MEMORIA
ANDROID - OpenDomo

Autor: José Manuel Recarey Quintáns
Consultor: Gregorio Robles Martínez
Tutor externo: Oriol Palenzuela
Empresa asociada: OpenDomo Services S.L
Fecha: 9/06/2013

INDICE

LICENCIA

MEMORIA.

Autor: José Manuel Recarey Quintáns

I. INTRODUCCIÓN/MOTIVACIÓN

Como introducción a la memoria de desarrollo de la aplicación, se empezará diciendo que todo lo aquí expuesto trata de explicar cómo desarrollar y poner en funcionamiento una aplicación para un dispositivo Android que se conectará a una máquina con un sistema OpenDomo para poder iniciar sesión en ella y pasarle ciertos parámetros (como pueden ser las coordenadas GPS).

Con respecto a este primer apartado, decir que ha habido varios factores que han influido en la elección de este proyecto. Entre los más importantes se podrían destacar:

- El desarrollo de una aplicación para Android. Además de resultar interesante el desarrollar una aplicación para uno de los sistemas con más demanda del mercado y con una creciente demanda (ya que cada vez se extiende más sobre todo en smartphones), juega también un papel importante los conocimentos que se adquieren sobre este sistema.
- La introducción a la domótica que supone el uso de OpenDomo. Al igual que en el caso de Android, la domótica está en plena evolución y expansión, y resulta un tema interesante.
- El uso del software libre. Realmente este factor ha sido determinante en la elección del máster, no siendo tanto en la elección del proyecto. Se presupone que dado que el usuario tiene cada día más cerca estos sistemas, y que una buena parte son gratuitos (cosa importante sobre todo en tiempos de crisis), crecerá la demanda de éstos.

Por otro lado aunque hay muchas formas de explicar el desarrollo de la aplicación, se procederá en primer lugar a explicar las tecnologías utilizadas, luego la implantación o configuración de éstas, y por último se explicará el desarrollo del programa. Teniendo en cuenta lo anterior, y para hacer una introducción a los diferentes apartados de este documento, se muestra a continuación un breve resumen de su estructura:

- **Tecnologías utilizadas:** Este punto muestra de manera muy general las tecnologías que se van a necesitar para comenzar con el desarrollo de la aplicación.
- **Preparación del entorno de desarrollo:** En este apartado explicará que pasos hay que seguir para preparar en entorno básico necesario para el desarrollo de la aplicación.
- **Desarrollo de la aplicación:** Esta es la parte central del trabajo. Es aquí donde se explicará como se desarrolla el código fuente de la aplicación.
- **Pruebas:** De manera resumida, en este punto mencionará como debería ser el desarrollo de las pruebas.
- **Documentación:** Este punto explica cómo se prepara el entorno para desarrollar la documentación. Para realizarla se va a usar Doxygen, y se explica cómo obtener la documentación una vez preparado el código fuente. No se muestran los comentarios añadidos al código, porque éstos pueden verse en los fuentes entregados. Además, esta memoria no intenta hacer un manual para Doxygen.

- **Trabajo futuro:** Este apartado hace un apunte sobre las áreas que a largo plazo pueden sufrir modificaciones (mejoras, nuevas funcionalidades...).
- **Conclusiones y lecciones aprendidas:** Se expondrá de manera breve que ha aportado al programador el llevar a cabo este proyecto.

II.OBJETIVOS

Como se dijo anteriormente, este documento trata de explicar cómo desarrollar una aplicación para Android que permita hacer login en un servidor OpenDomo. Explicándolo un poco más en detalle, el desarrollo puede dividirse en las siguientes tareas:

- **Desarrollo de la pantalla principal de la aplicación.**
  Este punto consistirá en desarrollar una pantalla donde a través de botones se permita guardar parámetros de configuración de usuario o hacer login en el servidor. También habrá un enlace a la página Web de OpenDomo.
- **Desarrollo de la pantalla de configuración de usuario.**
  Esta tarea consistirá en crear un método mediante el cual el usuario pueda guardar datos necesarios para el login en el servidor (usuario y contraseña). También se permitirá guardar ciertos parámetros relativos al Geoposicionamiento, o URLs de conexión.
- **Desarrollo de la pantalla de login.**
  Este apartado será el más complicado. Aquí, a través de dos botones, se permitirá:
  - Hacer login en un servidor Opendomo con los datos guardados mediante la funcionalidad anterior.
  - Envíar peticiones GET al servidor mencionado. Estas peticiones pueden enviarse en el momento en que se cierre la aplicación, o posteriormente de manera cíclica incluyendo las coordenadas GPS.

III. TECNOLOGÍAS UTILIZADAS

En este apartado se numeran las tecnologías utilizadas. Así pues, y dada la naturaleza de la aplicación, se utilizarán las siguientes:

- Android – Se utilizará un sistema Android para la instalación y prueba de la aplicación desarrollada. Así, aunque se podría hacer con otro terminal, seguramente se elija para el uso de esta tecnología un smartphone tipo "Samsung Galaxy S".
  Se usará también Android para el desarrollo de la aplicación, en particular, se necesita un plugin para eclipse y el SDK de Android.

- Sistema Linux – Para la instalación del entorno de programación, así como de de los emuladores y máquinas virtuales necesarias. En referente a este punto, el sistema a usar será una versión de Lubuntu.
  Lubuntu, es un sistema es similar a Ubuntu (que a su vez tiene parecido con una versión Debian de Linux). Tanto Ubuntu como Debian serían unas buenas opciones a tener en cuenta, sin embargo Lubuntu cuenta con ciertas ventajas, entre las que destaca que el entorno gráfico es mucho más ligero. Es por esto, y por que el entorno de Virtualbox y el emulador de Android consumen bastantes recursos, el motivo por el que se elige un sistema que además de libre es muy ligero.

- VirtualBox - Este entorno de virtualización se utilizará para la instalación de la máquina virtual de OpenDomo, para poder probar de manera virtual como se haría la conexión mediante el dispositivo Android.
  Aunque existen otros sistemas similares, se elige VirtualBox por su fácil integración con Lubuntu. Además, la máquina virtual descargada de la Web de OpenDomo (sobre la que se probará el sistema de login y peticiones GET), es soportada por este sistema.

- Wireshark - Este programa ayudará en la fase de pruebas de la aplicación. Al hacer un escaneo de la red, y poder filtrar el tráfico por IP, será muy fácil captar las peticiones enviadas al servidor.
  Aunque existen múltiples maneras de capturar el tráfico de red, algunas incluso desde consola, Wireshark aporta una interfaz muy sencilla de usar. Por otro lado este programa ya ha sido utilizado en otras asignaturas del máster, por lo que se su manejo se facilta todavía más. Es por ello, por su facilidad de uso, el motivo por el que se elije de esta tecnología.

- Doxygen - Ofrece una muy buena alternativa para generar la documentación.
  En el apartado de generación de la documentación existen múltiples opciones, como podrían ser Javadoc, NDoc... Sin embargo, el manejo de Doxygen ya ha sido estudiado en una asignatura del máster. Además a través del plugin Eclox, se integra muy fácilmente con Eclipse.

IV. PREPARACIÓN DEL ENTORNO DE DESARROLLO

Previamente al comienzo del desarrollo del código de la aplicación, hay que tener en cuenta unas cosas de las que ya se ha hablado en el plan de trabajo. Es por ello que nos basaremos en este último para la implantación del entorno de programación y pruebas.

Visto lo anterior, hay que empezar preparando el entorno de desarrollo para la aplicación (se podría entender como una análisis de requisitos), por lo que se podrían seguir los puntos dados a continuación:

1. Para empezar vamos a instalar el entorno virtual que nos ofrece OpenDomo para pruebas ya en su página web. Así, procedemos a entrar mediante un navegador a la dirección web http://es.OpenDomo.org/downloads, donde se podrá descargar la máquina virtual **OpenDomo v.1.0.0.** (Se elegirá el formato OVA, porque así nos será ya directamente compatible con VirtualBox).

2. Como segundo punto hay que instalar el VirtualBox, y ya que estamos usando una versión de Lubuntu se hará desde consola. Aunque no es competencia de esta memoria, se puede decir que usaremos los siguientes comandos en el orden establecido:
    - Se edita el fichero "sources.list" con "sudo gedit /etc/apt/sources.list" para añadir la siguiente línea:
        - deb http://download.virtualbox.org/virtualbox/debian natty contrib
    - Con el comando "wget -q http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -O- | sudo apt-key add -" actualizamos la lista de paquetes a descargar.
    - Finalmente con "sudo apt-get install virtualbox-4.0" instalamos el Virtualbox. Opcionalmente se puede instalar también el paquete dkms.

