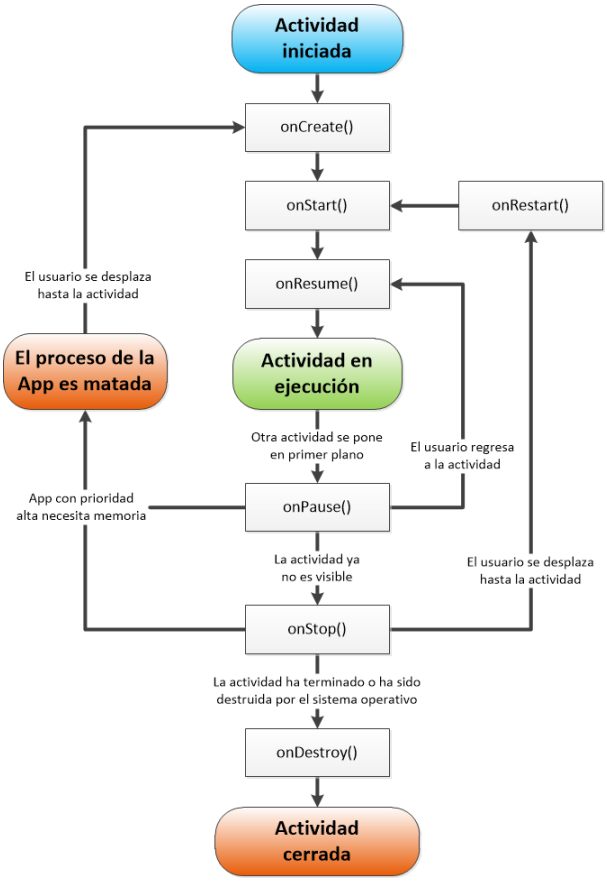


Ilustración 8: Ciclo de vida de una Activity. Fuente: jarroba.com

Es necesario explicar cada uno de los métodos:

- **onCreate():** Se dispara cuando la Activity es llamada por primera vez. Aquí es donde se debe crear la inicialización normal de la aplicación, crear vistas, hacer los bind de los datos, etc. Este método te da acceso al estado de la aplicación cuando se cerró. Después de esta llamada siempre se llama al onStart().
- **onRestart():** Se ejecuta cuando tu Activity ha sido parada, y quieres volver a utilizarla. Si ves el diagrama podrás ver que después de un onStop() se ejecuta el onRestart() e inmediatamente llama a un onStart().
- **onStart():** Se ejecuta cuando la Activity se está mostrando apenas en la pantalla del dispositivo del usuario.
- **onResume()**: Se ejecuta una vez que la Activity ha terminado de cargarse en el dispositivo y el usuario empieza a interactuar con la aplicación. Cuando el usuario ha terminado de utilizarla es cuando se llama al método onPause()**.**
- **onPause():** Se ejecuta cuando el sistema arranca una nueva Activity que necesitará los recursos del sistema centrados en ella. Hay que procurar que la llamada a este método sea rápida ya que hasta que no se termine su ejecución no se podrá arrancar la nueva Activity. Después de esta llamada puede venir un onResume() si la Activity que haya ejecutado el onPause() vuelve a aparecer en primer plano o un onStop() si se hace invisible para el usuario.

- **onStop():** Se ejecuta en el momento en el que la Activity ya no es visible para el usuario porque otra Activity ha pasado a primer plano. Según el diagrama, después de que se ha ejecutado este método existen tres opciones: ejecutar el onRestart() para que la Activity vuelva a aparecer en primer plano, que el sistema elimine este proceso porque otros procesos requieran memoria o ejecutar el onDestroy() para apagar la aplicación.
- **onDestroy():** Esta es la llamada final de la Activity, después de ésta, es totalmente destruida. Esto pasa por los requerimientos de memoria que tenga el sistema o porque de manera explícita el usuario manda a llamar este método. Si se quisiera volver a ejecutar la Activity se arrancaría un nuevo ciclo de vida.

En esta aplicación tan solo se usará el método onCreate() para definir la vista de la actividad con los layouts que se han diseñado. También se definirán aquí las variables necesarias para el funcionamiento de la aplicación.

Un ejemplo de una implementación básica de este método seria el siguiente:

```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_login);
   TextView advertencia= (TextView) findViewById(R.id.textViewLogin);
   TextView usuario = (EditText) findViewById(R.id.editTextUsuario);
   TextView contraseña = (EditText) findViewById(R.id.editTextContraseña);
}
```

Este ejemplo corresponde a la primera Activity de la aplicación, la actividad login que se ejecuta nada más abrir la aplicación. En este método se define el layout “activity_login” como el contenido de la vista que se mostrará en pantalla.

También se definen 3 elementos TextView que servirán para almacenar las variables de usuario y contraseña para entrar en la aplicación.

Se han implementado pues, los métodos onCreate() en todas las Activity definiendo las vistas correspondientes a cada una.

6.1.8. Intents

El componente Intent sirve principalmente, para realizar invocaciones a otros componentes Android (Activities, Services, Broadcast Receivers, escétera). El componente Intent, como mecanismo para invocar componentes es muy sencillo de utilizar. Además, permite lanzar aplicaciones que son externas a la que se está desarrollando, lanzar eventos a los que otras aplicaciones puedan responder, lanzar alarmas periódicas, etcétera.

El propósito más utilizado del componente Intent es el que abarca la inicialización de Activities propias del desarrollador. Con esta funcionalidad se da soporte al desarrollador para crear el sistema de navegación de una aplicación. La manera más sencilla y más común de realizar una transición a una nueva Activity mediante el componente Intent es la siguiente:

```
Intent i = new Intent (clase_origen.this, clase_destino.class);
startActivity(i);
```

Hay ocasiones en las que es interesante enviar datos de una Activity a otra. Para ello se utiliza el método putExtra() que ofrece la clase Intent. Dicho método permite enviar datos de las siguientes clases y tipos primitivos: int, long, boolean, float, double, short, char, String, CharSecuence, byte y Parcelable. Cabe destacar que dichos datos siempre van etiquetados mediante un String que hace el papel de clave o etiqueta:

```
Intent i = new Intent (clase_origen.this, clase_destino.class);
i.putExtra("etiqueta","valor");
startActivity(i);
```

Una vez que se envían los datos desde la activity origen, es necesario recogerlos en la Activity destino, esto se consigue utilizando el método getExtras() de la clase Intent que devuelve los datos en una instancia de la clase Bundle, en el caso que se hayan enviado correctamente:

```
Bundle extras = getIntent().getExtras();
```

Por último, para obtener los datos del objeto Bundle, es necesario llamar al método get que se corresponda con el tipo de dato enviado. Dicho método siempre tiene que contener la cadena de caracteres con la que se han etiquetado los datos al enviarse desde la Activity destino:

```
String datos = extras.getString("etiqueta");
```

6.1.9. Entradas de Android

Los teléfonos Android suelen incorporar una pantalla táctil, que es utilizada como dispositivo principal de entrada. El uso más importante de la pantalla táctil es como sustituto del ratón de un ordenador de sobremesa. De esta forma se puede seleccionar, arrastrar y soltar cualquier elemento utilizando tan solo la pantalla.

Android captura los distintos eventos de usuario de forma homogénea y se los pasa a la clase encargada de recogerlos. Por lo general va a ser un objeto tipo "View" el que recogerá estos eventos por medio de dos técnicas alternativas. Los escuchadores de eventos (Event Listener) y los manejadores de eventos (Event Handler). En este trabajo tan solo se tratarán los escuchadores de eventos, ya que serán los únicos que se utilizarán.

Un Escuchador de eventos o Event Listener es una interfaz de la clase View que contiene un método callback que ha de ser registrado. Cada Escuchador de Eventos tiene solo un método callback, que será llamado por Android cuando se produzca la acción correspondiente. Se tienen los siguientes escuchadores de eventos:

- **onClick():** Método de la interfaz View.onClickListener. Se llama cuando el usuario selecciona un elemento. Se puede utilizar cualquier medio como la pantalla táctil, las teclas de navegación o el track ball.
- **onLongClick():** Método de la interfaz View.OnLongClickListener. Se llama cuando el usuario selecciona un elemento durante más de un segundo.
- **onFocusChange():** Método de la interfaz View.OnFocusChangeListener. Se llama cuando el usuario navega dentro o fuera de un elemento.
- **onKey():** Método de la interfaz View.OnKeyListener. Se llama cuando se pulsa o se suelta una tecla del dispositivo.
- **onTouch():** Método de la interfaz View.OnTouchListener. Se llama cuando se pulsa o se suelta o se desplaza en la pantalla táctil.
- **onCreatecontextMenu():** Método de la interzaz View.OnCreateContextMenuListener. Se llama cuando se crea un menú de contexto.

Un ejemplo de implementación de código consistiría en añadir un la interfaz OnClickListener() a un objeto botón, por ejemplo:

```
Button boton = (Button) findViewById (R.id.boton);
Botón.setOnClickListener( new OnClickListener() {
        Public void onClick (View v) {
                //Acciones a realizar
        }
});
```

En este ejemplo se busca primero el elemento deseado y se guarda en la variable “boton”. A continuación se le asigna un nuevo OnClickListener() donde su método principal OnClick() contendrá las acciones a realizar cuando se pulse el botón.

6.1.10. Dimensiones

Algunas ocasiones, se ha de indicar el tamaño que ha de tener un objeto dado, como los textos por ejemplo. Dado que la aplicación podrá ejecutarse en gran variedad de dispositivos con resoluciones muy diversas, así que es de interés que la aplicación sea compatible con todas las pantallas posibles. Android permite indicar cualquier tamaño o coordenada con diversas unidades de medida:

– **dp (Density-independent Pixels)**

Es una unidad abstracta que se basa en la densidad física de la pantalla. Esta unidad es equivalente a un píxel en una pantalla con una densidad de 160 dpi. Cuando se está ejecutando en una pantalla de mayor densidad, se aumentan el número de píxeles utilizados para dibujar 1dp según los dpi’s de la pantalla. Por otro lado, si la pantalla es de menor densidad, el número de píxeles utilizados para 1dp se reducirán.

– **sp (Scale-independent Pixels)**

Esta unidad es como la anterior, pero se escala según el tamaño de fuente configurada. Se ajusta tanto para la densidad de pantalla y como a las preferencia del usuario.

– **pt (Points)**

Es un 1/72 de una pulgada, según el tamaño físico de la pantalla.

– **px (Pixels)**

Corresponde a un píxel real en la pantalla.

– **mm (Milímetros)**

Son milímetros reales según el tamaño físico de la pantalla.

– **in (Pulgadas)**

Son pulgadas reales según el tamaño físico de la pantalla.

Utilizar las unidades “dp” en lugar de píxeles es la solución más simple para tratar los diferentes tamaños de pantalla de los dispositivos, aunque se recomienda utilizar las unidades “sp” si se especifican tamaños de fuente.

Las unidades de medida “pt”, “px”, “mm” y “in” no se recomienda porque la representación real puede variar según el dispositivo en el que se ejecute, ya que cada uno de ellos puede tener un número diferente de píxeles por pulgada y pueden tener más o menos píxeles totales disponibles en la pantalla.

6.2. Programas utilizados

Para la realización del trabajo será necesaria la utilización de varios programas informáticos, todos ellos ejecutados bajo el sistema operativo Windows 8. Estos programas son los siguientes:

6.2.1. Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para la plataforma Android. Ha sido desarrollado por Google y puede descargarse gratuitamente a través del siguiente enlace: https://developer.android.com/sdk/index.html.

Android Studio será el programa principal con el que se realizará el trabajo.

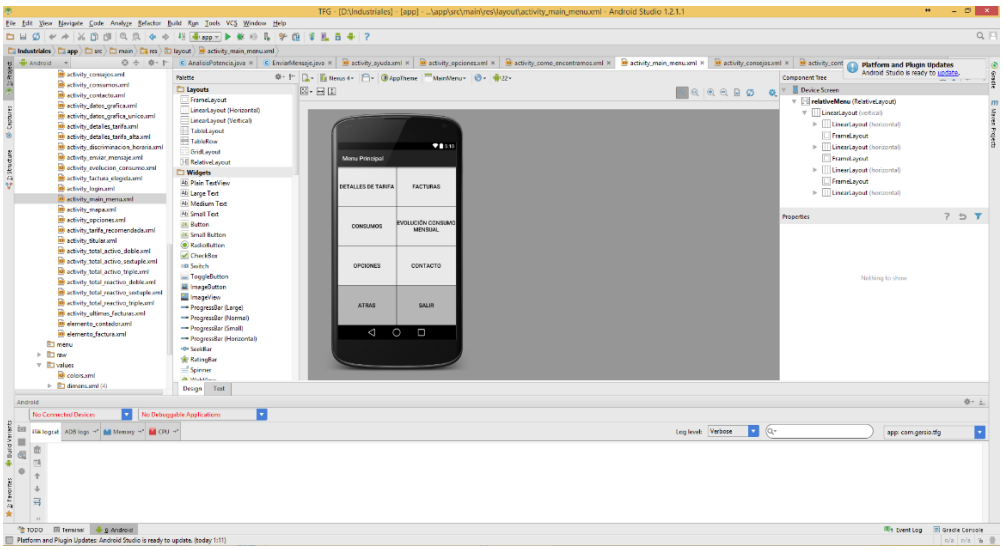


Ilustración 9: Android Studio

6.2.2. SDK Manager

SDK son las siglas de “Software Development Kit”, que significa “Kit de Desarrollo de Software”. Mediante éste kit se pueden desarrollar aplicaciones y ejecutar un emulador de la versión de Android.

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

Cada vez que aparece una nueva versión de Android, Google libera el código fuente y publica el SDK con la nueva versión de Android. Esto sirve para que los desarrolladores puedan adaptar sus aplicaciones a la nueva versión. Estos SDKs son gestionados por el programa informático SDK Manager y su descarga viene junto con la de Android Studio.

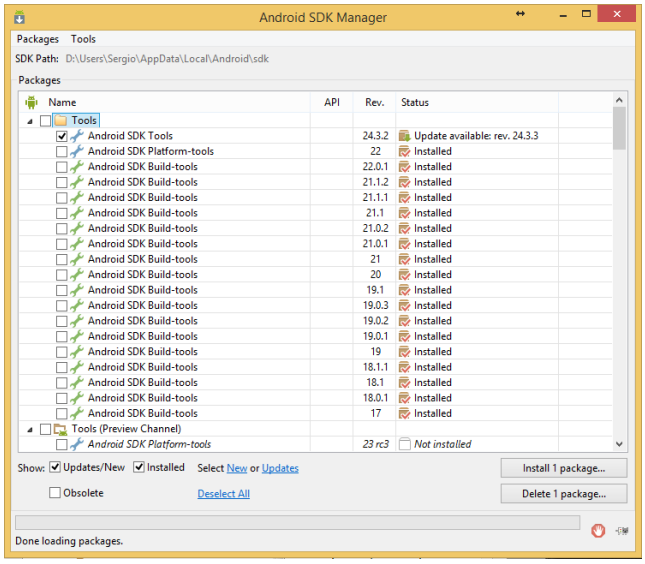


Ilustración 10: SDK Manager

6.2.3. phpMyAdmin

phpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 62 idiomas.

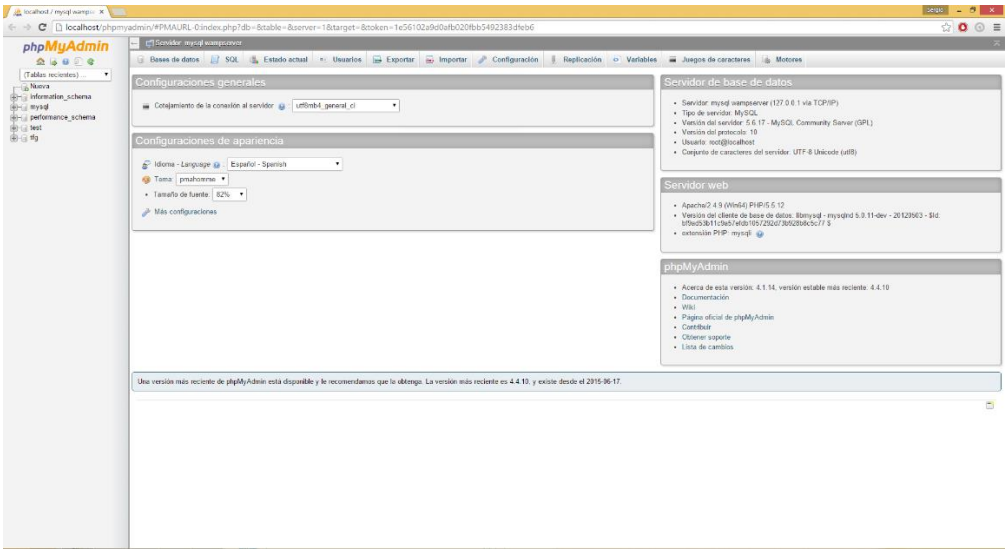


Ilustración 11: phpMyAdmin

6.3. Diseño:

6.3.1. Listado de funciones

6.3.1.1. Funciones primarias:

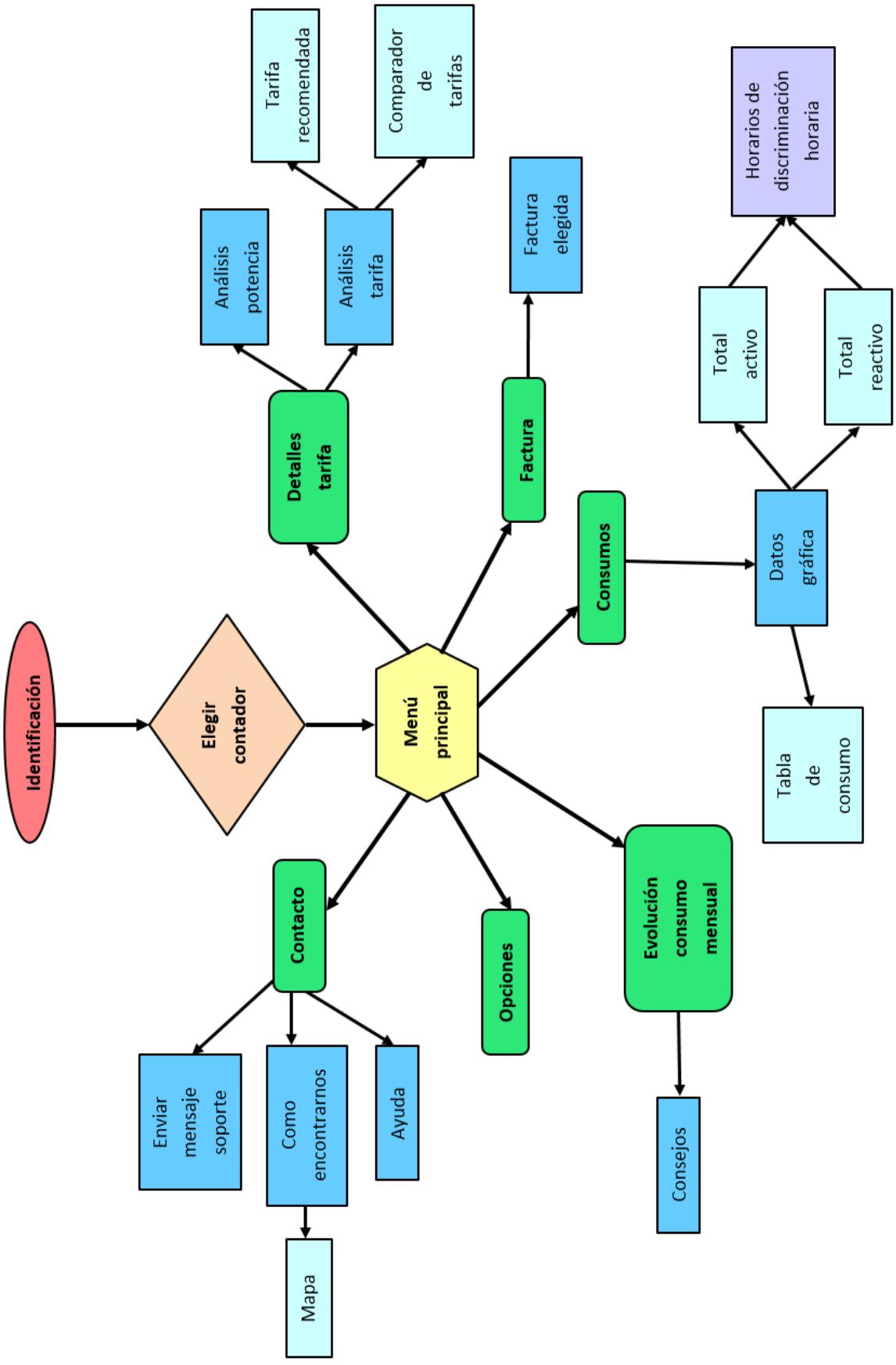
- Mostrar una lista de todos los contadores que posea el usuario. A partir de este punto, todas las funciones primarias estarán referidas al contador seleccionado.
- Mostrar información sobre la potencia y la tarifa contratada.
- Si la potencia contratada es menor a 15KW, efectuar una simulación del importe de las facturas con las tarifas disponibles para determinar si la tarifa contratada es la óptima para el usuario.
- Si la potencia contratada es mayor a 15KW, mostrar la potencia máxima consumida para analizar si el usuario necesita aumentar o disminuir la potencia contratada.
- Mostrar las últimas facturas, así como incluir la posibilidad de seleccionar cada una de ellas y ver en detalle los diferentes términos en los que se descompone la factura.
- Representar en forma de gráfica y en forma de tabla de valores la energía consumida por un usuario durante un determinado periodo de tiempo dado. Así como mostrar los consumos totales en cada periodo de discriminación horaria.
- Comparación de la energía consumida media del último mes con la misma media referida al año anterior.

6.3.1.2. Funciones secundarias:

- Mostrar una lista de consejos con los que ahorrar en la factura eléctrica.
- Soporte para varios idiomas: Castellano, Valenciano e Inglés
- Soporte para diferentes tamaños de pantalla
- Mostrar mapa de google con la ubicación de las oficinas centrales.
- Capacidad de enviar un mensaje al soporte de la aplicación.

Todas aquellas funciones que lo necesiten, efectuarán una conexión web con una base de datos alojado en un servidor externo.

6.3.2. Diagrama de pantallas



6.3.3. Diseño de pantallas:

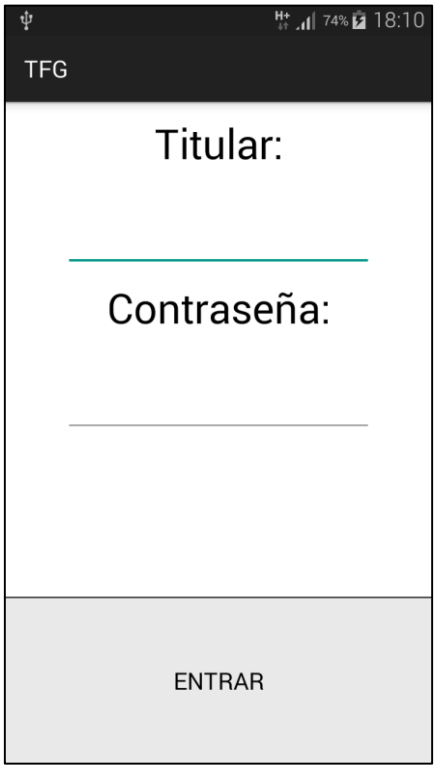


Ilustración 12: Identificación

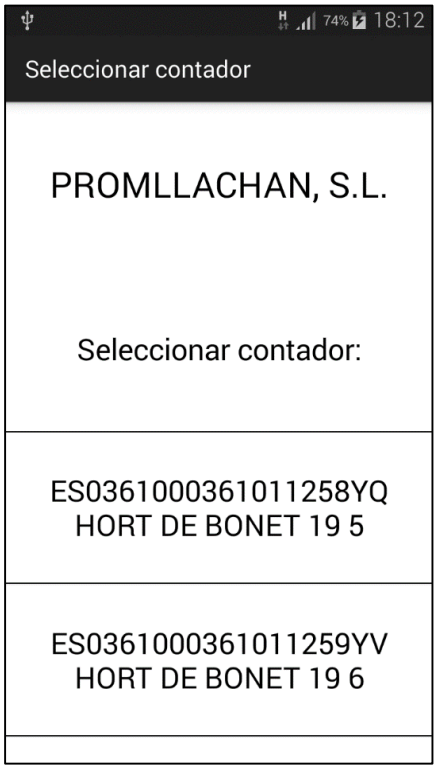


Ilustración 13: Seleccionar contador

Ilustración 14: Menú Principal

Ilustración 15: Detalles de tarifa (baja potencia)

Ilustración 16: Detalles de tarifa (alta potencia)

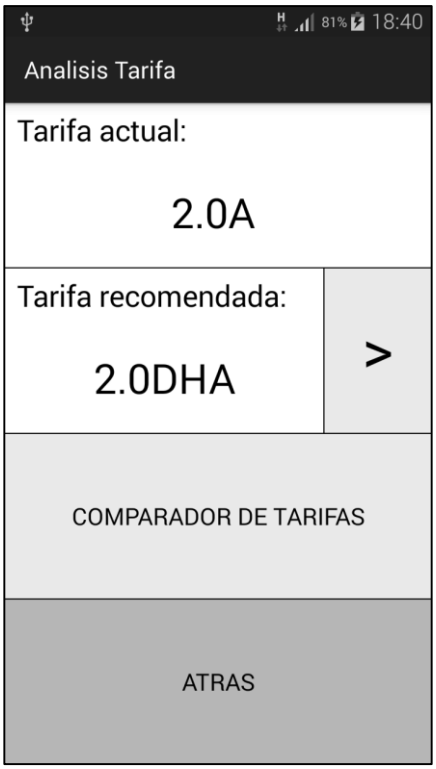


Ilustración 17: Análisis de tarifa

Ilustración 18: Tarifa recomendada

Ilustración 19: Comparador de tarifas

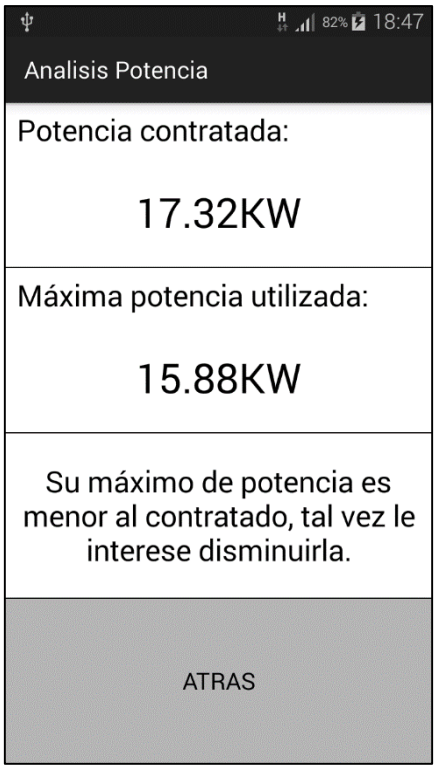


Ilustración 20: Análisis de potencia

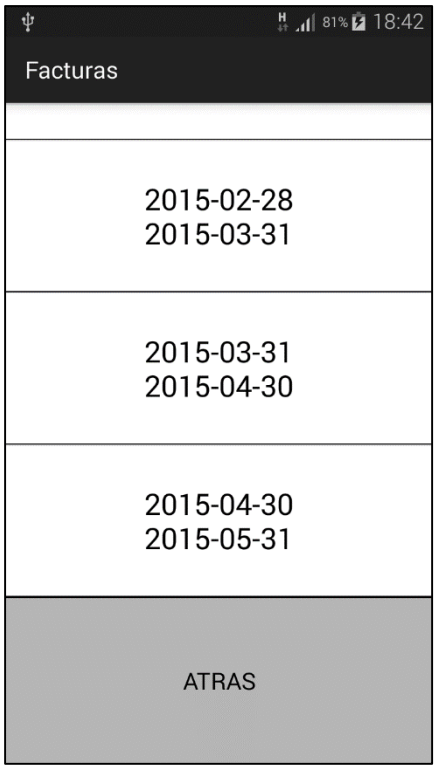


Ilustración 21: Facturas

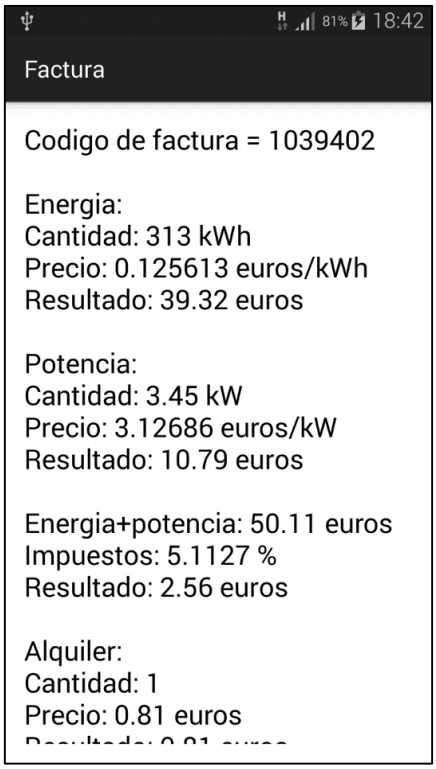


Ilustración 22: Factura elegida

Ilustración 23: Consumos

Ilustración 24: Datos gráfica (Sin DH)

Ilustración 25: Datos gráfica (con DH)