3. En este punto vamos a añadir la máquina virtual descargada de la página de OpenDomo al VirtualBox. Ésta se dejará con las opciones por defecto, quedando la tarjeta de red virtual en modo puente. En nuestro caso se le asignará la IP 192.168.1.101 a OpenDomo, y la IP 192.168.1.103 al equipo anfitrión del Virtualbox. Es importante que ambas IP estén en el mismo rango para que no tengamos problemas con las conexiones.

   A partir de aquí en principio mediante un navegador web de un equipo, entrando a la dirección IP 192.168.1.101, ya se nos carga la página http://192.168.1.101/cgi-bin/od.cgi/ donde se nos pide un usuario y contraseña para conectarse a la máquina con OpenDomo. Dejaremos de momento esta máquina hasta que lleguemos al punto de Login en ella.

4. Para continuar con la preparación del entorno de desarrollo de la aplicación vamos a instalar las siguientes aplicaciones o paquetes:
    1. ***Java versión 6.*** Se instalará con los comandos:

       sudo add-apt-repository ppa:webupd8team/java
       sudo apt-get update
       sudo apt-get install oracle-jdk6-installer

    2. ***Eclipse.*** Para la instalación de este entorno de desarrollo simplemente hay que

lanzar en consola el comando “sudo apt-get install eclipse eclipse-jdt”.

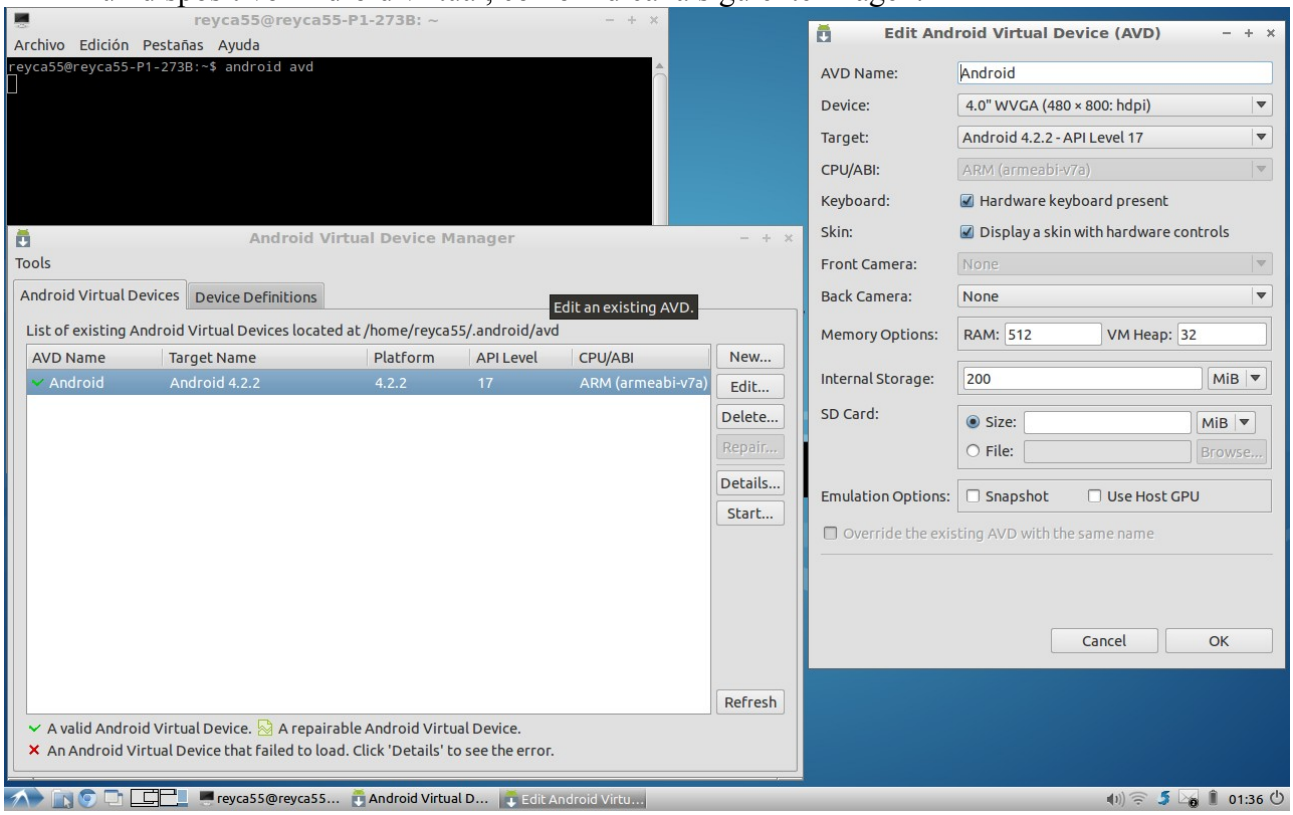
3. ***SDK de Android.*** Primeramente se necesita descargar el fichero “**adt-bundle-linux-x86-20130219.zip**”desde la página web “**http://developer.android.com/sdk/index.html**”.

   Luego se descomprime este paquete en nuestra carpeta de desarrollo dentro de una subcarpeta que se llamará “**android-sdk-linux_86”,** y desde un terminal editamos el fichero **“.bashrc”**de nuestro directorio con **“gedit ~/.bashrc”**, añadiendo la línea “**export PATH=${PATH}:/home/{usuario}/Desarrollo/android-sdk-linux_86/tools**”. (en nuestro caso **{usuario}** se sustituye por “**reyca55**”.

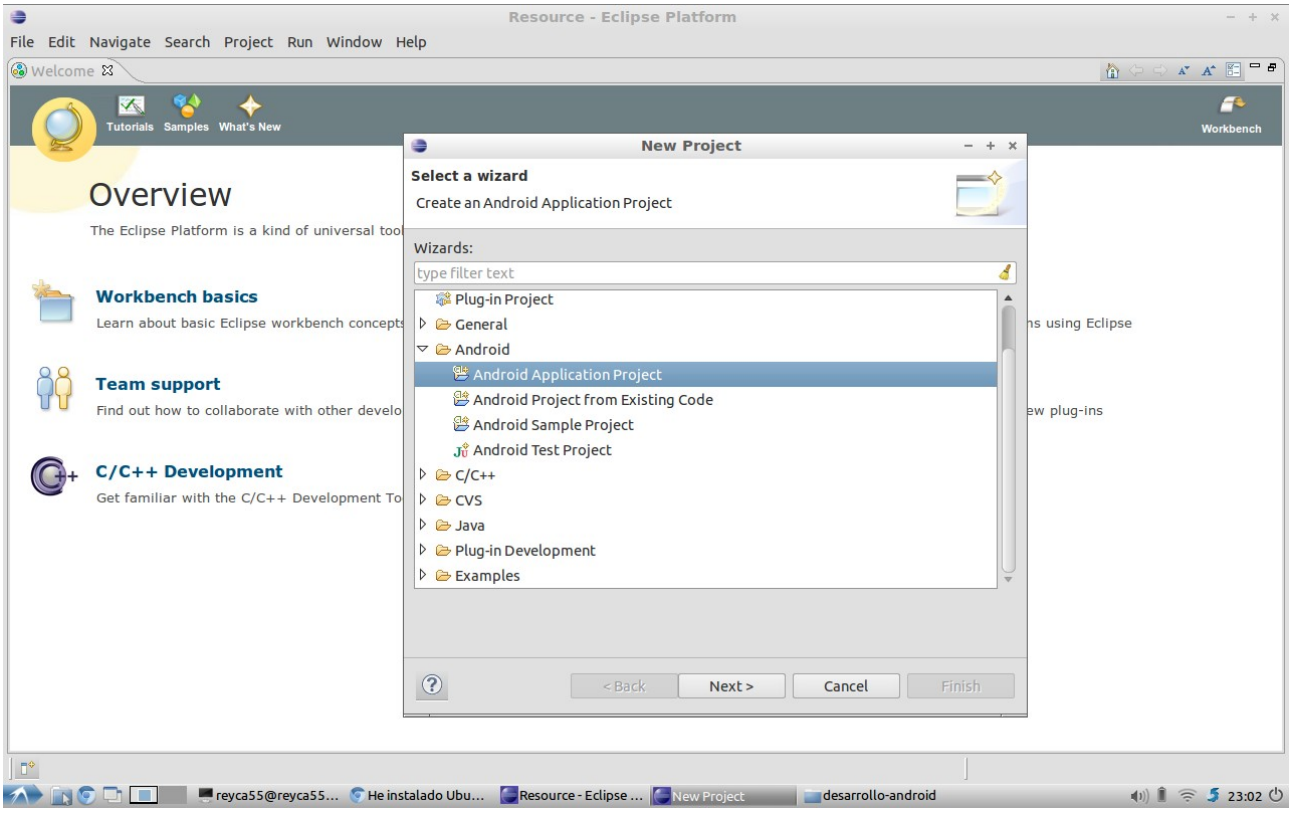
   Cerramos la terminal y la reabrimos para aplicar los cambios con el comando **“android”**.