Ilustración 26: Total activa doble

Ilustración 27: Total activa triple

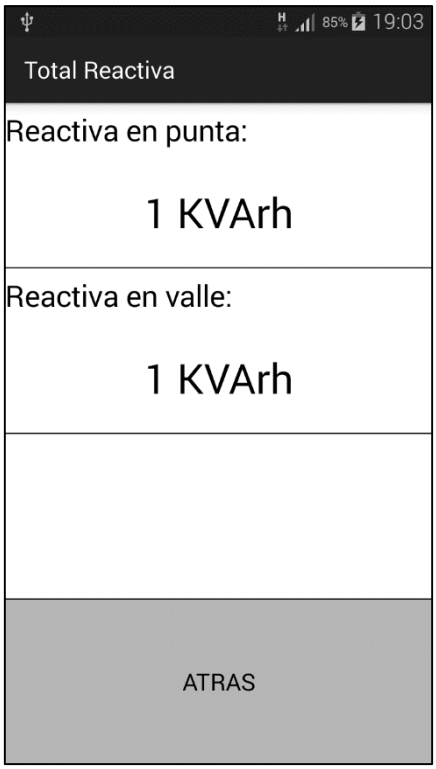


Ilustración 28: Total reactiva doble

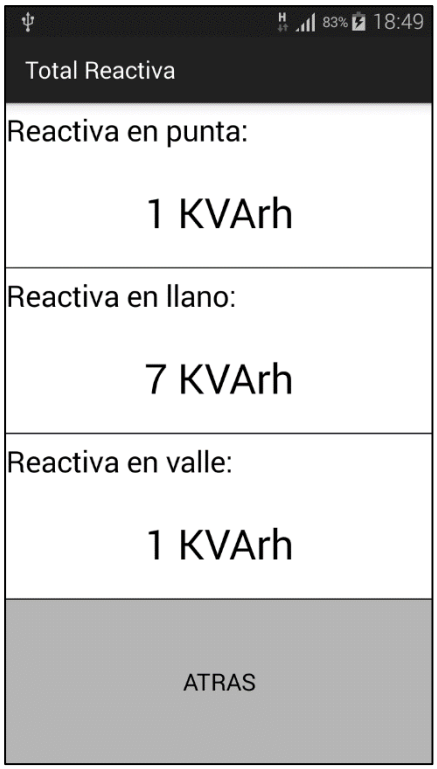


Ilustración 29: Total reactiva triple

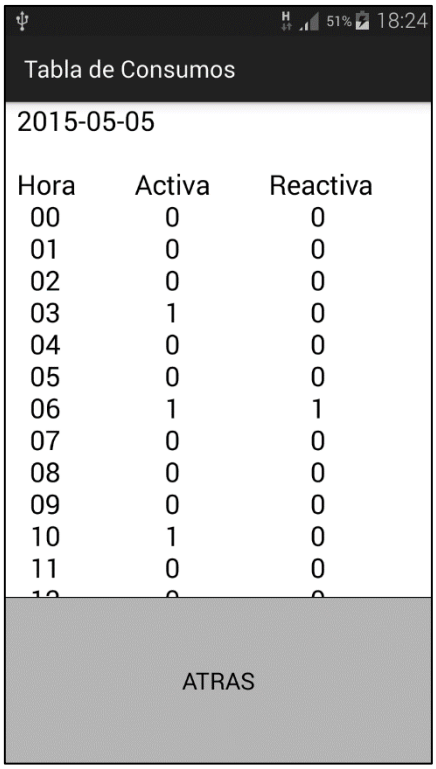


Ilustración 30: Tabla de consumos

Ilustración 31: Evolución consumo mensual

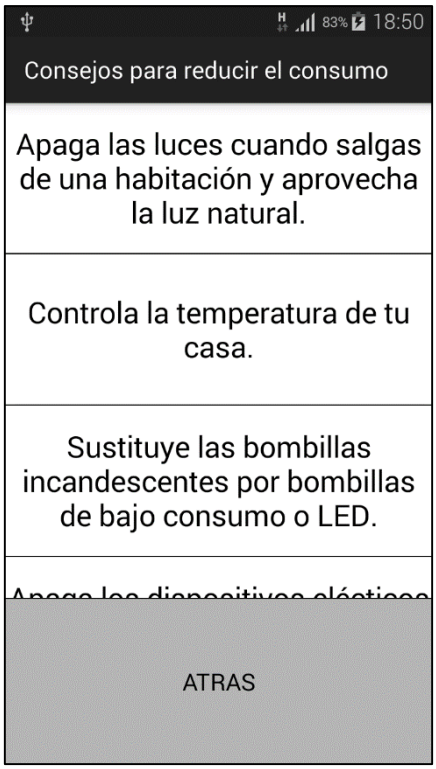


Ilustración 32: Consejos

Ilustración 33: Opciones

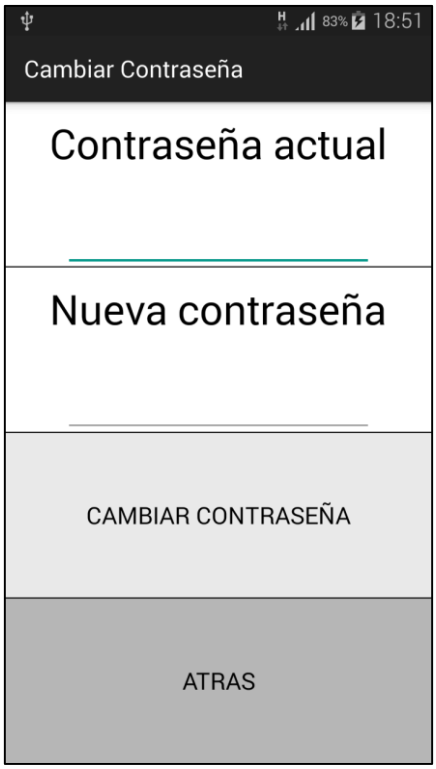


Ilustración 34: Cambiar contraseña

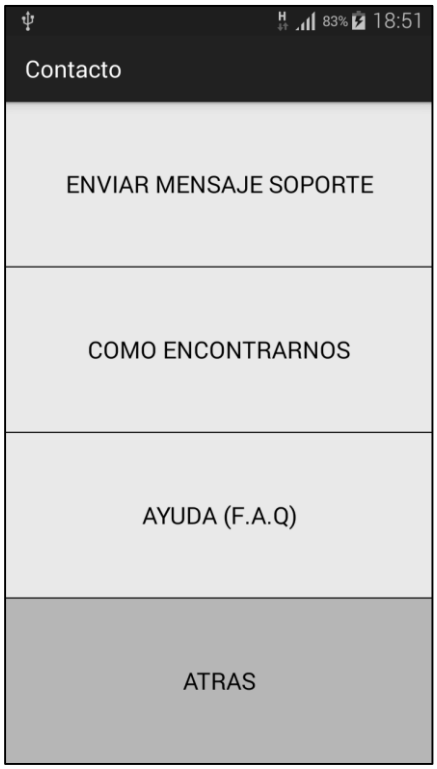


Ilustración 35: Contacto

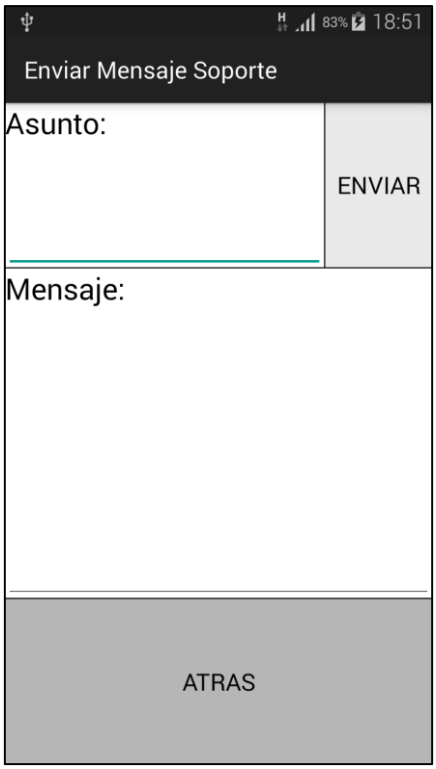


Ilustración 36: Enviar mensaje soporte

Ilustración 37: Como encontrarnos

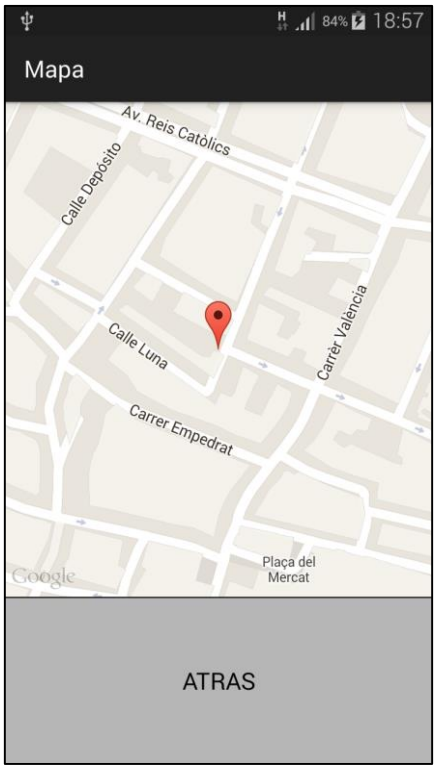


Ilustración 38: Mapa

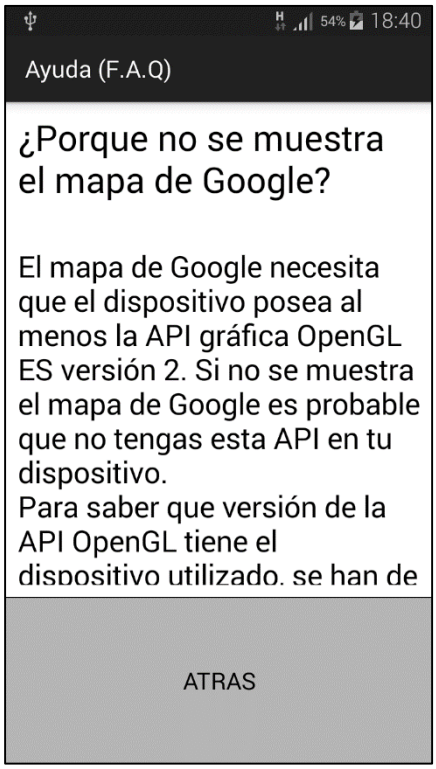


Ilustración 39: Ayuda

6.4. Elección de API mínima.

El primer paso para desarrollar una aplicación en Android es elegir cual será la mínima versión de Android que soportará la aplicación. Hay que tener en cuenta que hay muchas funcionalidades que solo están disponibles a partir de una API mínima de Android, pero también hay que pensar que si la aplicación a desarrollar debe llegar al mayor número de dispositivos posible, es necesario tener en cuenta que versiones son las más utilizadas actualmente por los usuarios. Es necesario saber que la evolución siempre es ascendente, es decir, cada vez quedarán menos dispositivos con las versiones más antiguas. A fecha de 01 de junio de 2015, la distribución de versiones de Android es la siguiente:

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.1%
4.1.x	Jelly Bean	16	14.7%
4.2.x		17	17.5%
4.3		18	5.2%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	11.6%
5.1		22	0.8%

Ilustración 40: Distribución versiones Android. Fuente: developer.android.com

La versión mínima con la que se puede acceder a la Google Play es la 8, por lo que se empieza a realizar el análisis desde esa versión. Las versiones más antiguas a la 8 son consideradas obsoletas. Tampoco aparecen las versiones Android con menos de un 0,1% de distribución.

La funcionalidad de la aplicación a desarrollar con una exigencia de API más alta es la del mapa de google. El mapa de google necesita de una API mínima de 11. Se puede definir perfectamente esta API como la mínima necesaria para ejecutar la aplicación, pues abarcaría el 94,1% del total de los dispositivos. Se podría definir también como API mínima la versión 15, puesto que las versiones intermedias no tienen prácticamente distribución y no influyen en el porcentaje.

Para definir la API mínima en el proyecto de la aplicación, puede elegirse tanto al crear la aplicación, como cambiando el código necesario una vez ya creado el proyecto. Para ello, habrá que dirigirse al archivo build.gradle y modificar la línea de código “minSdkVersion” y sustituir el siguiente número por el API mínima que se desee, en este caso “minSdkVersion 11”.

6.5. Layouts utilizados

Para el correcto diseño de las diferentes pantallas en Android Studio, han sido necesarias la utilización de varios tipos de layouts. Como regla general, se ha usado un Linear Layout como elemento raíz con cuatro Frame Layout. Estos Frame Layout tendrán fijas las propiedades de altura (height) y peso (weight) como “0dp” y “1” respectivamente, mientras que la propiedad anchura (widgth) será “match_parent”:

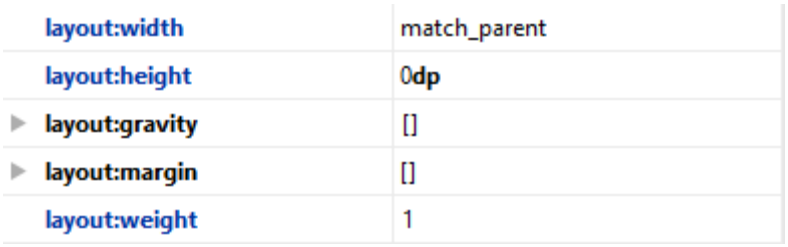


Ilustración 41: Propiedades Layout

Esto permite que los cuatro elementos ocupen toda la pantalla, todos con la misma altura sin importar el tamaño ni la densidad de la pantalla, para poder implementar efectivamente el diseño de la aplicación.

En el último FrameLayout, se situará un botón, con el texto “ATRÁS”, que servirá para navegar a la pantalla anterior. Este botón tendrá las propiedades de altura y anchura como “match_parent”, por lo que ocupará todo el FrameLayout en el que está contenido.

Las líneas separadoras se han implementado con otros tres FrameLayouts situados entre los cuatro primeros que contienen los elementos principales. Estos tres FrameLayouts tendrán una anchura de “1dp” y un color negro de fondo (propiedad background).

Para la implementación de las listas de consejos se utilizará un elemento ScrollView dentro de un FrameLayout con el peso del mismo ajustado, ya que el elemento ScrollView no admite la propiedad peso.

Para implementar la lista de contadores o la lista de facturas, se utilizará el elemento ListView, también dentro de un FrameLayout.

Para la utilización del mapa de Google y de la gráfica GraphView, se utilizarán Fragments dentro de FrameLayouts y se les asignará su función en la clase java de la Activity correspondiente.

6.6. WampServer

WAMP es el acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

- Windows, como sistema operativo;
- Apache, como servidor web;
- MySQL, como gestor de bases de datos;
- PHP (generalmente), Perl, o Python, como lenguajes de programación.

El uso de un WAMP permite servir páginas "html" a internet, además de poder gestionar datos en ellas. Al mismo tiempo, un WAMP proporciona lenguajes de programación para desarrollar aplicaciones web.

En este trabajo se ha utilizado la plataforma WampServer para montar la infraestructura web necesaria para que la aplicación funcionara correctamente. WampServer es un entorno de desarrollo web en el sistema operativo Windows. Permite crear aplicaciones webs con Apache2, PHP y bases de datos MySQL. Además, incluye el programa informático PhpMyAdmin que permite manejar fácilmente las bases de datos.

6.6.1. Instalación:

El primer paso a seguir es la descarga, instalación y configuración del Wampserver. Se descarga el programa gratuito desde la página web http://www.wampserver.com/en/ y se procede a su instalación:

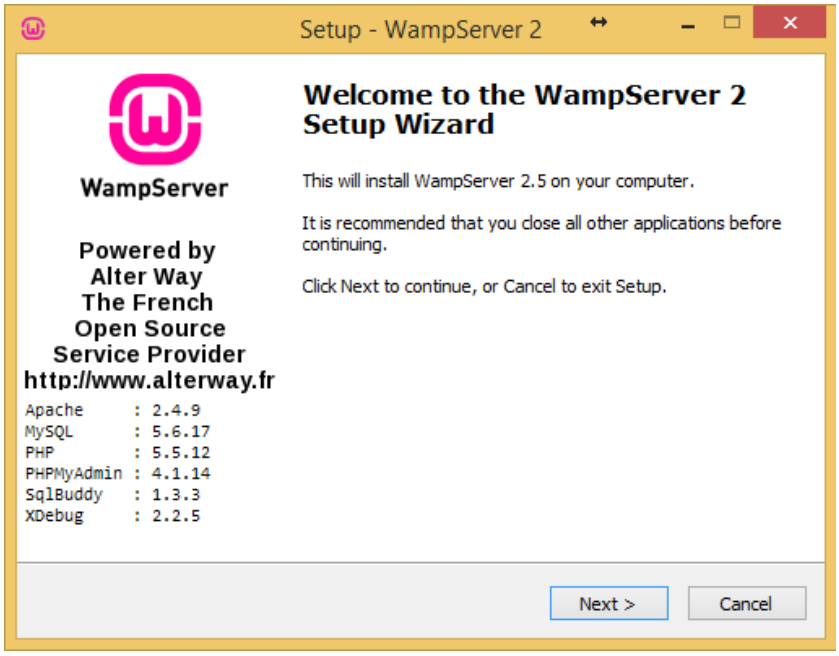


Ilustración 42: Instalación WampServer

Para instalar Wampserver, hay que hacer doble clic sobre el instalador y seguir las instrucciones. Se aceptan los términos y condiciones, se elige el directorio de instalación y se extraen los archivos. Al extraer los archivos se preguntará cual es el navegador que Wampserver utilizará. Por defecto esta seleccionado el navegador Internet Explorer aunque en este trabajo se ha cambiado para funcionar en Google Chrome. Finalmente, se preguntará por los parámetros del correo PHP. Al acabar este último paso, Wampserver estará finalmente instalado.

6.6.2. Ejecutar WampServer:

Para ejecutar WampServer, tan solo hay que hacer doble clic en su icono y aparecerá un icono en la barra de herramientas. Haciendo clic en él se obtiene el siguiente menú:

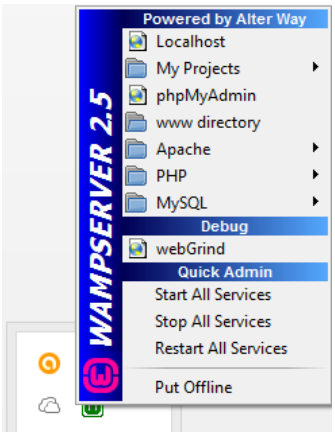


Ilustración 43: Menú WampServer

Los servicios y archivos más importantes son:

- **Aplicaciones**: *Localhost* (lleva a la página principal) y *Phpmyadmin* para manejar las Bases de Datos.
- **Directorio www**: Es el directorio donde se situan los proyectos y páginas webs.
- **Log Files**: Donde se registran las acciones e incidencias de cada aplicación (*apache, php, mysql*)
- **Config Files**: Donde se pueden acceder a los ficheros de configuración del Apache (*httpd.conf*), php (*php.ini*) y MySQL (*my.ini*) y cambiar sus parámetros
- **Control sobre las aplicaciones**: También se podrá detener o reanudar cada aplicación. Si está todo funcionando, el icono deberá estar en verde. Si no está todo, aparecerá en amarillo, y si no hay nada, en rojo.

6.6.3. phpMyadmin

Si se ejecuta phpMyAdmin haciendo clic en la pestaña del menú del Wampserver, este llevará a la pantalla principal de gestión de las bases de datos:

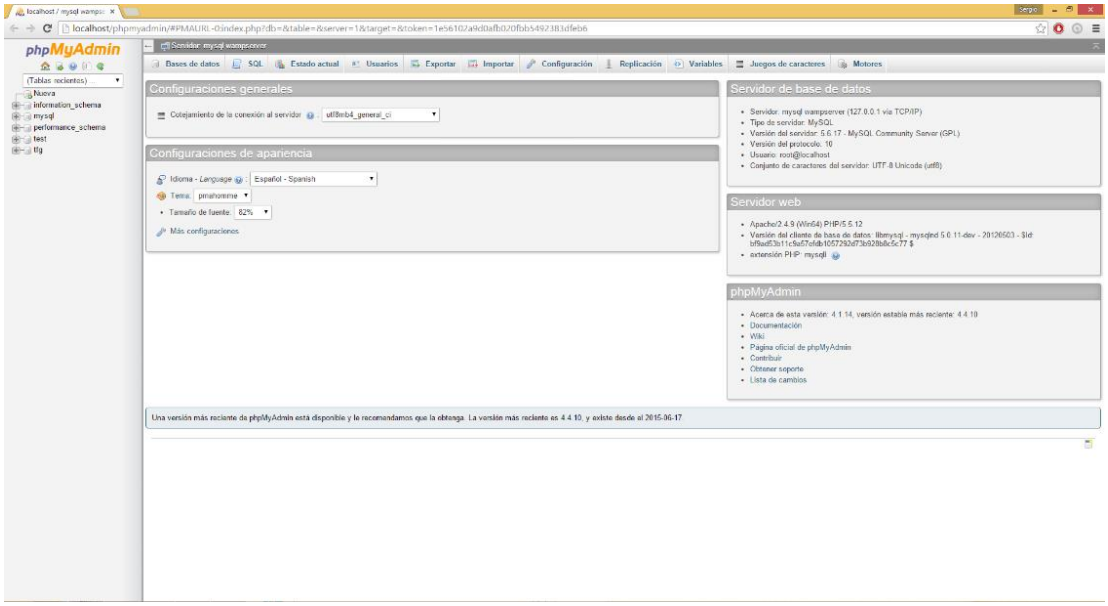


Ilustración 44: Página principal phpMyAdmin

Las distintas pestañas a las que se puede acceder están situadas en la parte superior de la pantalla. Las más utilizadas son las siguientes:

- **Bases de datos:** En esta pestaña se puede observar una lista con las distintas bases de datos situadas en el servidor. Se pueden también crear nuevas bases de datos y comprobar los privilegios de las que ya están creadas:

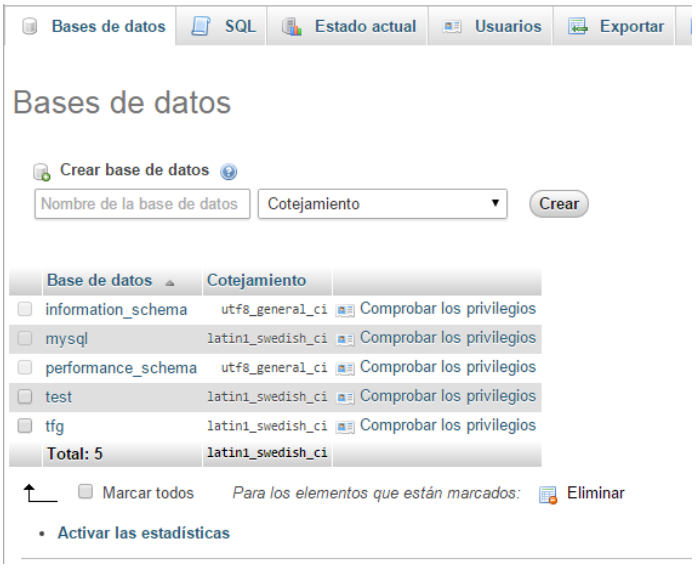


Ilustración 45: Pestaña "Bases de datos"

- **SQL:** Aquí se muestra un cuadro de texto editable sobre el que se pueden escribir las consultas deseadas de forma sencilla.
- **Estado actual:** En esta pestaña se nos proporciona información sobre las estadísticas de uso del servidor, como por ejemplo el tráfico de red desde que ha estado activo, el tiempo que ha estado activo y el total de conexiones realizadas.
- **Usuarios:** En esta pestaña, se obtiene una lista de usuarios con detalles como el nombre, servidor, la contraseña o los privilegios de cada usuario. Si se selecciona "Editar los privilegios" de cualquier usuario, es posible editar las instrucciones mySql que se pueden hacer con ese usuario, marcando o desmarcando los cuadros correspondientes:

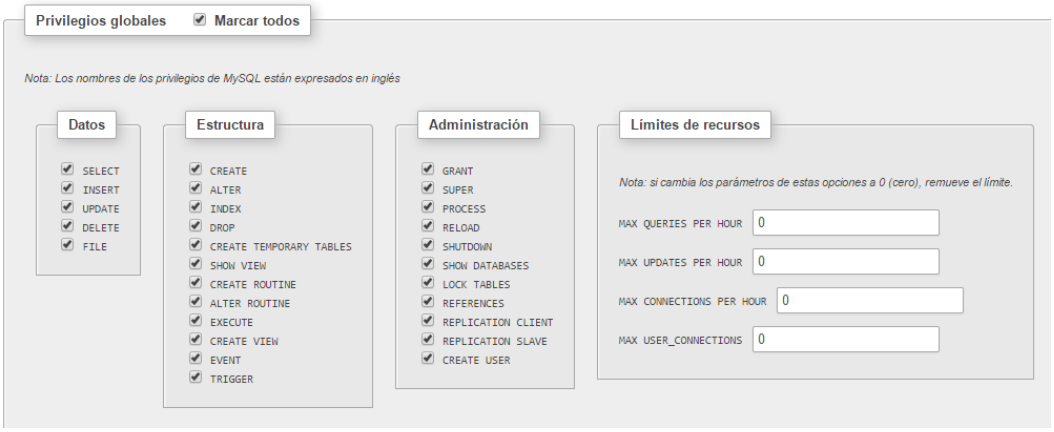


Ilustración 46: Pestaña "Editar los privilegios"

Se puede también editar otros detalles como el nombre, el servidor o la contraseña de usuario:

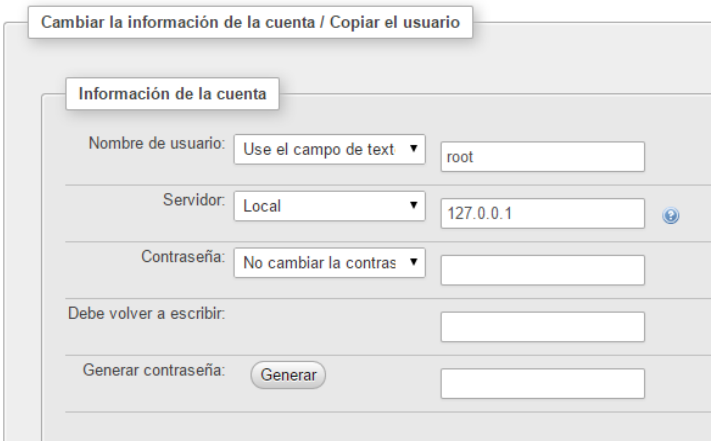


Ilustración 47: Cambiar información de la cuenta

Hay que tener en cuenta que si se desea cambiar el nombre o añadir o cambiar una contraseña se deberá acceder al archivo **"**config.inc.php" que está situado en *C:\wamp\phpmyadmin*. Allí se deberá buscar la línea que pone *root //MySQL user* y modificarlo por el nuevo nombre o contraseña que está justo debajo.

6.7. Conectividad:

6.7.1. Creación de base de datos y tablas

El primer paso a realizar es la implementación de la base de datos con los datos que la empresa colaboradora ha proporcionado en la base de datos del WampServer.

Los datos disponibles se encuentran ubicados en distintos archivos de texto (.txt) que representan las diferentes tablas en las que está dividida la base de datos. Y dentro de cada uno se encuentran los datos a implementar, con una línea por cada entrada de la base de datos y cada campo de cada entrada separado por un punto y coma (;). Las siguientes imágenes corresponden a los diferentes archivos de texto disponibles y al principio del contenido del archivo de texto TERCER.TXT respectivamente:

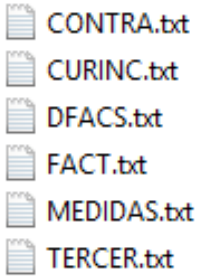


Ilustración 48: Archivos cedidos

```
CODIGO;CIF;NOMBRE
C06007353;27283854N;ACOSTA YAÑEZ, CORAL
C06008563;B97016729;AGRO LOHER S.L
C06008092;73772419B;ALIAGA CARRALERO, JULIAN
C06009218;Y1142623C;ANISOARA VASILICA BOUROS
C06007670;B96565346;ARICOTRANS, S.L.
C06000290;53203542B;AVELLANEDA ANDRES, NATALIA TERESA
C06008465;24385984G;AVELLANEDA ARCOS, ANTONI
C06000291;22696887G;AVELLANEDA CALATAYUD, JAVIER
C06000308;P4603100A;AYUNTAMIENTO DE ALGINET
C06008594;73583466A;AÑON PARRA, OSCAR
C06008635;73557823M;BAEZA PEREZ, JOSE ANTONIO
C06007476;20808434N;BAYARRI LOZANO, PEDRO
```

Ilustración 49: Estructura TERCER.txt

Cada archivo de texto contiene información necesaria para la aplicación, aunque no todos los campos de cada archivo serán necesarios ya que estos archivos provienen de una base de datos mayor empleada por la empresa que ha prestado los datos:

- **TERCER.txt** incluye información general sobre el cliente: nombre, CIF y código de cliente. Este archivo servirá para efectuar la identificación del usuario, utilizando el CIF y el código como usuario y contraseña respectivamente.
- **CONTRA.txt** contiene la información sobre los contadores: el código del elemento, la dirección en la que se encuentran, el usuario al que pertenece y la tarifa y potencia contratadas en ese momento.
- **FACT.txt** contiene los datos básicos de las facturas, incluyendo el código de la factura, el contador al que se le aplica, las fechas de inicio y de fin de las facturas, la base imponible, la cuota imponible y el importe total de la factura.
- **DFACS.txt** detalla cada factura desglosándola detalladamente en los diversos términos que se divide la factura: la energía consumida, la potencia contratada, el alquiler del contador y los impuestos aplicados.
- **MEDIDAS.txt** contiene la información de los detalles de la medida de energía consumida por cada contador, así como los días de inicio y fin de lectura.
- **CURINC.txt** incluye todas las medidas de todos los contadores por horas. Esta es con diferencia la tabla más extensa.

Para incluir estos datos en la base de datos, el primer paso será crear la base de datos con las tablas correspondientes en el WampServer. Para ello, abrimos phpMyAdmin y, en el menú de la izquierda, hay que pinchar en “Nueva” para crear una base de datos, se le pone nombre, en este caso “tfg” y se pincha en crear.

Una vez hecho esto, se procederá a crear todas las tablas con los correspondientes campos. Para conseguir una rápida integración, se incluirán todos los campos al igual que los datos proporcionados aunque no se vayan a usar. Se deben definir también el tipo de datos de cada campo:

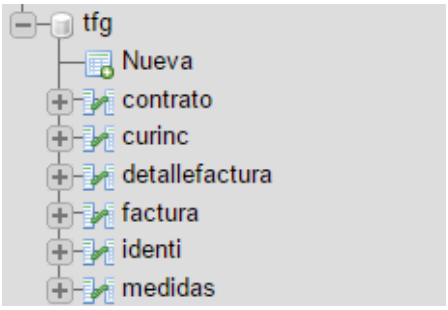


Ilustración 50: Estructura base de datos

#	Nombre	Tipo
1	CODELE	int(11)
2	CUPS	varchar(20)
3	DIRABO	varchar(70)
4	CODPRO	tinyint(4)
5	CODMUN	smallint(6)
6	NOMMUN	varchar(20)
7	NUMCON	varchar(15)
8	CODABO	varchar(9)
9	FECHAA	date
10	TARIFA	varchar(5)
11	POTENCIA	float

Ilustración 51: contrato

#	Nombre	Tipo
1	CODELE	int(11)
2	HORA	datetime
3	BANDERA	tinyint(4)
4	ACTIMP	tinyint(4)
5	ACTEXP	tinyint(4)
6	REACQ1	tinyint(4)
7	REACQ2	tinyint(4)
8	REACQ3	tinyint(4)
9	REACQ4	tinyint(4)

Ilustración 52: curinc

#	Nombre	Tipo
1	CODFAC	mediumint(9)
2	ORDFE	tinyint(4)
3	TERMIN	varchar(2)
4	CODPER	varchar(5)
5	CANT2	float
6	CANT	float
7	UNIDAD	varchar(3)
8	PRECIO	float
9	RESULT	float

Ilustración 53: detallefactura

#	Nombre	Tipo
1	CODFAC	mediumint(9)
2	CODELE	int(11)
3	DOCFAC	varchar(13)
4	FECFAC	date
5	FECINI	date
6	FECFIN	date
7	BASIMP	float
8	CUOIMP	float
9	IMPTOT	float

Ilustración 54: factura

#	Nombre	Tipo
1	CODIGO	varchar(9)
2	CIF	varchar(9)
3	NOMBRE	varchar(50)

Ilustración 55: identi

#	Nombre	Tipo
1	CODFAC	mediumint(9)
2	FECINI	date
3	FECFIN	date
4	DIAS	tinyint(4)
5	CODMAG	varchar(2)
6	CODPER	varchar(5)
7	LECINI	float
8	LECFIN	float
9	CANT	float

Ilustración 56: medidas

Una vez creadas las tablas en la base de datos, el último paso de la integración es trasladar los datos de los archivos de texto a la base de datos. Para ello, se utilizarán las siguientes líneas de código en MySql:

```
LOAD DATA INFILE 'D:\CURINC.txt'
INTO TABLE tfg.curinc
FIELDS TERMINATED BY ';'
```

Estas líneas de código copian el contenido de un archivo de texto en la tabla indicada, así que habrá que modificar el código para cada tabla.

Una vez finalizada la integración, hay que añadir dos cosas más a la base de datos:

En primer lugar, hay que añadir una columna en la tabla “identi” llamada “CONTRASEÑA” en la que se copiaran los valores de la columna “CODIGO”. Esta columna servirá para almacenar la contraseña de cada usuario siendo el valor inicial de estos el código de usuario. Por tanto, la estructura de la tabla “identi” quedará de la siguiente forma:

	#	Nombre	Tipo
☐	1	CODIGO	varchar(9)
☐	2	CIF	varchar(9)
☐	3	NOMBRE	varchar(50)
☐	4	CONTRA	varchar(20)

Ilustración 57: identi final

En segundo lugar, hay que crear una tabla nueva llamada “mensajes”, en la que se almacenarán los mensajes enviados por los usuarios mediante la opción “Enviar mensaje soporte”. Esta tabla tendrá la siguiente estructura:

	#	Nombre	Tipo
☐	1	asunto	varchar(500)
☐	2	mensaje	varchar(5000)

Ilustración 58: mensajes

Esto significa que el asunto de los mensajes no podrá tener más de 500 caracteres, y el texto del mensaje no podrá pasar de los 5000 caracteres.

6.7.2. Consultas con archivos PHP

Para efectuar las consultas sobre la base de datos, el método empleado será el de crear una serie de archivos PHP con scripts que, dados unos parámetros, realicen unas consultas predefinidas sobre la base de datos y devuelvan los datos necesarios codificados en JSON. Así, el usuario no puede acceder directamente a la base de datos sino que accederá a los archivos PHP y obtendrá el resultado devuelto por el script del archivo PHP.