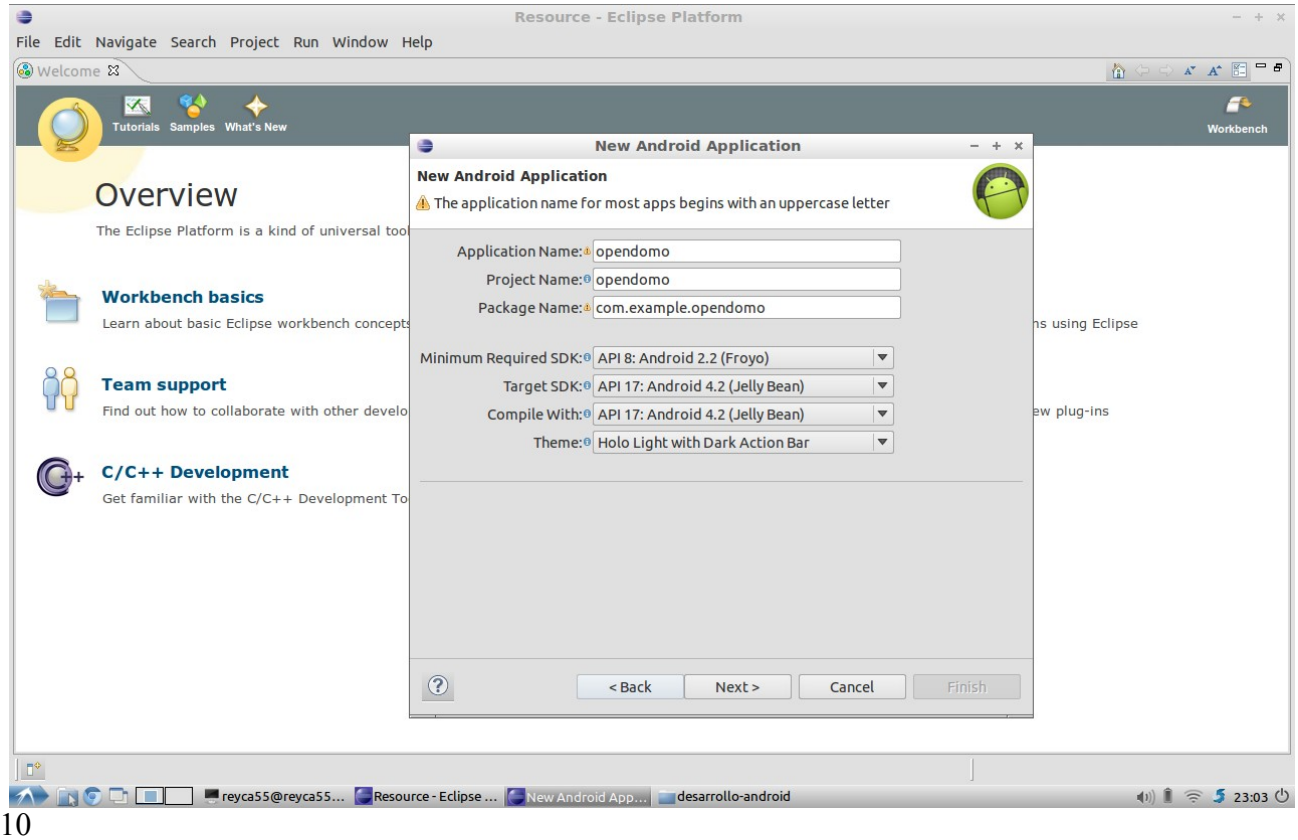
   Cuando se inicie Eclipse, o a través del menú **“Window → Preferences”** en la sección **“Android”**, hay que especificar la ruta del SDK de Android. En nuestro caso será: “/home/reyca55/Desarrollo/android-sdk-linux_86/adt-bundle-linux-x86-20130219/sdk”.

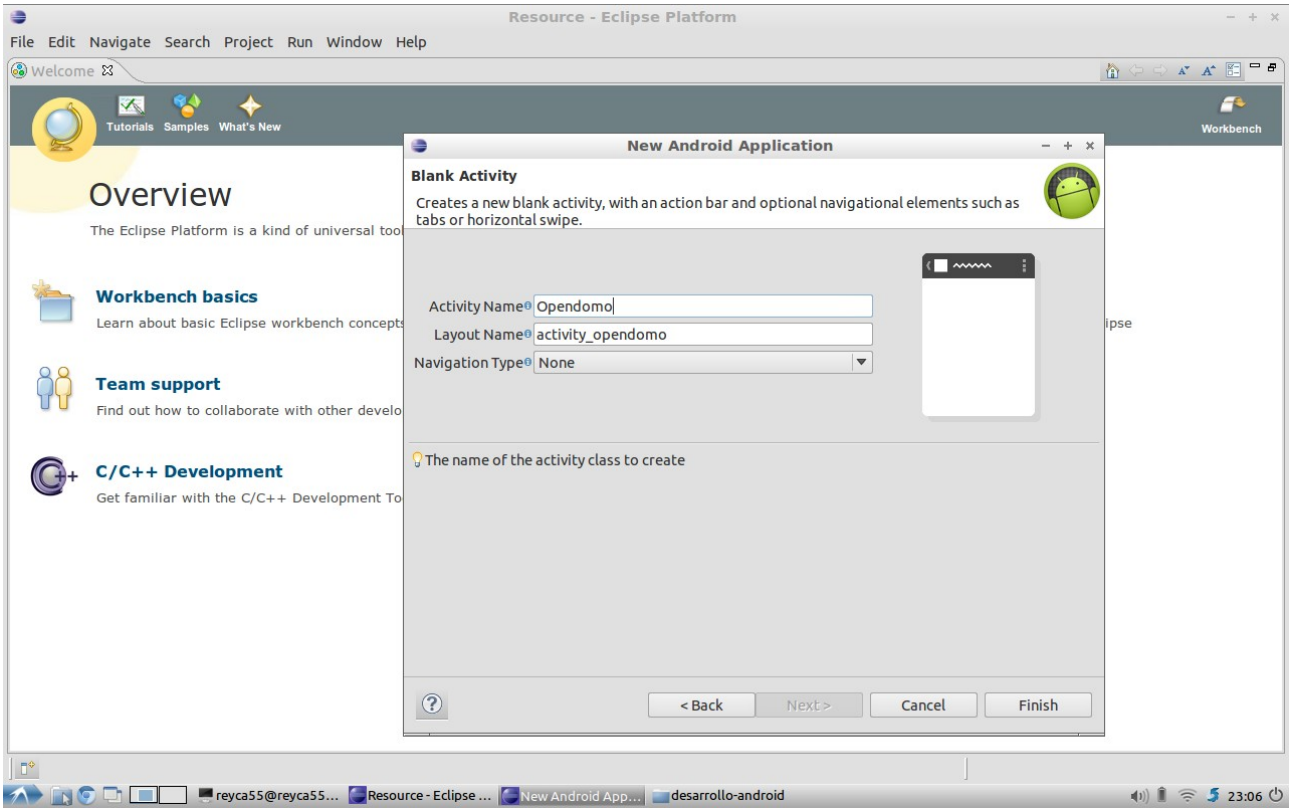
5. Como último punto para poder empezar a crear la aplicación, hay que decir que necesitamos un emulador donde se pueda ir probando la aplicación a medida que la desarrollamos. Para ello, desde la consola, si lanzamos el comando **“android avd”**, se nos abrirá una interfaz llamada “Android Virtual Device Manager” donde a través del botón “New” se puede crear un dispositivo Android virtual, como indica la siguiente imagen:



6. Llegados a este punto, ya podemos empezar a crear la aplicación. Para ello abrimos Eclipse, y a través del menú “File → New → Project”, creamos un proyecto nuevo para Android tal cómo indican las siguientes imágenes:

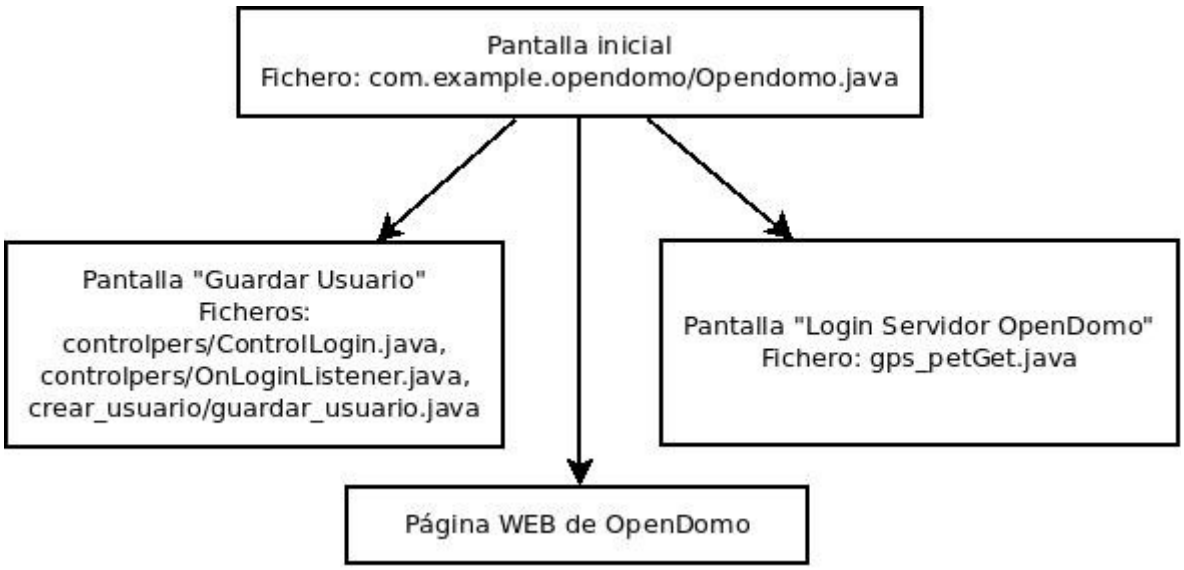





Resource - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Welcome
Tutorials
Samples
What's New
Workbench
Overview
The Eclipse Platform is a kind of universal tool
Workbench basics
Learn about basic Eclipse workbench concepts
Team support
Find out how to collaborate with other develo
C/C++ Development
Get familiar with the C/C++ Development To
New Android Application
Blank Activity
Creates a new blank activity, with an action bar and optional navigational elements such as tabs or horizontal swipe.
Activity Name
Opendomo
Layout Name
activity_opendomo
Navigation Type
None
The name of the activity class to create
< Back
Next >
Cancel
Finish
reyca55@reyca55...
Resource - Eclipse ...
New Android App...
desarrollo-android
23:06

V. DESARROLLO DE LA APLICACIÓN

Previo al desarrollo de la aplicación, y siguiendo los objetivos establecidos en el punto II, se puede establecer el siguiente diagrama de la estructura del proyecto:



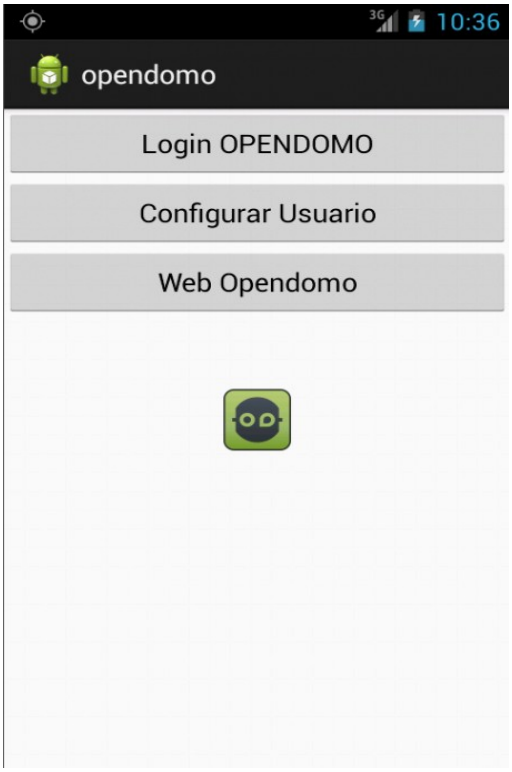
Visto lo anterior, se procede al desarrollo de la aplicación de la siguiente manera:

A.) Desarrollo de la pantalla principal del programa:

Se crea la pantalla de inicio que nos dará a elegir entre diferentes opciones, entre los que destacan:

- **Login OpenDomo:** Esta opción sirve para conseguir logearse en la máquina OpenDomo. También leerá las coordenadas GPS del móvil, para pasárselas a dicha máquina.
- **Configurar Usuario:** A través de este botón, guardaremos un nombre de usuario y una contraseña para hacer el login en la máquina de OpenDomo. Esta opción nos permite que no tengamos que hacer este paso en cada logueo, ya que guarda estas credenciales en un fichero. Guarda además otros datos como la "activación o no" de GPS, su refresco, y la URL del servidor OpenDomo.
- **Web OpenDomo:** Clicleando en este botón nos llevará directamente a la página web de OpenDomo.

Visto lo anterior el aspecto de la pantalla principal podría ser:



Nota: se ha descargado el logo de OpenDomo en formato png, y se ha guardado en la carpeta "/home/reyca55/eclipse/OpenDomo/res/drawable-xxhdpi", para luego añadirlo a la pantalla principal de la aplicación.

Para realizar esta pantalla se han de crear y editar los siguientes ficheros:

- **inicio.xml**, en la carpeta "res → layout", con el siguiente contenido:

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
        android:id="@+id/widget49"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical"
        xmlns:android="http://schemas.android.com/apk/res/android"
        >

    <Button
        android:id="@+id/Login_op"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Login_op"
        android:paddingLeft="20dip"
        android:paddingRight="20dip"  />

    <Button
        android:id="@+id/Conf_usuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Conf_usuario"
        android:paddingLeft="20dip"
        android:paddingRight="20dip" />

    <Button
        android:id="@+id/URL_OpenDomo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/URL_OpenDomo"
        android:paddingLeft="20dip"
        android:paddingRight="20dip" />

    <ImageView android:id="@+id/ImgOD"
        android:contentDescription="@string/Imagen_OD"
        android:scaleType="centerInside"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="42dp"
        android:layout_marginLeft="25dp"
        android:layout_marginRight="25dp"
        android:layout_marginTop="50dp"
        android:src="@drawable/od" />
</TableLayout>
```

En el fichero arriba descrito se puede ver cómo se diseña la pantalla mediante una tabla, añadiendo 3 botones, además de la imagen del logo de OpenDomo que previamente se había guardado. Hecho lo anterior, ahora hay que dotar los botones de funcionalidad. Para ello se creará el siguiente fichero.

- **OpenDomo.java**. Contendrá el siguiente código:

```
package com.example.OpenDomo;

import crear_usuario.guardar_usuario;
import gps_login.gps_petGet;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View.OnClickListener;
import android.view.View;
import android.widget.Button;

public class OpenDomo extends Activity {

        private Button btLog, btUSR, btURL;

        @Override
    protected void onCreate(Bundle savedInstanceState) { //se activará nada más crearse la aplicación
        super.onCreate(savedInstanceState);              //llamada a la clase padre Activity mediante
        setContentView(R.layout.inicio);          //carga de la interfaz inicio

        btLog = (Button) findViewById(R.id.Login_op);      //definición de la funcionalidad del botón de LOGIN
        btLog.setOnClickListener(new OnClickListener()
        {
                        @Override
            public void onClick(View v)
            {
                //A partir de aquí se obtendrán las coordenadas GPS. Luego se hará la conexión a OpenDomo
                 Intent i = new Intent(Opendomo.this,gps_petGet.class);  /*método que abrirá la segunda ventana
al presionar el botón LOGIN*/
                 startActivityForResult(i, 11);
                 finish(); //línea añadida para cerrar la aplicación
            }
        });

        btUSR = (Button) findViewById(R.id.Conf_usuario);  //definición de la funcionalidad del botón de
"CONFIGURACIÓN DE USUARIO"
        btUSR.setOnClickListener(new OnClickListener()
        {
         @Override
         public void onClick(View view)
         {
                 Intent i = new Intent(OpenDomo.this,guardar_usuario.class);  /*método que abrirá la segunda
ventana al presionar el botón CONFIGURAR USUARIO */
                 startActivityForResult(i, 11);
         }
        });