Los archivos creados para el trabajo son los siguientes:

- db_config.php
- login.php
- facturas.php
- detallefactura.php
- consumos.php
- evolución.php
- cambiacontra.php
- mensaje.php

db_config.php

En este archivo se definen todas las variables necesarias para conectarse a la base de datos:

```
<?php
define('DB_USER', "guest");
define('DB_PASSWORD', "");
define('DB_DATABASE', "tfg");
define('DB_SERVER', "localhost");
?>
```

Así pues, se incluirá la siguiente línea de código en los demás archivos para definir las variables:

```
require_once __DIR__ . '/db_config.php';
```

Esto permitirá hacer cambios en la configuración en caso necesario simplemente modificando este archivo.

login.php

Con este archivo se comprobará que el usuario y contraseña introducidos corresponden a alguna entrada en la base de datos. Si es así, obtendrá los datos de los contadores que posee el usuario y sus direcciones para utilizarlo en la siguiente pantalla. También obtendrá los detalles de contrato de cada contador como la tarifa y la potencia para ser utilizado más adelante:

```
<?php
$response = array();
require_once __DIR__ . '/db_config.php';
$con = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD,DB_DATABASE) or die(mysql_error());
if (isset($_GET['usuario']) && isset($_GET['pass'])){
        $usuario = $_GET['usuario'];
        $contraseña = $_GET['pass'];
        $query = "SELECT *FROM identi WHERE CIF = '".$usuario."' AND CODIGO =
                '".$contraseña."'";
        $resultado = mysqli_query($con, $query);
        if ($resultado==false) echo "Busqueda fallida";
        if (mysqli_num_rows($resultado) > 0) {
                $aux = mysqli_fetch_array($resultado);
                $response["success"] = "Y";
                $response["nombre"] = $aux["NOMBRE"];
                $response["contadores"] = array();
                $query2 = "SELECT *FROM contrato WHERE CODABO =
                        '".$contraseña."'";
                $contratos = mysqli_query($con, $query2);
                while ($row = mysqli_fetch_array($contratos)){
                        $contadores = array();
                        $contadores["codigo"] = $row["CODELE"];
                        $contadores["cups"] = $row["CUPS"];
                        $contadores["direccion"] = $row["DIRABO"];
                        $contadores["tarifa"] = $row["TARIFA"];
                        $contadores["potencia"] = $row["POTENCIA"];
                        array_push($response["contadores"],$contadores);
```

```
                    }
            } else {  $response["success"] = "N";      }
    }else{    $response["success"] = "N";      }
    mysqli_close($con);
    echo json_encode($response);
    ?>
```

Esto es un ejemplo del código completo utilizado en el archivo PHP. A partir de este momento solamente se mostrarán fragmentos de código que sirvan para la explicación del archivo correspondiente.

facturas.php

En este archivo, se obtiene el código de identificación de un contador y se devuelve un array codificado en JSON con los códigos de factura vinculados a ese contador y las fechas de inicio y fin.

Se obtendrá el código de identificación del contador mediante la siguiente línea de código:

```
$cup = $_GET['cup'];
```

Para efectuar la consulta que selecciona las facturas que contengan el código de identificación se utilizará el siguiente código:

```
$query = "SELECT *FROM factura WHERE CODELE = '".$cup."'";
$resultado = mysqli_query($con, $query);
```

Seguidamente, para guardar el resultado en un array se utilizará este fragmento:

```
while ($row = mysqli_fetch_array($resultado)){
        $facturas = array();
        $facturas["fecini"]=$row["FECINI"];
        $facturas["fecfin"]=$row["FECFIN"];
        $facturas["codigo_factura"]=$row["CODFAC"];
        array_push($response["facturas"],$facturas);
}
```

Finalmente, para codificar el resultado obtenido en JSON y devolverlo como respuesta será necesaria la siguiente línea:

```
echo json_encode($response);
```

detallefactura.php

Este archivo recibirá el código de la factura elegida, y efectuara cuatro búsquedas, una para cada apartado de la factura (energía, potencia, alquiler e impuestos):

```
$codigo = $_GET['codigo'];
$query_energia = "SELECT *FROM detallefactura WHERE CODFAC = '".$codigo."' AND TERMIN = 'EN'";
$query_potencia = "SELECT *FROM detallefactura WHERE CODFAC = '".$codigo."' AND TERMIN = 'PO'";
$query_alquiler = "SELECT *FROM detallefactura WHERE CODFAC = '".$codigo."' AND TERMIN = 'AL'";
$query_impuestos = "SELECT *FROM detallefactura WHERE CODFAC = '".$codigo."' AND TERMIN = 'IM'";
$resultado_energia = mysqli_query($con, $query_energia);
$resultado_potencia = mysqli_query($con, $query_potencia);
$resultado_alquiler = mysqli_query($con, $query_alquiler);
$resultado_impuestos = mysqli_query($con, $query_impuestos);
```

Una vez efectuado las consultas, guardara cada tipo de dato en un array y lo pasará codificado en JSON tal como se ha hecho en los anteriores archivos.

consumos.php

En este archivo, se recibirán dos fechas codificadas como aa-mm-dd y se devolverán todos los consumos efectuados por horas desde la primera fecha recibida a las 0:00h hasta la segunda fecha a las 24:00h.

La consulta realizada por este archivo será la siguiente:

```
$query = "SELECT *FROM curinc WHERE CODELE = '".$cup."' AND HORA BETWEEN '".$fecini."' AND '".$fecfin." 23:00:00'";
$resultado = mysqli_query($con, $query);
```

Todo lo demás se realizará de manera similar a los archivos anteriores.

evolucion.php

Este archivo realiza una función similar al archivo consumos.php, salvo que recibirá dos pares de fechas y devolverá los consumos de los dos periodos dados.

Se efectuarán pues, dos consultas a la base de datos:

```
$query1 = "SELECT *FROM curinc WHERE CODELE = '".$cup."' AND HORA BETWEEN '".$fecini1."' AND '".$fecfin1." 12:00:00'";

$resultado1 = mysqli_query($con, $query1);

$query2 = "SELECT *FROM curinc WHERE CODELE = '".$cup."' AND HORA BETWEEN '".$fecini2."' AND '".$fecfin2." 12:00:00'";

$resultado2 = mysqli_query($con, $query2);
```

cambiacontra.php

Este archivo cambiará la contraseña del usuario por una de su elección. Para ello, recibirá como parámetros el código de usuario, la contraseña actual, y la nueva contraseña.

En este caso no se efectuará ningún tipo de consulta pues el objetivo no es recibir ningún dato sino simplemente el cambiar un dato existente.

El código utilizado para realizar dicha función es el siguiente:

```
if (isset($_GET['antigua']) && isset($_GET['nueva'])&& isset($_GET['codigo'])){
        $antigua = $_GET['antigua'];
        $nueva = $_GET['nueva'];
        $usuario = $_GET['codigo'];
        $query = "UPDATE identi SET CONTRA = '".$nueva."' WHERE CONTRA = '".$antigua."'     AND CODIGO = '".$usuario."'";
        if (mysqli_query($con, $query)) {
                $response["success"] = "Y";
        } else {
                $response["success"] = "N";
        }
}
```

mensaje.php

Este archivo realizará la función de recibir un mensaje del usuario y almacenarlo en la base de datos para su posterior lectura por parte del administrador de la aplicación.

En este caso tampoco se efectuará una consulta en la base de datos sino que se insertarán nuevas filas en una tabla designada específicamente para tal uso. El código utilizado será el siguiente:

```
if (isset($_GET['asunto']) && isset($_GET['mensaje'])){
        $asunto = $_GET['asunto'];
        $mensaje = $_GET['mensaje'];
        $query = "INSERT INTO `mensajes`(`asunto`, `mensaje`) VALUES
        ('".$asunto."','".$mensaje."')";
        if (mysqli_query($con, $query)) {
                $response["success"] = "Y";
        } else {
                $response["success"] = "N";
        }
}
```

6.7.3. Abrir y redireccionar puertos router

Para poder acceder a los PHP creados que realizan las consultas sobre la base de datos, hay que abrir y redireccionar el puerto 80 (puerto utilizado por el protocolo HTTP) del router al puerto del servidor local. En el caso de este trabajo, fue necesario contactar con la empresa proveedora de internet para desbloquear la entrada de peticiones en el puerto 80 ya que se encontraban cerrados por motivos de seguridad y no podía hacerse manualmente.

En cuanto al redireccionamiento, hay que acceder a la configuración del router. En este caso, al tener contratado una línea de fibra óptica, el router instalado se trata de una ONT, y el usuario y contraseña son también suministrados por la empresa proveedora de internet.

Ilustración 59: Identificación router

Una vez dentro del router, hay que acceder a la pestaña “Application”, y dentro de ella, a la subpestaña “NAT”. Aquí se configurarán los parámetros necesarios para redireccionar las peticiones HTTP:

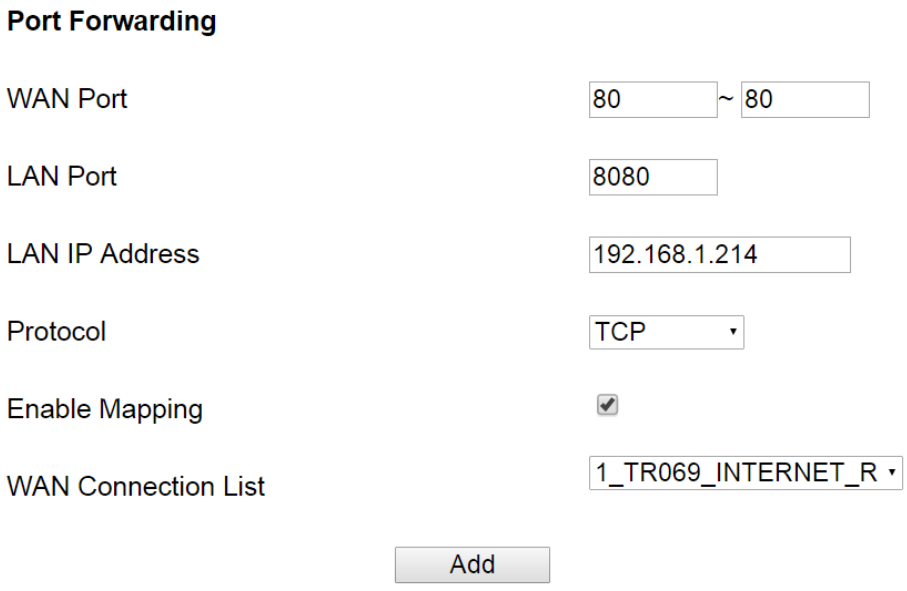


Ilustración 60: NAT

- **WAN Port:** Rango de puertos del router que se desea redireccionar, en este caso solamente el puerto 80.
- **LAN Port:** Se indica a qué puerto interno se desea redireccionar las peticiones recibidas. Puede ser cualquier puerto, aunque se evitarán los puertos ya usados por otras aplicaciones. Hay que tener en cuenta que a continuación hay que configurar el servidor WAMP para que "escuche" las peticiones del puerto que se le hayan indicado.
- **LAN IP Address:** Dirección IP de la red interna del ordenador en el que esté instalado el WAMPServer.
- **Protocol:** Protocolo utilizado para la transferencia de archivos. Los posibles protocolos utilizados son TCP, UDP o ambos. En este caso se utilizará el protocolo TCP.
- **WAN Connection List:** Hay que elegir desde que punto de conexión a internet se desea aplicar esta configuración. En este caso solamente existe una opción que será la que elegiremos.

Finalizado esto, simplemente hay que pulsar el botón "Add" y el redireccionamiento se aplicará en el router.

6.7.4. Permisos Android

Para proteger ciertos recursos y características especiales, Android define un esquema de permisos. Toda aplicación que acceda a estos recursos está obligada a declarar su intención de usarlos. En caso de que una aplicación intente acceder a un recurso del que no ha solicitado permiso, se generará una excepción de permiso y la aplicación será interrumpida inmediatamente.

Mediante el tag <uses-permisions> se especifican que permisos va a necesitar la aplicación para poder ejecutarse, además, el usuario deberá aceptar dichos permisos antes de instalarlos.

Por tanto, para que la aplicación sea capaz de acceder a servicios Web, esta deberá incluir el permiso correspondiente en AndroidManifest.xml. Para ello, se debe añadir la línea de código siguiente al elemento "manifest" dentro del archivo:

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

Esto permite acceder a alguna parte protegida del API que proporciona el sistema Android. Esta declaración alertará a los usuarios que la aplicación utilizará ciertos permisos.

6.7.5. Conexión HTTP y JSON

Una de las vías más comunes para que una aplicación se comunique con el mundo exterior es establecer una conexión con servidores Web. Los servidores Web utilizan el protocolo HTTP (Hypertext Transfer Protocol, Protocolo para la transferencia de hipertexto) para contactar con los clientes Web. Este protocolo tiene dos ventajas; que pesa muy poco y se utiliza mucho.

Aparte de los servicios Web HTTP, hay otro tipo de servicios Web que se utiliza para el intercambio de datos. Se trata de JSON (JavaScript Object Notation, Notación para objetos JavaScript), que es un formato muy ligero y que se ha diseñado para que resulte más fácil de procesar, tanto por los humanos como por las máquinas. JSON utiliza llaves ("{" y "}"), dos puntos (":") y corchetes ("[" y "]") para organizar los pares nombre/valor y los array.

En este trabajo, para contactar con los servicios Web y obtener las cadenas JSON de resultados, se conectará al servidor utilizando los métodos HTTP y se obtendrá una cadena de resultados en formato JSON.

Un ejemplo de código utilizado para efectuar la conexión a la base de datos, es el siguiente:

```
public String leerJSON(String URL) {
   StringBuilder stringBuilder = new StringBuilder();
   HttpClient httpClient = new DefaultHttpClient();
   HttpGet httpGet = new HttpGet(URL);
   try{
      HttpResponse response = httpClient.execute(httpGet);
      StatusLine statusLine = response.getStatusLine();
      int statusCode = statusLine.getStatusCode();
      if (statusCode==200){
         HttpEntity entity = response.getEntity();
         InputStream inputStream = entity.getContent();
         BufferedReader reader = new BufferedReader( new
                                 InputStreamReader(inputStream));
         String line;
         while((line = reader.readLine()) != null){   stringBuilder.append(line);   }
         inputStream.close();
      }else{   Log.d("JSON","Fallo en la conexión");   }
   }catch(Exception e){   Log.d("readJSON",e.getLocalizedMessage());   }
   return stringBuilder.toString();
}
```

Este es el método utilizado para la identificación del usuario. Este método devolverá un String que estará codificado en JSON. La forma de ejecutar el método y guardar el resultado será el siguiente:

```
user = usuario.getText().toString();

pass = contraseña.getText().toString();

URL = "http://5.56.1.1/login.php?usuario="+user+"&&pass="+pass;

resultado = return readJSON(URL);
```

Se observa cómo se definen las variables que vamos a utilizar y se codifican como variables en una URL (). Hecho esto, se ejecuta el método leerJSON() con la URL que se acaba de crear y se guarda el resultado en la variable resultado. Un ejemplo del resultado obtenido codificado en JSON podría tener la siguiente forma:

URL = http://localhost/login.php?usuario=B96815766&&pass=C06003443

resultado = “{ "success" : "Y" , "nombre" : "PROMLLACHAN, S.L." , "contadores" : [{ "codigo" : "361011258" ," cups" : "ES0361000361011258YQ" , "direccion" : "HORT DE BONET 19 5" , "tarifa" : "2.0A" , "potencia" : "3.45" },{ "codigo" : "361011259" , "cups" : "ES0361000361011259YV" , "direccion" : "HORT DE BONET 19 6" , "tarifa" : "2.0A" , "potencia" : "3.45" },{ "codigo" : "361011260" , "cups" : "ES0361000361011260YH" , "direccion" : "HORT DE BONET 19" , "tarifa" : "2.0A" , "potencia" : "6.93" },{ "codigo" : "361011264" , "cups" : "ES0361000361011264YE" , "direccion" : "HORT DE BONET 21 3" , "tarifa" : "2.0A" , "potencia" : "1.15" },{ "codigo" : "361011267" , "cups" : "ES0361000361011267FW" , "direccion" : "HORT DE BONET 21" , "tarifa" : "2.0A" , "potencia" : "6.93" },{ "codigo" : "361011270" , "cups" : "ES0361000361011270FM" , "direccion" : "HORT DE BONET 23 3" , "tarifa" : "2.0A" , "potencia" : "3.45" },{ "codigo" : "361011271" , "cups" : "ES0361000361011271FY" , "direccion" : "HORT DE BONET 23 4" , "tarifa" : "2.0A" , "potencia" : "3.45" },{ "codigo" : "361011273" , "cups" : "ES0361000361011273FP" , "direccion" : "HORT DE BONET 23" , "tarifa" : "2.0A" , "potencia" : "6.93" },{ "codigo" : "361011540" , "cups" : "ES0361000361011540HE" , "direccion" : "HORT DE BONET 23" , "tarifa" : "2.0A" , "potencia" : "3.45"}]}”

Una vez obtenido el objeto JSON, se puede obtener sus parámetros con el método getString(). En este caso, se pide el valor del elemento “success” que indicará si la identificación ha sido válida o no.

```
JSONObject jsonObject = new JSONObject(resultado);

success = jsonObject.getString("success");
```

Finalmente, una vez obtenido el valor de “success”, se pasará a comprobar su valor y, en caso de que haya resultado positivo (“Y”), se procederá al inicio de la siguiente pantalla, en la que se elegirán entre los contadores del usuario. También se obtendrá un elemento tipo JSONArray que contendrá les datos relativos a los contadores que posea el usuario:

```
try{
   if(success.equals("Y")){
      advertencia.setVisibility(View.INVISIBLE);
      Intent i = new Intent(Login.this, Titular.class);
      i.putExtra("cups", jsonObject.getJSONArray(“cups”));
      startActivity(i);
   }else{
      advertencia.setVisibility(View.VISIBLE);
   }
}catch(Exception e){
}
```

6.8. Hilos de ejecución

Todos los componentes de una aplicación Android, tanto las actividades, los servicios, o los *broadcast receivers* se ejecutan en el mismo hilo de ejecución, llamado *hilo principal*, *main thread* o *GUI thread.*

Es por ello, que cualquier operación larga o costosa que se realice en este hilo bloqueará la ejecución del resto de componentes de la aplicación y por supuesto también la interfaz, produciendo al usuario un efecto evidente de lentitud, bloqueo, o mal funcionamiento en general, algo que se debería evitar. Además, Android monitoriza las operaciones realizadas en el hilo principal y detecta aquellas que estén en ejecución más de 5 segundos, en cuyo caso se muestra un mensaje con el texto "*Application Not Responding*" (ANR) y el usuario debe decidir entre forzar el cierre de la aplicación o esperar a que termine.