        btURL = (Button) findViewById(R.id.URL_OpenDomo);          //definición de la funcionalidad del botón de
WEB DE OpenDomo
        btURL.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {    //este método abrirá un navegador con la página de OpenDomo cargada
                 Intent browserIntent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.OpenDomo.com"));
                 startActivity(browserIntent);
            }
        });
         }
}
```

Explicar esta clase es muy sencillo. Aquí lo único que se hace es dar funcionalidad a los 3 botones de la interfaz "**inicio.xml**", donde los dos primeros botones nos abrirán una nueva ventana (para la búsqueda de coordenadas GPS y LOGUEO en OpenDomo, o para configurar un usuario), y el tercero nos abrirá la página web de OpenDomo. Fijarse también, que como la aplicación tiene que cerrarse para dejar corriendo algún proceso en segundo plano (peticiones HTTP) cuando se elija la primera opción, se añade la línea "`finish()`", que cierra el entorno gráfico.

Hay que decir también que la clase de la que hablamos (OpenDomo.java) pertenece al paquete “com.example.OpenDomo” creado junto con el proyecto nuevo dentro de la carpeta “src”. Todas las clases colgarán de sus respectivos paquetes, y éstos a su vez de “src”.

- **AndroidManifest.xml.** Este fichero se crea junto con el proyecto en la carpeta “res”. Sin embargo para cada nueva pantalla que creamos, hay que añadir en él unas líneas que definirán sus atributos.
  En este caso en particular sería:

```
<activity
android:name="com.example.OpenDomo.OpenDomo"
android:label="@string/app_name" >
   <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>
```

- Como últimos fichero a modicar o a crear destaca el fichero “res → values → strings.xml”, donde se definirán las cadenás de caracteres presentadas en el programa (se mostrará una captura al final).

B.) Desarrollo de la pantalla de Configuración de Usuario

Para desarrollar la pantalla de Configuración de Usuario, al igual que en el caso anterior, hay que crear varios ficheros entre los que destacan:

- **guardar_usuario.class**, en el paquete "src → crear_usuario".

```
package crear_usuario;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import com.example.OpenDomo.R;
import controlpers.ControlLogin;
import controlpers.OnLoginListener;
import android.os.Bundle;
import android.app.Activity;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.ToggleButton;

public class guardar_usuario extends Activity{
    private ControlLogin ctlLogin;
    private EditText user, passwd, refresc_gps, URL_Serv, URL_PGet;
    private ToggleButton estado_GPS;

        @Override
    protected void onCreate(Bundle savedInstanceState) {
        //con éste método se crea la pantalla de credenciales
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_OpenDomo);

        //se asignan las variables user y passwd a los respectivos campos de la pantalla
        user = (EditText) findViewById(R.id.TxtUsuario);
        passwd = (EditText) findViewById(R.id.TxtPassword);
        estado_GPS = (ToggleButton) findViewById(R.id.on_off_GPS);
        refresc_gps = (EditText)findViewById(R.id.refresco_gps);
        URL_PGet = (EditText)findViewById(R.id.URL_pet_GPS);
        URL_Serv = (EditText)findViewById(R.id.URL_servidor);

        String[] ficheros = fileList();

        //Este IF sirve para cargar los datos previamente introducidos en la pantalla
        if (existe(ficheros, "user_params.txt"))
            try {
                InputStreamReader fichero = new InputStreamReader(openFileInput("user_params.txt"));
                BufferedReader br = new BufferedReader(fichero);
                String linea = br.readLine();
                String usuario_cargado = "";
                String passwd_cargado = "";
                String GPS_activo = "";
                String tiempo_gps = "";
                String URL_Host = "";
               String URL_Get = "";

                while (linea != null) {
                    usuario_cargado = usuario_cargado + linea;
                    linea = br.readLine();
                    passwd_cargado = passwd_cargado + linea;
                    linea = br.readLine();
                    GPS_activo = GPS_activo + linea;
                    linea = br.readLine();
                    tiempo_gps = tiempo_gps + linea;
                    linea = br.readLine();
                    URL_Get = URL_Get + linea;
                    linea = br.readLine();
                    URL_Host = URL_Host + linea;
                    linea = br.readLine();
                }
                br.close();
                fichero.close();
                user.setText(usuario_cargado);
                passwd.setText(passwd_cargado);
                if (GPS_activo.equals("on_gps"))
                 estado_GPS.setChecked(true);
                else
                 estado_GPS.setChecked(false);
                refresc_gps.setText(tiempo_gps);
                URL_PGet.setText(URL_Get);
```

```
                URL_Serv.setText(URL_Host);

            } catch (IOException e) {
            }

        ctlLogin = (ControlLogin)findViewById(R.id.CtlLogin);

        ctlLogin.setOnLoginListener(new OnLoginListener(){
            //con este método se llama a la función correspondiente para guardar las credenciales
            @Override
            public void onLogin(String usuario, String password)
            {
                guardar();
            }
        });
    }

    //función que comprueba si el fichero de credenciales existe
    boolean existe(String[] ficheros, String fichero_busca) {
        for (int f = 0; f < ficheros.length; f++)
            if (fichero_busca.equals(ficheros[f]))
                return true;
        return false;
        }

    //esta función no es necesaria
        /*@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.OpenDomo, menu);
        return true;
    }*/

        //función que guarda "usuario" y "contraseña" en un fichero de texto
        public void guardar() {
        try {
            OutputStreamWriter archivo = new OutputStreamWriter(openFileOutput("user_params.txt",
Activity.MODE_PRIVATE));
            archivo.write(user.getText().toString());      //después de abrir o crear el fichero guarda el
"usuario"
            archivo.write("\n");
            archivo.write(passwd.getText().toString());    //se guarda la contraseña en el fichero
            archivo.write("\n");
            if (estado_GPS.isChecked())
                archivo.write("on_gps");  //se guarda la información de que el GPS está activo
            else
                archivo.write("off_gps"); //se guarda la información de que el GPS está inactivo
            archivo.write("\n");
            archivo.write(refresc_gps.getText().toString());       //se guarda el tiempo de refresco de las
coordenadas GPS
            archivo.write("\n");
            archivo.write(URL_PGet.getText().toString());  //se guarda la URL para las peticiones
            archivo.write("\n");
            archivo.write(URL_Serv.getText().toString());  //se guarda la URL del servidor OpenDomo
            archivo.write("\n");
            archivo.flush();
            archivo.close();
        } catch (IOException e) {
        }
        if (user.getText().toString().equals("") || passwd.getText().toString().equals("") ||
                refresc_gps.getText().toString().equals("") || URL_Serv.getText().toString().equals("") ||
                URL_PGet.getText().toString().equals(""))
        {
        Toast t = Toast.makeText(this, "Faltan datos!!",Toast.LENGTH_SHORT);
        t.show();
        }
        else
        {
        Toast t = Toast.makeText(this, "Los datos fueron grabados",Toast.LENGTH_SHORT);
        t.show();        //se avisa de que los datos se han grabado correctamente
        finish();        //se cierra ventana
        }
    }
}
```

En esta clase existen varios métodos definidos. Se pueden explicar de la siguiente manera:

- **onCreate**: crea la pantalla de credenciales en función del contenido del "fichero activity_OpenDomo.xml" , y carga los datos del fichero de credenciales, si es que éste existe. Una vez pulsado el botón para guardar nuevas credenciales se llama a la función "guardar()", que a parte de guardar la información relativa al usuario, contraseña, estado anterior definido para el botón de activar el GPS y su tiempo de

refresco definido, URL del servidor de OpenDomo y URL para peticiones GET (con geoposicionamiento), también avisará con un mensaje de que el proceso ha finalizado. Por otro lado hay que tener en cuenta que si faltan datos saltará un aviso no dejando continuar con el salvado de credenciales.

– **existe(**String[] ficheros, String fichero_busca**)**: comprueba si existe el fichero de credenciales.

– **guardar**: este método guarda en un fichero la información siguiente, teniendo en cuenta que cada dato va en una línea:

1. "Nombre de usuario"
2. "Su contraseña"
3. Información sobre si el usuario quiere o no el GPS activo.
4. Tiempo de refresco del GPS
5. URL para peticiones GET (con geoposicionamiento).
6. URL del servidor OpenDomo.

– **activity_OpenDomo.xml**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".OpenDomo" >

    <controlpers.ControlLogin
        android:id="@+id/CtlLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

Con este fichero y con el siguiente (controlpers.ControlLogin) conseguimos que el "usuario", "contraseña", "estado de GPS" y las susodichas URLs guardados en el fichero, se muestren en la pantalla que hemos abierto para cambiar las credenciales. Así, por ejemplo, podemos modificar una contraseña sin tener que volver a insertar el usuario.

Además, como todos los ficheros XML, se colgará directamente de "res → layout".

– **ControlLogin.java**: Esta clase se encuentra en "src → controlpers", con el siguiente contenido.