Así pues, siempre que exista la posibilidad de que una tarea pueda bloquear el hilo del interfaz de usuario, como efectuar cálculos complejos o acceder a la red, hay que ejecutar esas mismas tareas en un nuevo hilo de ejecución, para que realice este trabajo intensivo. De esta forma no se bloquea el hilo principal, que puede seguir atendiendo los eventos de usuario.

Un hilo (Thread) es una parte de código que se encarga de realizar alguna acción a la misma vez que se está ejecutando otra. Proporciona su propia unidad de ejecución, y define sus propios argumentos, variables y pila de llamada a métodos. Funciona por tanto, de manera paralela al hilo principal.

Se pueden crear tantos hilos como se quiera, teniendo en cuenta que el hilo deja de formar parte de la aplicación y funciona de manera independiente. El hilo no puede modificar ni insertar datos en el hilo principal, ya que esto causaría un error.

Siguiendo la manera tradicional, se pueden crear hilos usando la clase Thread e implementando el método "run()", sin embargo, el SDK de Android provee una clase de lo más útil que facilitará mucho la tarea. La clase en cuestión es AsyncTask, y contiene un conjunto de métodos a sobrescribir que se ejecutan en el hilo correspondiente para llevar a cabo cualquier operación que requiera actualizaciones en la interfaz.

6.8.1. AsyncTask

Clase genérica de la que se debe heredar. Define tres tipos genéricos:

- **Params**: Tipo de parámetro que se recibirá como entrada para la tarea en el método **doInBackground(Params).**
- **Progress**: Parámetros para actualizar el hilo principal o **UIThread**.
- **Result**: Es el resultado devuelto por el procesamiento en segundo plano.

Los métodos que se heredan de la clase AsyncTask son los siguientes:

- OnPreExecute(): Método llamado antes de iniciar el procesamiento en segundo plano.
- doInBackground(Params...): En este método se define el código que se ejecutará en segundo plano. Recibe como parámetros los declarados al llamar al método execute(Params).

- OnProgressUpdate(Progress...): Este método es llamado por publishProgress(), dentro de doInBackground(Params) (su uso es muy común para por ejemplo actualizar el porcentaje de un componente ProgressBar).
- onPostExecute(Result...): Este método es llamado tras finalizar doInBackground(Params). Recibe como parámetro el resultado devuelto por doInBackground(Params).
- OnCancelled(): Se ejecutará cuando se cancele la ejecución de la tarea antes de su finalización normal.

La AsyncTask requiere tres tipos parametrizados. El primero indica de qué tipo será la lista de objetos que le llegue al método doInBackground y el segundo al tipo de los que se pasarán a onProgressUpdate. El último se corresponde con el resultado final, y por tanto será el valor que retorne doInBackground y que reciba onPostExecute.

En este trabajo, se ha utilizado la clase AsyncTask para realizar todas las conexiones y consultas a la base de datos. Para ello, se creará una nueva clase para cada tipo de consulta que se desee realizar:

```
private class ConsumosSimulacionJSON extends AsyncTask<String, String, String>
```

Los métodos utilizados serán OnPreExecute(), doInBackground() y onPostExecute():

OnPreExecute():

En este método se creará una actividad que se mostrará por pantalla con el mensaje "conectando":

Ilustración 61: Conectando

El código utilizado para realizar esta tarea será siempre el mismo, con muy pocas variaciones:

```
protected void onPreExecute() {
   super.onPreExecute();
   pDialog = new ProgressDialog(Login.this);
   pDialog.setMessage("Conectando");
   pDialog.setIndeterminate(false);
   pDialog.setCancelable(true);
   pDialog.show();
}
```

doInBackground(Params...)

En este método se creará la String con la dirección URL y añadiendo las variables necesarias para efectuar la consulta. En el siguiente ejemplo, se obtienen las variables "usuario" y "contraseña" y se efectua la consulta sobre el archivo login.php añadiendo las dos variables:

```
protected String doInBackground(String... args) {
   user = usuario.getText().toString();
   pass = contraseña.getText().toString();
   URL = "http://5.56.1.1/login.php?usuario="+user+"&&pass="+pass;
   return readJSON(URL);
}
```

onPostExecute(Result...)

En este método se incluirá todas las operaciones que se deberán hacer con el resultado de la consulta. Un ejemplo es el método utilizado para obtener las fechas y códigos de las facturas disponibles:

```
protected void onPostExecute(String result) {
   try{
      JSONObject jsonObject = new JSONObject(result), aux;
      facturas = jsonObject.getJSONArray("facturas");
     success = jsonObject.getString("success");
      String[] fechas = new String[facturas.length()];
```

```
            String[] codigos_facturas = new String[facturas.length()];
            for ( int j = 0; j < facturas.length(); j++){
               aux = facturas.getJSONObject(j);
               fechas[j] = aux.getString("fecini") + "\n" + aux.getString("fecfin");
               codigos_facturas[j] = aux.getString("codigo_factura");
            }
            if(success.equals("Y")){
               Intent i = new Intent(MainMenu.this, UltimasFacturas.class);
               i.putExtra("fechas", fechas);
               i.putExtra("codigos_facturas", codigos_facturas);
               startActivity(i);
            }
         }catch(Exception e){
         }
         pDialog.dismiss();
      }
```

Como se puede observar, se almacena el resultado codificado en JSON en la variable jsonObject, extrae los elementos que contiene y almacena las fechas en un array de Strings y los códigos de los contadores en otro array de Strings. A continuación, en este caso, si la variable donde se ha almacenado el Array que indica si la identificación ha tenido éxito (success) es positiva (“Y”), se pasará a entrar en la pantalla de selección de contador. En caso negativo (“N”), se mostrará el mensaje de advertencia.

6.9. Simulación de tarifas

El en apartado “Detalles de tarifa” se efectúa una simulación del coste de la energía. Para ello, se realizan una serie de pasos descritos a continuación:

6.9.1. Obtención de los datos

Para realizar la simulación es necesario obtener todos los datos de consumo en cada una de las facturas pasadas. En este trabajo se obtendrán los datos de las últimas 4 facturas, aunque se puede modificar fácilmente cambiando el valor de la variable “num_meses”. Para obtener los datos se ha utilizado el siguiente código:

```
private class ConsumosSimulacionJSON extends AsyncTask<String, String, String> {
   private ProgressDialog pDialog;
   protected void onPreExecute() {
      super.onPreExecute();
      pDialog = new ProgressDialog(DetallesTarifa.this);
      pDialog.setMessage("Procesando "+ (contador+1)+ " de " +
                        Integer.toString(num_meses));
      pDialog.setIndeterminate(false);
      pDialog.setCancelable(true);
      pDialog.show();
   }
   protected String doInBackground(String... args) {
      URL = "http://5.56.1.1/consumos.php?cup=" + cup + "&&fecini=" +
                        fecini[contador] + "&&fecfin=" + fecfin[contador];
      return Login.readJSON(URL);
   }
   protected void onPostExecute(String result) {
      try {
         JSONObject jsonObject = new JSONObject(result), aux;
         JSONArray consumos = jsonObject.getJSONArray("consumos");
         String[] hora = new String[consumos.length()];
         int[] activa = new int[consumos.length()];
         int[] reactiva = new int[consumos.length()];
         for (int j = 0; j < consumos.length(); j++) {
```

```
                aux = consumos.getJSONObject(j);
                hora[j] = aux.getString("hora");
                activa[j] = aux.getInt("activa");
                reactiva[j] = aux.getInt("reactiva");
            }
            datos[contador]=new DatosFacturas(hora,activa,reactiva);
            contador++;
        } catch (Exception e) {
        }
        pDialog.dismiss();
        if (contador ==num_meses){
            Intent i = new Intent(DetallesTarifa.this, AnalisisTarifa.class);
            for (int j=0;j<num_meses;j++){
                i.putExtra("hora"+Integer.toString(j),datos[j].hora);
                i.putExtra("activa"+Integer.toString(j),datos[j].activa);
                i.putExtra("reactiva"+Integer.toString(j),datos[j].reactiva);
            }
            i.putExtra("potencia_contratada",potenciacontratada);
            i.putExtra("tarifa_contratada",tarifa_contratada);
            i.putExtra("fechas",fechas);
            contador = 0;
            startActivity(i);
        }else {
            new ConsumosSimulacionJSON().execute();
        }
    }
```

En este fragmento de código, se ha utilizado una estructura de datos personalizada, creada por una clase llamada DatosFacturas y que contiene 2 arrays de enteros y 1 array de Strings. También se utiliza la recursividad, que consiste en que el método se llame a sí mismo para ejecutar la siguiente consulta de datos cuando acabe la consulta anterior. Este método es mucho más sencillo de utilizar que el iterativo.

6.9.2. Agrupación de datos

Una vez obtenidos los datos de consumo necesario, es preciso agrupar todos estos datos por facturas para poder manejarlos cómodamente. Para ello se crearán 2 arrays, una para los consumos totales de cada mes, y otro con 2 dimensiones que representaran los consumos de cada periodo horario en cada factura:

```
public void simulardatos(){
   for (int i = 0;i<num_meses;i++){
      for (int j=0;j<datos[i].activa.length;j++){
         aux = datos[i].hora[j];
         horas = aux.split(" ");
         horas = horas[1].split(":");
         energia_total[i]+=datos[i].activa[j];
         if (Integer.parseInt(horas[0])<=12||Integer.parseInt(horas[0])>22){
            energia_periodos[i][0]+=datos[i].activa[j];
         }else{
            energia_periodos[i][1]+=datos[i].activa[j];
         }
      }
   }
}
```

6.9.3. Simulación de precios

Una vez agrupados los consumos, se procede a la simulación de las tarifas, para ello, se asigna un precio a la energía según su tarifa y periodo y se multiplican los consumos obtenidos con el precio de la energía de la siguiente manera:

```
   public void simularprecios(){
      precios = new float[3];
      if (potencia<10){
         precios[0]= (float)0.125613;
         precios[1]= (float)0.069132;
         precios[2]= (float)0.148832;
      }else{
```

```
            precios[0]= (float)0.139478;
            precios[1]= (float)0.077776;
            precios[2]= (float)0.162267;
        }
        for (int i = 0;i<num_meses;i++){
            precio_total[i]+= precios[0]*(float)energia_total[i];
            precio_periodos[i] += precios[1] * (float) energia_periodos[i][0] + precios[2] *
                                  (float) energia_periodos[i][1];
        }
    }
```

6.10. Listas dinámicas con ListView

ListView es un tipo de Layout que dispone los elementos que contiene en una lista. La diferencia es que los elementos que contiene el ListView no están definidos previamente sino que son insertados dinámicamente a partir de un array de elementos. Esto permite poder mostrar listas dinámicas que cambien de contenido, como por ejemplo el número y contenido de los contadores de cada usuario.

Para utilizar ListView, se usará el elemento <ListView> y se añadirá a la interfaz de usuario. Para rellenar la lista con elementos, se necesita una referencia a ListView. Para ello, se utilizará el método findViewById():

```
List = (ListView) findViewById(R.id.listView);
```

Una vez establecida dicha referencia, se creará el objeto ArrayAdapter, que contendrá un array de Strings con los elementos con los que se rellenará el ListView, y se vinculará el ArrayAdapter creado con el ListView:

```
ArrayAdapter<String> adapter = new ArrayAdapter<> (this, R.layout.elemento_contador, R.id.textViewContador, cups);
list.setAdapter(adapter);
```

Para controlar el proceso de selección de un elemento que se encuentra dentro de ListView, se llamará al método setOnItemClickListener() a fin de asignarle una instancia de la clase OnItemClickListener y sobrescribir el método onItemClick():

```
list.setOnItemClickListener(new AdapterView.OnItemClickListener() {
   @Override
   public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
      Intent i = new Intent(Titular.this, MainMenu.class);
      i.putExtra("codigo_cup",codigos[position]);
      i.putExtra("tarifa",tarifas[position]);
      i.putExtra("potencia",potencias[position]);
      i.putExtra("codabo",extra.getString("codabo"));
      startActivity(i);
   }
});
```

6.11. Creación de gráficas con GraphView

GraphView es una librería gratuita para Android que permite crear gráficas y diagramas de forma sencilla. En esta aplicación se usará esta librería. A continuación se explicarán los pasos para la instalación y manejo de la librería.

Instalación: Para poder añadir la librería GraphView al proyecto Android, simplemente hay que añadir la siguiente línea al archivo build.gradle del proyecto:

```
compile 'com.jjoe64:graphview:4.0.1'
```

Añadir fragmento: La gráfica se añadirá como un fragmento en la actividad correspondiente, en este caso en la actividad "Consumos". El código XML correspondiente al elemento será el siguiente:

```
<com.jjoe64.graphview.GraphView
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:id="@+id/graph"
   android:onClick="entrarDatosGrafica" />
```

Dibujar gráfica: Para dibujar la gráfica se ha implementado un método en la clase java "consumos". Este método distinguirá entre si los consumos se refieren a un solo día o a varios. En el caso de que el consumo a analizar sea de un solo día, se dibujará una gráfica que muestre el consumo por horas. En caso contrario, la gráfica mostrará el consumo por días. A continuación se muestra el código utilizado para dibujar las gráficas necesarias:

```
private void dibujarGrafica() {
   graph.removeAllSeries();
   if (hora.length<=24) {
      valores = new DataPoint[hora.length];
      for (int i = 0; i < hora.length; i++) {
         valores[i] = new DataPoint(i, activa[i]);
      }
      LineGraphSeries<DataPoint> series = new LineGraphSeries<DataPoint>(valores);
      graph.addSeries(series);
      graph.getGridLabelRenderer().setHorizontalLabelsVisible(false);
```

```
            graph.getGridLabelRenderer().setVerticalLabelsVisible(true);
            graph.getViewport().setXAxisBoundsManual(true);
            graph.getViewport().setYAxisBoundsManual(true);
            graph.getViewport().setMinY(0);
            graph.getViewport().setMaxX(24);
        }
        else{
            valores = new DataPoint[hora.length/24];
            for (int i = 0; i < hora.length/24; i++){
                int auxiliar=0;
                for (int j = 0; j < 24; j++){
                    auxiliar+=activa[24*i+j];
                }
                valores[i]=new DataPoint(i,auxiliar);
            }
            LineGraphSeries<DataPoint> series = new LineGraphSeries<DataPoint>(valores);
            graph.addSeries(series);
            graph.getGridLabelRenderer().setHorizontalLabelsVisible(false);
            graph.getGridLabelRenderer().setVerticalLabelsVisible(true);
            graph.getViewport().setXAxisBoundsManual(true);
            graph.getViewport().setYAxisBoundsManual(true);
            graph.getViewport().setMinY(0);
            graph.getViewport().setMaxX(valores.length);
        }
    }
```

6.12. Funciones secundarias

6.12.1. E-mail y llamadas

Es posible utilizar el Intent para ejecutar aplicaciones externas.