```
package controlpers;

import com.example.OpenDomo.R;
import android.content.Context;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.ToggleButton;

public class ControlLogin extends LinearLayout
{
      private EditText txtPassword, txtUsuario, refresco_gps, URL_Servidor, URL_PetGET;
      private Button btnLogin;
      private ToggleButton estado_GPS;
      private TextView lblMensaje;
      private OnLoginListener listener;

      public ControlLogin(Context context) {
      super(context);
      inicializar();
   }

   public ControlLogin(Context context, AttributeSet attrs) {
```

```
      super(context, attrs);
      inicializar();
   }

   private void inicializar()
   {
       //Utilizamos el layout 'control_login' como interfaz del control
       String infService = Context.LAYOUT_INFLATER_SERVICE;
       LayoutInflater li = (LayoutInflater)getContext().getSystemService(infService);
       li.inflate(R.layout.configurar_usuario, this, true);

       //Obtenemoslas referencias a los distintos controles
       txtUsuario = (EditText)findViewById(R.id.TxtUsuario);
       txtPassword = (EditText)findViewById(R.id.TxtPassword);
       estado_GPS = (ToggleButton)findViewById(R.id.on_off_GPS);
       refresco_gps = (EditText)findViewById(R.id.refresco_gps);
       URL_PetGET = (EditText)findViewById(R.id.URL_pet_GPS);
       URL_Servidor = (EditText)findViewById(R.id.URL_servidor);
       btnLogin = (Button)findViewById(R.id.BtnGuardar);
       lblMensaje = (TextView)findViewById(R.id.LblMensaje);

       //Asociamos los eventos necesarios
       asignarEventos();
   }

   public void setOnLoginListener(OnLoginListener l)
   {
           listener = l;
   }

   private void asignarEventos()
   {
       btnLogin.setOnClickListener(new OnClickListener()
       {
           @Override
           public void onClick(View v) {
               listener.onLogin(txtUsuario.getText().toString(),
                           txtPassword.getText().toString());
           }
       });
   }

   public void setMensaje(String msg) {
           lblMensaje.setText(msg);
       }
}
```

Esta clase tiene varios métodos definidos, pero tal vez el más destacable es “inicializar()”. A través de éste, se pueden obtener datos del usuario (usuario y contraseña a través de la función “asignarEventos()”).

En este mismo paquete (controlpers) se encuentra también una clase necesaria para el funcionamiento de la anterior, llamada “OnLoginListener.java”, y con el siguiente contenido:

```
package controlpers;

public interface OnLoginListener
{
    void onLogin(String usuario, String password);
}
```

- **configurar_usuario.xml**. Con este fichero se realiza el diseño de la pantalla de credenciales de usuario. Este fichero posee el siguiente código, y origina una salida como la que se ve a continuación:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="vertical"
  android:padding="10dip">
    <LinearLayout android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <TextView android:id="@+id/usuario"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/usuario"
            android:textStyle="bold" />

        <EditText android:id="@+id/TxtUsuario"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:inputType="text" />
    </LinearLayout>

<LinearLayout android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="5dip">

    <TextView android:id="@+id/contrasenha"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/contrasenha"
            android:textStyle="bold" />

    <EditText android:id="@+id/TxtPassword"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:inputType="textPassword" />
</LinearLayout>

<LinearLayout android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="15dip">

        <TextView android:id="@+id/estado_GPS"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/estadoGPS"
            android:textStyle="bold" />

        <ToggleButton android:id="@+id/on_off_GPS"
            android:textOn="@string/on"
            android:textOff="@string/off"

            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>

    <LinearLayout android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="10dip">

    <TextView android:id="@+id/tiempo_gps"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/tiempo_gps"
            android:textStyle="bold" />

    <EditText android:id="@+id/refresco_gps"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:inputType="number" />
    </LinearLayout>

    <TextView android:id="@+id/URL_petGPS"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/URL_petGPS"
        android:textStyle="bold"
        android:layout_marginTop="10dip"/>

    <EditText android:id="@+id/URL_pet_GPS"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:inputType="textUri"/>

    <TextView android:id="@+id/URL_server"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/URL_Server"
        android:textStyle="bold"
        android:layout_marginTop="10dip"/>

    <EditText android:id="@+id/URL_servidor"
        android:layout_height="wrap_content"
```

```
            android:layout_width="match_parent"
            android:inputType="textUri" />

    <LinearLayout android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="20dip">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/BtnGuardar"
            android:text="@string/guardar"
            android:paddingLeft="20dip"
            android:paddingRight="20dip" />

        <TextView android:id="@+id/LblMensaje"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="10dip"
            android:textStyle="bold"
            android:textIsSelectable="true"/>

    </LinearLayout>
</LinearLayout>
```

Así, definimos la forma de la pantalla que nos permitirá guardar el usuario y contraseña. Además hay añadido un botón que permitirá iniciar o no el GPS (para poder capturar las coordenadas GPS), además de otros campos de entrada de datos como el "tiempo de refresco de GPS", la "URL para el envío de las peticiones GET de Geoposicionamiento" y la "URL de Opendomo".

Para finalizar este apartado hay unos ficheros a mayores que hay que modificar. En alguno de ellos ya se han realizado algunos cambios para la primera pantalla de la aplicación. Estos ficheros son:

- **strings.xml**. Fichero ubicado en "res → values", sirve para indicar a la aplicación los valores a mostrar por pantalla de algunas variables del programa. Su contenido a esta altura sería:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">OpenDomo</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="usuario">Usuario: </string>
    <string name="contrasenha">Contraseña: </string>
    <string name="login">LOGIN</string>
    <string name="titulo">Configuración Usuario</string>
    <string name="Login_op">Login OpenDomo</string>
    <string name="Conf_usuario">Configurar Usuario</string>
    <string name="URL_OpenDomo">Web OpenDomo</string>
    <string name="guardar">GUARDAR</string>
    <string name="Longitud">Longitud</string>
    <string name="Imagen_OD">OpenDomo 2013</string>
    <string name="conf_usr">CONFIGURACIÓN DE USUARIO</string>
    <string name="estadoGPS">Estado del GPS</string>
    <string name="on">On</string>
    <string name="off">Off</string>
    <string name="URL_petGPS">URL para peticiones GET de Geoposicionamiento</string>
    <string name="URL_Servidor">URL de conexión a OpenDomo</string>
    <string name="tiempo_gps">Tiempo de refresco GPS: </string>

</resources>
```

- **AndroidManifest.xml**. Este fichero ya ha sido descrito anteriormente. Se han hecho algunos cambios para incluir la nueva pantalla (configuración de usuario), y se ha aprovechado para incluir permisos y otras cosas para poder usar el GPS cuando toque leer las coordenadas.

  El contenido del fichero sería:

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

```
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="9"
        android:targetSdkVersion="17" />

         <uses-permission android:name="android.permission.ACCESS_GPS" />
        <uses-permission android:name="android.permission.ACCESS_LOCATION" />
        <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
        <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
        <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name="com.example.OpenDomo.OpenDomo"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="crear_usuario.guardar_usuario"
            android:label="@string/conf_usr">
        </activity>
        <activity
            android:name="gps_login.gps_petGet"
            android:label="@string/login">
        </activity>

    </application>

</manifest>
```

NOTA: fijarse que la línea "`android:minSdkVersion="9"`"establece la versión mínima del SDK de Android. La versión que aparece por defecto es la 8, pero debe cambiarse a la 9 para que funcione la instrucción "`myLocationManager.requestSingleUpdate(LocationManager.GPS_PROVIDER, myLocationListener, Looper.getMainLooper());`".

C.) Desarrollo de la pantalla de Login

El desarrollo de esta funcionalidad ha sido sin lugar a dudas la que más complicación ha tenido. Para empezar y al igual que en las ocasiones anteriores, vamos a ir explicando los ficheros que intervienen en esta funcionalidad, y luego se verá una prueba:

- **gps_petGet.java**: Esta clase se encuentra en "src → gps_login", con el siguiente contenido:

```
package gps_login;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import java.net.URISyntaxException;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Looper;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.URLUtil;
import android.widget.Button;
import android.widget.Toast;
import com.example.opendomo.R;
import crear_usuario.guardar_usuario;

public class gps_petGet extends Activity{

        private Button btSalir;
        private Button btServidor_OD;
        private String string_url;
        private String URL_Host;
        private String URL_login;
        private String URL_Pet_GET;
        private String latitud;
        private String longitud;
        private long retardo_gps;
        private String GPS_activo;
        private String cadena_congps;
        private String tiempo_gps;
        public String usuario_cargado;

        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.gps_login); // Define el esquema de la pantalla final

            string_url = "";
            string_url = login_OD();

            btSalir = (Button) findViewById(R.id.Salir);   //Definición de la funcionalidad del botón de SALIR
            btSalir.setOnClickListener(new OnClickListener()
            {
                        @Override
                public void onClick(View v)
                {
                    //Se vuelven a cargar datos. puede ser necesario
                    string_url = "";
                    string_url = login_OD();
                    Continuar_URL();
                    //Se cerrará la ventana en cuestión y se continúa con el desarrollo en background si la URL
es correcta
                    finish();
                }
            });

            btServidor_OD = (Button) findViewById(R.id.Servidor_Opendomo);  //Definición de la funcionalidad del
```

```
botón de Servidor_OD
            btServidor_OD.setOnClickListener(new OnClickListener()
            {
                @Override
                public void onClick(View v)
                {
                    boton_Server();
                }
            });
        }

        private void boton_Server() {
            //Se abrirá la página del servidor de OD
            //La URL será del tipo ->     Servidor_OD/cgi-bin/od.cgi?USER=admin&PASS=opendomo
            try {
                string_url = "";
                string_url = login_OD();
                String aviso =  "Servidor de Conexión: " + URL_Host;
                Toast t = Toast.makeText(getApplicationContext(), aviso ,Toast.LENGTH_LONG);
                t.show();
                Intent browserIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(URL_login));
                startActivity(browserIntent);
            } catch (Exception e) {
                defecto_URL();
            }
        }

        private void defecto_URL() {
            String Err_login =  "ERROR EN LOS DATOS DE USUARIO!!! Defecto en la URL del Servidor";
            Toast t2 = Toast.makeText(getApplicationContext(), Err_login ,Toast.LENGTH_LONG);
            t2.show();
            Intent i = new Intent(gps_petGet.this,guardar_usuario.class);  /*método que abrirá la ventana
CONFIGURAR USUARIO */
            startActivityForResult(i, 11);
        }

        void Continuar_URL(){
           if (URLUtil.isValidUrl(URL_Pet_GET) && URLUtil.isValidUrl(URL_Host)){
               pet_get_backgrnd Get_Backgnd = new pet_get_backgrnd();
               Get_Backgnd.execute();
           }else{
                String Err_login =  "ERROR EN LOS DATOS DE USUARIO!!! Defecto en alguna de las URL";
                Toast t2 = Toast.makeText(getApplicationContext(), Err_login ,Toast.LENGTH_LONG);
                t2.show();
           }
        }

        private String login_OD() {
            String[] ficheros = fileList();

            //Este IF sirve para cargar los datos previamente introducidos en la pantalla
            if (existe(ficheros, "user_params.txt")){
                try {
                    InputStreamReader fichero = new InputStreamReader(openFileInput("user_params.txt"));
                    BufferedReader br = new BufferedReader(fichero);
                    String linea = br.readLine();
                    usuario_cargado = "";
                    String passwd_cargado = "";
                    URL_Pet_GET = "";
                    URL_Host = "";
                    URL_login = "";
                    GPS_activo = "";
                    cadena_congps = "";
                    tiempo_gps = "";

                    while (linea != null) {
                        usuario_cargado = usuario_cargado + linea;
                        linea = br.readLine();
                        passwd_cargado = passwd_cargado + linea;
                        linea = br.readLine();
                        GPS_activo = GPS_activo + linea;
                        linea = br.readLine();
                        tiempo_gps = tiempo_gps + linea;
                        linea = br.readLine();
                        URL_Pet_GET = URL_Pet_GET + linea;
                        linea = br.readLine();
                        URL_Host = URL_Host + linea;
                        linea = br.readLine();
                    }
                    br.close();
                    fichero.close();
                    //Se crea la URL de conexón a OD
                    URL_login = URL_login + URL_Host + "?USER=" + usuario_cargado + "&PASS=" + passwd_cargado;
                } catch (IOException e) {
                }
```

```
            }
            else{
                //Si no existe el fichero de configuración de usuario, se pedirán los datos
                Intent i = new Intent(gps_petGet.this,guardar_usuario.class);  /*método que abrirá la ventana
CONFIGURAR USUARIO */
                startActivityForResult(i, 11);
            }
                return URL_Host;
        }

        boolean existe(String[] ficheros, String fichero_busca) {
            for (int f = 0; f < ficheros.length; f++)             //Comprueba si existe el fichero de
credenciales
           if (fichero_busca.equals(ficheros[f]))
               return true;
            return false;
        }

        public class pet_get_backgrnd extends AsyncTask<Context, Object, Object>{

        @Override
        protected Object doInBackground(Context... params) {
            cadena_congps = URL_Pet_GET;
            if (GPS_activo.equals("off_gps")){
                //Si el usuario ha marcado el "Estado de GPS" a "OFF", se hace la petición GET directamente
                string_url = URL_Host;
                peticion_get();
            }else{
                if (GPS_activo.equals("on_gps")){
                        //Si el usuario ha marcado el "Estado de GPS" a "ON",
                        //se van calculando las coordenadas para las peticiones GET correspondientes
                        coordenadas_gps();
/*                      try {  //esto solamente se usa para pruebas. Se deja comentado
                                Thread.sleep(retardo_gps);
                        } catch (InterruptedException e) {e.printStackTrace();}*/
                }
            }
            return null;
        }

        private void peticion_get() {
            //Este método es el que se encarga de realizar las peticiones GET
            //Usará para ello la cadena "string_url".
            Thread thread = new Thread(new Runnable(){
                @Override
                public void run() {
                    HttpClient client = new DefaultHttpClient();
                    URI website = null;
                    try {
                        website = new URI(string_url);
                    } catch (URISyntaxException e) {
                        e.printStackTrace();
                    }

                    HttpGet request = new HttpGet();
                    request.setURI(website);
                    HttpResponse response = null;
                    try {
                        response = client.execute(request);
                        response.getStatusLine().getStatusCode();
                    } catch (ClientProtocolException e) {
                        e.printStackTrace();
                    } catch (IOException e) {
                        e.printStackTrace();
                    } catch (Exception g) {
                        g.printStackTrace();
                    }
                }
            });

            thread.start();
        }

        private void coordenadas_gps(){
            //En este método se calculan las coordenadas GPS, teniendo en cuenta tambien un retardo especificado
por el usuario
            //Para el cálculo de las coordenadas hará uso de la clase "MyLocationListener" que está a continuación
            LocationManager myLocationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
            LocationListener myLocationListener = new MyLocationListener();
            retardo_gps = Long.parseLong(tiempo_gps)*1000*60;//se ha de multiplicar x1000 x60

            while (GPS_activo.equals("on_gps")){
                myLocationManager.requestSingleUpdate(LocationManager.GPS_PROVIDER, myLocationListener,
Looper.getMainLooper());
                try {
```

```
                    Thread.sleep(retardo_gps);    //retardo especificado por usuario para el refresco GPS
                } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                string_url ="";
                string_url = login_OD();
                cadena_congps = URL_Pet_GET;
                retardo_gps = Long.parseLong(tiempo_gps)*1000*60;//se ha de multiplicar x1000 x60
            }
        }

        public class MyLocationListener implements LocationListener{
            private double dblLongitud;
            private double dblLatitud;

            @Override
            public void onLocationChanged(Location loc)
            {    //Metodo que obtiene la latitud y la longitud cuando haya un cambio
                dblLongitud = loc.getLongitude();
                String strLongitud = String.valueOf(dblLongitud);
                longitud = strLongitud;

                 dblLatitud = loc.getLatitude();
                 String strLatitud = String.valueOf(dblLatitud);
                latitud = strLatitud;

                string_url= cadena_congps + "?lat=" + latitud + "&lon=" + longitud;