Para enviar un mensaje de correo electrónico desde una aplicación, se utilizará el siguiente código:

```
public void correo(View v){
   Intent i = new Intent(Intent.ACTION_SEND);
   i.setData(Uri.parse("mailto:"));
   i.putExtra(Intent.EXTRA_EMAIL,"info@electricadealginet.com");
   i.setType("message/rfc822");
   startActivity(Intent.createChooser(i,"Email"));
}
```

Incluiremos este método al pulsar sobre la dirección de correo en la pantalla “Como encontrarnos”.

Para acceder a la pantalla de llamada al pulsar el número de teléfono mostrado en la pantalla de “Como encontrarnos” se utilizará el siguiente código:

```
public void llamada(View v){
   Intent i = new Intent(Intent.ACTION_DIAL,
        Uri.parse("tel:961751134"));
   startActivity(i);
}
```

6.12.2. Google Maps

Google Maps es una de las aplicaciones incluidas en la plataforma Android. Además de trabajar directamente con la aplicación Maps, se pueden incluir en las aplicaciones. El SDK de Android proporciona una vista MapFragment que muestra un mapa (con los datos que obtiene del servicio Google Maps).

Conviene destacar que a diferencia de Android, Google Maps no es un software libre, por lo que está limitado a una serie de condiciones de servicio. Se puede usar de forma gratuita siempre que no se soliciten más de 15.000 codificaciones geográficas al día. Se pueden incluir anuncios en los mapas e incluso se puede usar en aplicaciones de pago.

Para crear una nueva aplicación que use la API v2 de Google Maps Android se deben seguir ciertos pasos. Muchos de ellos solamente deberán ser seguidos una sola vez para efectuar la correcta configuración. Después, se pueden añadir tantos mapas se deseen de forma más sencilla. Los pasos a seguir para añadir un mapa a una aplicación Android por primera vez son los siguientes:

- Instalación del SDK de Android
- Descargar y configurar el Google Play Services (Servicios de Google Play), que incluye la API de Google Maps para Android.
- Obtener una clave de API. Para ello, tendrá que dar de alta un proyecto en la consola de desarrolladores de Google, y obtener un certificado de firma para su aplicación.
- Añada la clave API para su aplicación
- Añadir los ajustes necesarios en el manifiesto de la aplicación.
- Añadir un mapa en la aplicación.

Instalación del SDK de Android:

Al estar utilizando el programa Android Studio para la realización del trabajo, en la descarga del propio programa se incluye el SDK de Android, por lo que esta paso ya está cubierto.

Descargar y configurar el Google Play Services:

La API de Maps se distribuye como parte de los servicios del SDK de Google Play, así que se debe descargar el Google Play Services mediante el Android SDK Manager. Para ello se ejecutará el Android SDK Manager, se seleccionará la casilla de verificación del elemento Google Google Play Service que se encuentra dentro de la carpeta Extras y se hará clic en el botón “Install 1 package...” para instalar la librería de servicios para Google Play.

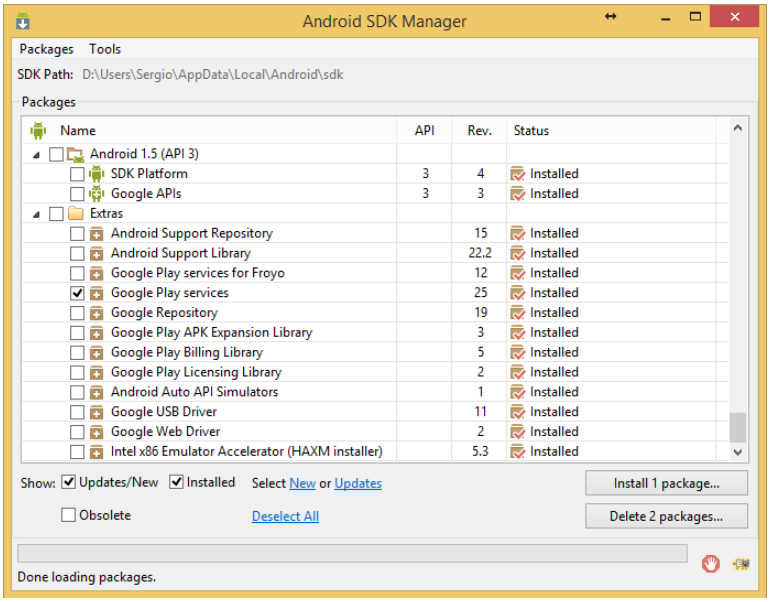


Ilustración 62: Descarga Google Play Services

Una vez descargado el paquete de servicios para Google Play, se encontrará en la carpeta del SDK de Android. Hay que importar el paquete al espacio de trabajo. Esto se hará siguiendo estos pasos:

- Abrir el archivo build.gradle dentro de su directorio de módulos de la aplicación.
- Añadir una nueva regla de generación en Dependencias de la última versión del "play-services":

```
apply plugin: 'com.android.application'

…

dependencies {

   compile 'com.google.android.gms:play-services:7.5.0'

}
```

- Guardar los cambios y haga clic en "Sincronización de Proyecto con archivos Gradle" en la barra de herramientas.
- Editar el archivo AndroidManifest.xml de la aplicación agregando la siguiente declaración en el elemento <application>. Este incorpora la versión de los servicios de Google Play con la que fue compilada la aplicación.

Obtener una clave de API

Para acceder a los servidores de Google Maps con la API de Google Maps, se debe agregar una clave de API de Google Maps a la aplicación. La clave es libre, se puede utilizar con cualquiera de las aplicaciones que llaman a la API de Google Maps , y es compatible con un número ilimitado de usuarios. Se puede obtener una clave de API de Google Maps desde la consola de desarrolladores de Google, proporcionando certificado de firma de su aplicación y su nombre del paquete.

Entender el proceso de registro de su solicitud y la obtención de una clave requiere algún conocimiento de proceso y los requisitos de publicación de Android. En resumen, todas las aplicaciones de Android deben estar firmadas con un certificado digital para el que se mantiene la clave privada. Debido a que los certificados digitales son únicas, proporcionan una forma sencilla de identificar de forma única su aplicación. Esto hace que sean útiles para el seguimiento de su aplicación en sistemas como Google Play Store, y para el seguimiento de uso de la aplicación de los recursos, como los servidores de Google Maps.

Las claves de la API están vinculadas a pares específicos certificado / paquete, en lugar de a los usuarios o aplicaciones. Sólo se necesita una clave para cada certificado, no importa cuántos usuarios tiene para una aplicación.

La obtención de una clave para su aplicación requiere varios pasos:

- Recuperar información sobre el certificado de su aplicación: La clave API de Google Maps se basa en una forma corta de certificado digital de su aplicación, conocida como SHA- 1. La huella digital es una cadena de texto único generado por la comúnmente usada SHA- 1 algoritmo hash. Debido a que la huella digital es en sí mismo único, Google Maps utiliza como una forma de identificar a la aplicación. Se puede mostrar de un certificado SHA- 1 utilizando el programa keytool con el parámetro -v.
- Ir a la consola de desarrolladores Google y crear un nuevo proyecto. En la barra lateral de la izquierda, expanda APIs y autenticación. A continuación, hacer clic en las API, activar la API de la lista de APIs, y por último, seleccionar el botón Activar API para la API.
- En la barra lateral de la izquierda, seleccionar Credenciales y crear una clave de API seleccionando “Crear nueva clave” y seleccionando “clave Android”.
- En el cuadro de diálogo resultante, introduzca el SHA- 1 de la aplicación, a continuación, un punto y coma (;), y a continuación, el nombre del paquete de su aplicación.
- La Consola de Desarrolladores Google mostrará una clave titulada “Key for Android applications“ seguidas de una clave de API de cuarenta caracteres.

Añada la clave API para su aplicación

En AndroidManifest.xml, hay que agragar el siguiente elemento como un hijo del elemento <application> , insertándolo justo antes de la etiqueta de cierre < / application> :

```
<meta-data
android:name="com.google.android.geo.API_KEY"
android:value="API_KEY"/>
```

Se ha de sustituir su clave API para API_KEY en el atributo value. Este elemento hace que la clave de API visible para cualquier objeto MapFragment en su aplicación.

Añadir un mapa:

Para añadir un mapa a la aplicación, es necesario añadir un fragmento al Layout correspondiente de la forma:

```
<fragment xmlns:android=http://schemas.android.com/apk/res/android
   android:id="@+id/map"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:name="com.google.android.gms.maps.MapFragment"/>
```

Una vez definido el fragmento, hay que referenciarlo en la clase java correspondiente y definir las coordenadas que se quieren marcar:

```
MapFragment mapFragment = (MapFragment) getFragmentManager()
      .findFragmentById(R.id.map);
mapFragment.getMapAsync(this);
public void onMapReady(GoogleMap map) {
   LatLng oficina = new LatLng(39.263596, -0.469858);
   map.moveCamera(CameraUpdateFactory.newLatLngZoom(oficina, 17));
   map.addMarker(new MarkerOptions()
         .title("Oficinas Centrales")
         .position(oficina));
}
```

7. CONCLUSIONES Y TRABAJOS FUTUROS

Se ha desarrollado una aplicación Android que cumple con las funciones especificadas. Con lo desarrollado en el ámbito del presente trabajo, tan sólo sería necesario el dotar al sistema de los ajustes de seguridad pertinentes para poder incluso publicar la aplicación.

La aplicación cumple con las funciones especificadas, completándose todos los requerimientos que se habían establecido tras las reuniones iniciales con el personal de la empresa comercializadora involucrada.

Actualmente la aplicación ha sido presentada al personal de la empresa comercializadora con el que se ha elaborado. Esta presentación ha tenido una gran aceptación por parte de este personal, tanto es así, que se presentará en un futuro próximo a la junta directiva de la misma con el fin de evaluar su integración dentro de los servicios de la empresa.

El interés suscitado como consecuencia de la aplicación desarrollada, ha llevado además a plantear futuras versiones con nuevas funcionalidades que completen la versión actual. Entre estas nuevas funcionalidades que conformarían el trabajo futuro a desarrollar cabe destacar las siguientes:

Ampliar la aplicación para todas las tarifas. Actualmente, la aplicación diseñada sirve para usuarios con contadores de baja tensión y con potencias menores que 450 KW, que incluye a las tarifas 2.0A, 2.0DHA, 2.1A, 2.1DHA y 3.1DHA. Tal y como se planteó inicialmente, las tarifas 3.1A y 6.1 no están incluidas en la aplicación. La tarifa 3.1A es obligatoria en alta tensión y potencias menores a 450 KW, mientras que la tarifa 6.1 es obligatoria en alta tensión y potencias mayores a 450 KW e incluye una discriminación horaria séxtuple dependiendo de la hora, el dio y el mes, por lo que es mucho más compleja que las otras tarifas.

Otra funcionalidad que sería interesante incluir en futuras versiones, sería la posibilidad de incluir un vínculo en cada contador que llevara al usuario a un mapa de Google en el que se mostrara la ubicación de dicho contador. De esta manera, si un usuario posee varios contadores, podría ver fácilmente cual es el que está seleccionando.

Una funcionalidad que se podría contemplar sería que los contenidos de la ayuda de la aplicación fueran dinámicos, de forma que se pudieran actualizar periódicamente con nuevas FAQ y/o contenidos que la empresa considere interesantes.

La última funcionalidad que se está estudiando incluir para futuras versiones es el añadir opciones operativas, y no solo de consulta, de forma que se permita a los usuarios operaciones como el cambio de tarifa o de potencia.

Dentro del posible trabajo futuro, se está planteando la posibilidad de portar la aplicación al sistema operativo IOS, instalado en los dispositivos Apple como pueden ser iPhone o iPad. Para ella se deberían portar los desarrollos realizados utilizando el lenguaje de programación Swift, que es el lenguaje de programación con el que se desarrollan aplicaciones para IOS.

8. BIBLIOGRAFÍA

https://persefoneblog.wordpress.com/formacion/android_introduccion/tema-1/1-1-arquitectura-de-android/

http://academiaandroid.com/activity-y-fragments/

http://www.androidcurso.com/index.php/tutoriales-android/37-unidad-6-multimedia-y-ciclo-de-vida/158-ciclo-de-vida-de-una-aplicacion

https://columna80.wordpress.com/2011/09/17/diseando-interfaces-en-android-xml-y-propiedades/

http://androideity.com/2011/07/06/ciclo-de-vida-de-una-actividad/

http://elbauldeandroid.blogspot.com.es/2013/12/fragments.html

http://www.androidcurso.com/index.php/tutoriales-android/36-unidad-5-entradas-en-android-teclado-pantalla-tactil-y-sensores/144-manejo-de-eventos-de-usuario

http://android-dcn.blogspot.com.es/

http://www.androidcurso.com/index.php/tutoriales-android/32-unidad-2-diseno-de-la-interfaz-de-usuario-vistas-y-layouts/114-layouts

https://emiliosedanogijon.wordpress.com/2014/10/16/diseno-de-la-interfaz-vistas-y-layouts/

https://prezi.com/ktmdyego0_qc/algoritmo-de-enrutamiento-camino-mas-corto/

http://jonsegador.com/2012/09/diferentes-unidades-de-medida-disponibles-en-android-dp-sp-pt-px-mm-in/

http://www.compascreativo.com/2013/08/28/disenando-para-android-que-es-la-medida-dp-breve-apunte/

http://celutron.blogspot.com.es/2007/12/ciclo-de-vida-de-una-aplicacin-android.html

http://jdeveloper.wikispaces.com/15.5.+Programaci%C3%B3n+en+Android++Interfaz+de+usuario

http://todosobreprogramacion.blogspot.com.es/

http://www.androidcurso.com/index.php/tutoriales-android/32-unidad-2-diseno-de-la-interfaz-de-usuario-vistas-y-layouts/114-layouts

http://www.alegsa.com.ar/Diccionario/C/4133.php

http://www.istas.net/web/index.asp?idpagina=2207

http://www.tutorialeshtml5.com/2012/05/android-thread-hilo-y-handler-proceso.html

http://academiaandroid.com/ejecucion-tareas-segundo-plano-android/

http://www.androidcurso.com/index.php/tutoriales-android/36-unidad-5-entradas-en-android-teclado-pantalla-tactil-y-sensores/271-hilos-de-ejecucion-en-android

http://www.limecreativelabs.com/asynctask-tareas-asincronas-en-android/

http://www.sgoliver.net/blog/tareas-en-segundo-plano-en-android-i-thread-y-asynctask/

http://www.androidcurso.com/index.php/tutoriales-android-fundamentos/41-unidad-7-seguridad-y-posicionamiento/223-google-maps-api-v2

https://developers.google.com/maps/documentation/android/start?hl=es

http://developer.android.com/

4-67 1296 N Biblioteca de informática

Desarrollo de aplicaciones seguras

Miguel Ángel Moreno

Informática 64

4-67 1369 N Biblioteca de informática

Android, Desarrollo de aplicaciones ganadoras

Wei-Meng Lee

Anaya Multimedia

4-67 725 Biblioteca de informática

El gran libro de Android

Jesús Tomás Gironés

Marcombo ediciones técnicas

MANUAL DE USUARIO

1. Requisitos

Los requisitos mínimos para la instalación de la aplicación son los siguientes:

- Dispositivo Android con una versión mínima de 3.0 (API 11). **

- Conexión a red 3g o Wi-Fi, necesaria tanto para la descarga de la aplicación como para la utilización de esta, ya que esta necesita poder acceder a los datos del servidor.

- Para poder mostrar el mapa de google en la aplicación, es necesario que el dispositivo posea al menos la API gráfica OpenGL ES versión 2.

** Para saber que versión se tiene instalado, se han de seguir los pasos indicados en el anexo del manual.

2. Instalación

Para la instalación de la aplicación, primero se debe acceder a la Play Store, que viene preinstalado en todos los dispositivos Android por defecto.

Una vez dentro de la Play Store, hay que acceder a la sección de búsqueda y teclear “TFG Sergio Alemany” y seleccionar la aplicación que aparece. Una vez dentro de la descripción de la app, hay que pulsar el botón instalar y aceptar los permisos que necesitará. La descripción de los permisos necesarios se encuentra en el siguiente punto.

En este momento, la descarga de la aplicación mediante la Play Store solamente está disponible en modo de prueba y tan solo para los dispositivos que hayan sido aceptados, por lo que la instalación deberá realizarse a través del archivo APK adjuntado en el trabajo. Para ello, es necesario copiar el archivo al dispositivo Android y, mediante un explorador de archivos, acceder al archivo copiado y hacer clic en él para acceder a la instalación.

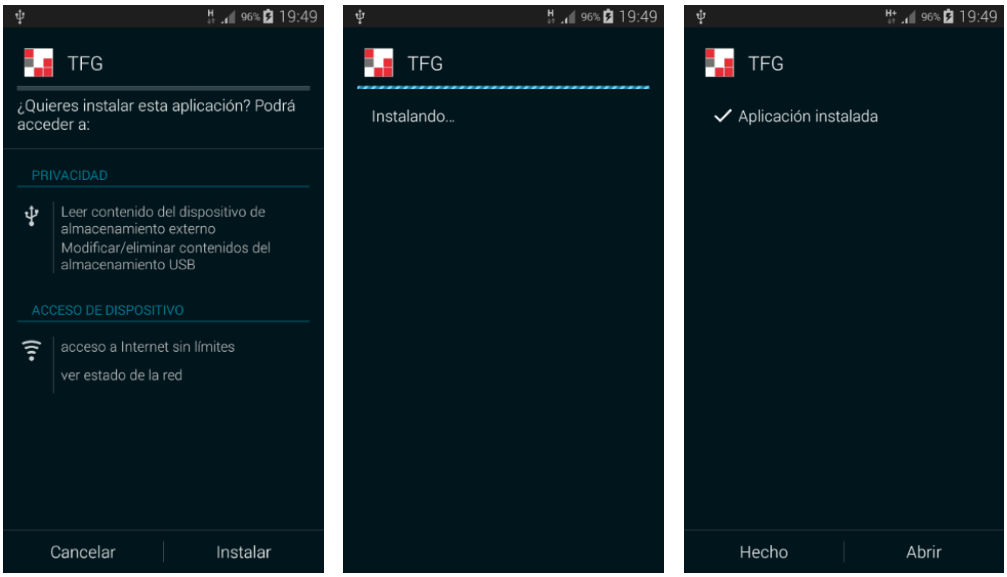
También se puede descargar el archivo APK a través del siguiente enlace, en caso de no tener acceso a los archivos adjuntados en el trabajo:

https://www.dropbox.com/s/703isgu3trof1wh/app-release.apk?dl=0

3. Permisos necesarios

Almacenamiento de datos: Necesario para poder guardar las preferencias de usuario, así como para poder mantener el nombre de usuario en la pantalla de identificación, si se desea.

Acceso a Internet: Necesario para poder conectarse al servicio y acceder a los datos necesarios para el funcionamiento de la aplicación.



Figuras 63/64/65: Instalación aplicación

4. Acceso e identificación.

Para acceder a la app tan solo hay que pulsar el icono correspondiente en el menú de Android. Esta le llevará a la pantalla de identificación de la aplicación.

Para poder entrar en la aplicación son necesarios dos requisitos previos por parte del usuario:

- Disponer de un contrato con la empresa comercializadora eléctrica para la cual esté implementada la aplicación.
- Disponer un usuario y una contraseña con las que poder identificarse en la aplicación.

En caso de cumplirse solamente el primer requisito, el usuario deberá ponerse en contacto con la distribuidora eléctrica correspondiente para obtener el usuario y la contraseña necesaria.

Una vez cumpla los dos requisitos, simplemente hay que introducir el usuario y la contraseña en los campos indicados y pulsar el botón ENTRAR.

Si la identificación ha sido satisfactoria, la aplicación pasará a la pantalla de elección de contador, donde el usuario podrá decidir sobre que contador en su propiedad desea realizar las consultas.

En caso de fallo en la identificación, ya sea por falta de conexión a Internet o porque los datos introducidos no son correctos, se mostrará un mensaje indicando que la identificación no se ha realizado con éxito.

Para motivos de evaluación, se proporciona cuenta auxiliar que permite a los evaluadores probar la aplicación. Tanto el usuario como contraseña de esta cuenta tienen como nombre “invitado” y la contraseña se restablecerá cada día en caso de que sea cambiada.

5. Elección de punto de suministro / contador

En la pantalla de elección de contador, se mostrará una lista de todos los contadores que el usuario posea, permitiéndole elegir sobre que contador desea hacer las consultas deseadas. Los contadores están indicados tanto por su número de serie, como por la dirección física en la que están contratados. Para seleccionar un contador, tan solo hay que pulsar en el botón correspondiente al contador y la aplicación pasará al menú principal del contador.

6. Detalles de tarifa

Si pulsa el botón DETALLES DE TARIFA dentro del menú principal, accederá a los datos tanto de potencia como de tarifa contratada, así como una lista de las tarifas que el usuario puede contratar con la potencia actual.

En caso de que la potencia contratada sea mayor que 15 KW, será posible acceder a la pantalla de "análisis de potencia" pulsando el botón de la derecha. En esta pantalla, se mostrará la potencia actual contratada y la potencia máxima utilizada, así como una breve recomendación dependiendo del resultado.

En caso de que la potencia contratada sea menos de 15 KW, será posible acceder a la pantalla de "análisis de tarifas" pulsando el botón de la derecha. En esta pantalla, aparecerá tanto la tarifa contratada, como la tarifa recomendada por el sistema. Esta tarifa recomendada será fruto de las simulaciones de las últimas 6 facturas con las diferentes tarifas y le indicará al usuario cual es la tarifa que más se adapta a sus necesidades.

En caso de que la tarifa recomendada no sea la misma que la que tiene contratada, será posible acceder a la pantalla de "análisis de tarifa recomendada" donde se mostrará al usuario la comparación realizada mes a mes desde los últimos 6 meses.

Si simplemente se desea comprobar cuáles serían los resultados con otras tarifas aunque no sean las óptimas calculadas por el sistema, habrá que pulsar el botón COMPARADOR DE TARIFAS situado dentro de la pantalla "análisis de tarifa" para acceder a la pantalla en la que se podrá comparar las tarifas que se desee en los meses que se desee.

7. Ver factura

Para poder acceder a las facturas antiguas emitidas, hay que pulsar en el botón FACTURAS, dentro del menú principal del contador. En la pantalla que aparece, se da la oportunidad de seleccionar el año y el mes sobre el que se quiere hacer la factura.

Una vez seleccionada la factura deseada, simplemente hay que pulsar el botón VER FACTURA que abrirá una pantalla con todos los detalles de la factura elegida.

En ella, se muestran desglosados los términos de potencia, energía y alquiler de contador:

- En el apartado "energía" se muestra la energía consumida en cada periodo (si existe discriminación horaria), el precio de cada kilovatio y el importe total por periodos.

- En el apartado “potencia” se muestra la potencia contratada y, en caso de que esta sea mayor de 15, la potencia facturada, que puede ser mayor o menor a la contratada, teniendo en cuenta el maxímetro para las tarifas de alta potencia.
- En el apartado “alquiler” se mostrará si dicho contador está alquilado o no y, en caso de estarlo, se mostrará el importe de dicho alquiler.
- Se muestra también el apartado “impuesto”, que representa el llamado ”impuesto de la energía”, que se aplica tanto a la energía consumida como la potencia.
- Finalmente, el último apartado incluye el subtotal, que es la suma de todos los apartados anteriores, el IVA, que muestra el 21% del subtotal, y el total, que es la suma del subtotal y el IVA.

8. Gráficos de consumo

Al pulsar el botón CONSUMOS dentro del menú principal, se accede a una pantalla que nos muestra una gráfica que representa la energía consumida por el usuario a lo largo de un periodo de tiempo. Este periodo será el del último mes por defecto. Si se desea cambiar el periodo mostrado, tan solo hay que seleccionar las fechas de inicio y de fin mediante las herramientas de selección de fechas situados debajo de la gráfica.

Esta gráfica tan solo mostrará el consumo de energía activa. Si se desea obtener unos datos más precisos, hay que pulsar la gráfica, y se mostrará una pantalla con la energía total activa y la energía total reactiva. Si se tiene contratado una tarifa con discriminación horaria, aparecerá también la opción al lado de cada dato, de acceder a otra pantalla con los datos de la energía consumida por periodo de discriminación horaria.

Así, por ejemplo, el contador que tenga contratado una discriminación horaria doble, obtendrá los datos de la energía consumida en punta y en valle, mientras que el contador que tenga contratado una discriminación horaria triple, obtendrá los datos pertenecientes a punta, valle y llano.

También se encontrará la opción de ver los horarios en los que se divide la discriminación horaria para mayor información para el usuario, pulsando el botón VER HORARIOS DE DISCRIMINACIÓN HORARIA.

Para obtener una tabla con todos los consumos exactos dispuestos por días u horas, hay que pulsar en el botón TABLA DE CONSUMOS situado en la primera pantalla de datos de la gráfica.

9. Evolución consumo mensual y consejos

Pulsando el botón EVOLUCIÓN CONSUMO MENSUAL en el menú principal, se accede a una pantalla en la que muestra el consumo medio diario en el mismo mes de hace un año, y el consumo medio diario que lleva consumido hasta el momento en el mes en el que se encuentra.

Si el consumo medio es menos al del año pasado se mostrará un mensaje felicitando al usuario por el ahorro energético. En caso contrario, se mostrará un mensaje de advertencia para que el usuario sepa que está consumiendo más energía que el año anterior.

En cualquier caso, esta pantalla mostrará un botón el cual mostrará otra pantalla en la que se podrán ver algunos consejos con lo que ahorrar electricidad. Pulsando en cualquiera de esos consejos, se mostrará un texto detallando cada consejo dado para una mayor información del usuario.

10. Opciones

Al pulsar el botón OPCIONES del menú principal se accederá a la pantalla de opciones. En esta pantalla tan solo hay una sola opción aunque se ha dejado espacio para añadir futuras opciones.

La única opción disponible es la de cambiar contraseña, al pulsar el botón correspondiente, el usuario accederá a la pantalla de cambio de contraseña, en la que se le da la opción al usuario de cambiar su contraseña.

Para ello, el usuario deberá introducir la contraseña que tiene en el momento para evitar un posible mal uso de terceros. A continuación, se deberá insertar la nueva contraseña en el apartado correspondiente y pulsar el botón CAMBIAR CONTRASEÑA.

La contraseña quedará cambiada si y solo si se ha introducido correctamente la anterior contraseña.

11. Contacto

Si se pulsa el botón CONTACTO en el menú principal, se abrirá la pantalla de contacto, en la cual hay diversos apartados:

- En el apartado ENVIAR MENSAJE SOPORTE, el usuario puede enviar un mensaje al soporte de la aplicación, indicando si tiene algún tipo de problema con la aplicación.
- En el apartado COMO ENCONTRARNOS, se encuentran las diferentes opciones que tiene un usuario para contactar con la empresa: Teniendo la dirección física puede acudir a las oficinas centrales; pulsando el botón de la derecha se accede a un mapa de Google donde está señalizado exactamente su localización. Se aporta también el teléfono y el correo electrónico de contacto. Pulsando sobre cada uno de ellos se accede al teléfono con el número marcado o a la aplicación de correo electrónico según corresponda.
- Por último, en el apartado AYUDA (F.A.Q.) se encuentra una lista con las preguntas más frecuentes que puedan llegar a surgir, así como sus respuestas.

12. Botón atrás

En todas las pantallas de la aplicación, hay un botón ATRÁS situado en la parte inferior de la pantalla. Dicho botón permite volver a la última pantalla en la que estaba.

Si se pulsa el botón cuando se encuentra en el menú principal, se volverá a la pantalla de selección de contador, pero no cerrará la sesión del usuario.

13. Salir de la aplicación

Para salir de la aplicación y volver a la pantalla de identificación, se deberá pulsar el botón SALIR situado en el menú principal.

Para volver a la pantalla principal de Android puede pulsar el botón atrás del dispositivo Android mientras se encuentra en la pantalla de identificación. También puede pulsar el botón inicio del dispositivo Android en cualquier pantalla de la aplicación, aunque hay que tener en cuenta que esta acción no cerrará la sesión hasta que se termine el proceso o hasta que vuelva a entrar en la aplicación y pulse el botón SALIR del menú principal.

14. Desinstalar

Para desinstalar la aplicación existen varios métodos:

- Desinstalar la aplicación mediante el menú de ajustes del dispositivo: Ve al menú del dispositivo y accede a Ajustes > Aplicaciones o Administrador de aplicaciones (según el dispositivo). Busca la aplicación y pulsa el botón desinstalar.
- Desinstalar la aplicación mediante la aplicación Google Play Store: Abre la aplicación de Google Play Store, accede a menú > Mis aplicaciones, toca cualquier aplicación etiquetada como "Instalada" y selecciona Desinstalar.

ANEXO: Averiguar versión instalada en el dispositivo

Para averiguar la versión de Android instalada en el dispositivo, hay que acceder a Ajustes > Acerca del dispositivo y buscar el apartado “Versión de Android. Si el número que aparece es 3.0 o mayor, el dispositivo es compatible con la aplicación.

PRESUPUESTO

1. Introducción al presupuesto

A continuación se presenta el presupuesto detallado de este proyecto. Es necesario tener en cuenta algunas consideraciones que se han tenido en cuenta durante su ejecución.

El presupuesto consta de 4 capítulos nombrados a continuación:

- Capítulo 1: Licencias de Software.
- Capítulo 2: Equipo informático.
- Capítulo 3: Servidor informático
- Capítulo 4: Mano de obra

2. Capítulo 1: Licencia de software

Para llevar a cabo el trabajo se han utilizado diferentes programas informáticos. El principal programa informático utilizado ha sido el Android Studio, con el que se ha realizado toda la programación de la aplicación. Se ha utilizado también la arquitectura WampServer con Apache para crear la base de datos y phpMyAdmin para gestionar esa misma base de datos.

Software	Unidades	Precio Unitario (€/ud)	Coste
Android Studio 1.2.1	1	Licencia gratuita	0,00€
Android SDK Manager	1	Licencia gratuita	0,00€
Apache 2.4.9	1	Licencia gratuita	0,00€
MySQL 5.6.17	1	Licencia gratuita	0,00€
phpMyAdmin 4.1.14	1	Licencia gratuita	0,00€
		Total	0,00€

Tabla 1: Licencia de software

3. Capítulo 2: Equipo informático

El ordenador utilizado tendrá las siguientes características:

- Sistema operativo Windows 8.1 Pro 64-bit
- Procesador Intel Core i5 4670K 3.40GHz
- Memoria RAM 8,00GB DDR3 1199MHz
- Placa base ASUSTeK COMPUTER INC. Z97-K (SOCKET 1150)
- Gráfica 2047MB NVIDIA GeForce GTX 660 (Gigabyte)
- Pantalla BenQ GL2250H
- Almacenamiento 119GB SanDisk SDSSDHP128G

Se considerará un periodo de amortización de 3 años, con una utilización media de 8 horas diarias durante 240 días al año (1920 horas al año).

Equipo	Unidades	Horas/año	Horas de trabajo	Precio unitario (€/ud)	Amortización	Coste (€)
PC	1	1920	300	1.000	5,21%	52,08€
					Total	52,08€

Tabla 2: Equipo informático

4. Capítulo 3: Servidor informático

A pesar de que para el desarrollo del trabajo se ha utilizado un ordenador personal para almacenar la base de datos, es necesario un servidor para almacenar la base de datos. Es importante tener en cuenta que el tamaño de la muestra prestada para el desarrollo del trabajo no en comparable al número de datos que posee la empresa colaboradora. Así pues, se supondrá la compra de un servidor dedicado para esta tarea o, al menos, el coste de ampliación de dicho servidor. La empresa colaboradora ya dispone de un servidor dedicado por lo que solo sería necesario ampliarlo y no comprar uno nuevo:

Equipo	Unidades	Precio unitario (€/ud)	Coste
Ampliación de servidor	1	12.000	12.000,00€
		Total	12.000,00€

Tabla 3: Servidor informático

5. Capítulo 4: Mano de obra

Tarea	Horas	Precio unitario	Coste
Diseño de pantallas	10h	30€	300,00€
Creación de la aplicación con Android Studio	240h	30€	7.200,00€
Creación y configuración de base de datos	20h	30€	600,00€
Creación de archivos PHP necesarios	30h	30€	900,00€
		Total	9.000,00€

Tabla 4: Mano de obra

6. Coste total

Para finalizar el presupuesto, se sumarán todas las partidas y se aplicará el IVA correspondiente además del 6% de beneficio industrial.

Partida	Coste (€)
Licencia de software	0,00€
Equipo informático	52.08€
Servidor informático	12.000,00€
Mano de obra	9.000,00€
Subtotal	21.052,08€
Beneficio industrial (6%)	1.263,12€
IVA (21%)	4.421,08€
TOTAL	26.736,28€

Tabla 5: Coste total