                //Se realiza la petición GET luego de haber obtenido las coordenadas
                peticion_get();
            }

            public void onProviderDisabled(String provider)
            {
                //Toast.makeText( getApplicationContext(),"Gps Desactivado",Toast.LENGTH_SHORT ).show();
            }
            public void onProviderEnabled(String provider)
            {
                //Toast.makeText( getApplicationContext(),"Gps Activo",Toast.LENGTH_SHORT ).show();
            }
            public void onStatusChanged(String provider, int status, Bundle extras){}
        }
    }
}
```

De esta clase, hay bastante que decir. Para empezar, se puede mencionar el hecho de que el principio es similar a alguna anterior. Se crea un esquema para la representación de la ventana final mediante el fichero "gps_login.xml" que se verá más adelante, donde se definirá un botón para hacer el Login al servidor OpenDomo, y otro que dejará abandonar el programa mediante la opción "Salir". Estos botones serán definidos en el método (**OnCreate**), que derivará también la ejecución de intrucciones según el botón pulsado de la siguiente manera:

- Botón Salir: Llamará a la función **login_OD** y luego al método **Continuar_URL**
- Botón Servidor_OD: Desviará el control a la función **boton_server**

Hecha la introducción anterior, será mejor ir explicando los métodos unos a uno:

-boton_server: Este método mediante un try-catch carga los datos de usuario necesarios a través de "login_OD". Luego hace login en el servidor de OpenDomo. En caso de que haya problemas con la URL de la página principal de OpenDomo, se llama al método "defecto_URL".

-defecto_URL: Esta función se usará para mostrar un mensaje indicando de que hay un error sintáctico en la URL del servidor. Además se encargará de abrir de nuevo la ventana de configuración de usuario para que pueda ser corregido.

-login_OD: Este método devuelve un String que se mostrará luego como un mensaje emergente indicando la URL del servidor. Además sirve para abrir el fichero de configuración de usuario y cargar los datos necesarios para crear parte del String que se necesitará para las peticiones GET. En caso de que no exista el fichero de configuración de usuario, se llamará a los métodos necesarios vistos anteriormente en la clase "guardar_usuario.class".

-**Continuar_URL**: Este método se encarga de validar que las URL que se usarán para las peticiones GET (las que se hacen en background) son correctas. En caso de error muestra un mensaje de aviso y se cierra el programa sin hacer nada. Si las URL fuesen correctas sintácticamente hablando, se preparán las peticiones GET correspondientes a través de las líneas:

```
pet_get_backgrnd Get_Backgnd = new pet_get_backgrnd();
Get_Backgnd.execute();
```

-**existe(String[] ficheros, String fichero_busca)**: Este método pudiera ser importado de otra clase, ya que se ha usado anteriormente. Sin embargo debido a que son líneas contadas se prefiere definirlo aquí otra vez.

Por otro lado, dentro dentro de esta clase se define otra de carácter privado de la siguiente manera: *`private class pet_get_backgrnd extends AsyncTask<Context, Object, Object>`*. Así pues todos los métodos siguientes se encontrarán dentro de ésta, y se ejecutarán en segundo plano. Entre éstos se encuentran los siguientes:

-**doInBackground**: Este método es el más importante. Es el que va a iniciar las operaciones en segundo plano. Aquí se diferencian dos opciones:

- Si el usuario ha elegido que quiere capturar las coordenadas GPS, se llama a la función `coordenadas_gps()`.

-Si el usuario ha elegido que NO quiere mostrar su ubicación, se realiza directamente una única petición GET.

-**peticion_get**: Este método usa la cadena "string_url", para realizar las peticiones GET correspondientes (1 o varias dependiendo de si se incluyen o no las coordenadas GPS).

-**coordenadas_gps**: En este punto se pasa el retardo de lectura de coordenadas GPS especificado por usuario (recordamos que estaba en minutos) a milisegundos. Se controlará este retardo a través de la intrucción "`Thread.sleep(retardo_gps);`". También se instancian los servicios necesarios (`LocationManager`, `LocationListener`), y se define mediante la línea "`myLocationManager.requestSingleUpdate(LocationManager.GPS_PROVIDER, myLocationListener, Looper.getMainLooper());`" cómo se refrescará la lectura de coordenadas GPS (lo que conllevará a nuevas peticiones GET). .

-**MyLocationListener**: Esta clase se ha intanciado en el punto anterior pero se declara aquí. Se usa para obtener las coordenadas GPS (latitud y longitud) y componer la segunda parte del String, que se sumará a la ya existente y que será necesaria para realizar peticiones HTTP. Luego se llamará a la función "peticion_get" que realizará dichas peticiones, pero esta vez con coordenadas GPS.

- **gps_login.xml**: Este fichero se encuentra en "res → layout" como ya se dijo anteriormente. En este caso el contenido es muy sencillo. De echo solo se definen dos botones ("Servidor Opendomo" y "Salir") y un TextView (para poner una indicación) como se muestra a continuación:

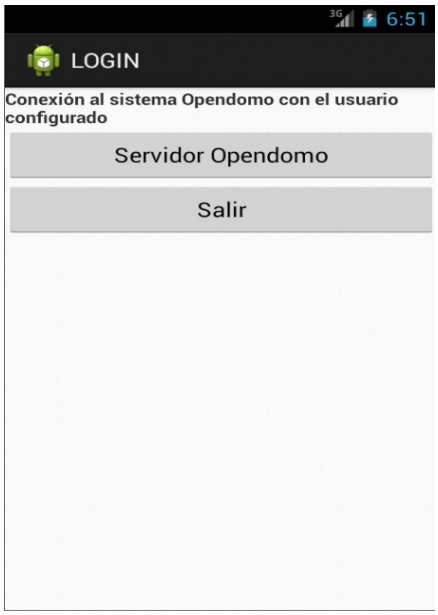
```
<?xml version="1.0" encoding="utf-8"?>
    <TableLayout
    android:id="@+id/widget49"
    android:layout_width="fill_parent"
```

```
        android:layout_height="fill_parent"
        android:orientation="vertical"
        xmlns:android="http://schemas.android.com/apk/res/android"
        >
         <TextView android:id="@+id/LOG"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/LOG"
        android:textStyle="bold" />

     <Button
        android:id="@+id/Servidor_Opendomo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Servidor_Opendomo"
        android:paddingLeft="20dip"
        android:paddingRight="20dip"  />

     <Button
        android:id="@+id/Salir"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Salir"
        android:paddingLeft="20dip"
        android:paddingRight="20dip"  />

</TableLayout>
```



- **strings.xml**: Este fichero se ha ido confeccionando durante todo el desarrollo del programa. Contiene toda la información textual que sale en pantalla salvo los mensajes emergentes. Para traducir la aplicación se debe tocar este fichero. Almacena las cadenas de texto utilizando el formato XML.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">OpenDomo</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="usuario">Usuario: </string>
    <string name="contrasenha">Contraseña: </string>
    <string name="login">LOGIN</string>
    <string name="titulo">Configuración Usuario</string>
    <string name="Login_op">Login OPENDOMO</string>
    <string name="Conf_usuario">Configurar Usuario</string>
    <string name="URL_Opendomo">Web Opendomo</string>
    <string name="guardar">GUARDAR</string>
    <string name="Longitud">Longitud</string>
    <string name="Imagen_OD">OpenDomo 2013</string>
    <string name="conf_usr">CONFIGURACIÓN DE USUARIO</string>
    <string name="estadoGPS">Estado del GPS</string>
    <string name="on">On</string>
    <string name="off">Off</string>
    <string name="URL_petGPS">URL para Geoposicionamiento</string>
    <string name="URL_Server">URL de conexión a Opendomo</string>
    <string name="tiempo_gps">Tiempo de refresco GPS: </string>
    <string name="Aceptar">Aceptar</string>
    <string name="Latitud">Latitud</string>
    <string name="LOG">Conexión al sistema Opendomo con el usuario configurado </string>
    <string name="Salir">Salir</string>
    <string name="Servidor_Opendomo">Servidor Opendomo</string>
</resources>
```

VI. PRUEBAS

En este apartado se citarán un par de pruebas que se usarán para la aplicación dada. Evidentemente una ha de ser probar la aplicación en un móvil, y se hará al final, pero previamente hay que probar la aplicación en modo virtual sobre todo para ver que funciona bien lo que se ejecuta en segundo plano.

Para realizar la prueba dada en primer lugar decir que se va a usar el emulador de Android que usamos hasta ahora. Una ver tengamos abierta la aplicación, y tengamos configurado el usuario, elegimos la primera opción que nos ofrece el programa. Estos pasos tenemos que realizarlos varias veces, indicando unas que se SÍ añadan las coordenadas GPS y otras que NO.

Ahora se necesita instalar un programa que ayudará a escanear la red y buscar las peticiones GET que por ahí van circulando. Para ello se abre una terminal y se lanza el comando "*sudo apt-get install wireshark*". Intalada esta aplicación, es necesario lanzarla con permisos de root, por lo que se abrirá desde consola con el comando "*sudo wireshark*".

Una ver abierto el programa, definiremos una regla para que filtre unicamente el tráfico que se dirige o parte hacia la dirección IP del servidor de OpenDomo.

Hecho todo lo anterior, abrimos un terminal en nuestro sistema operativo. En este caso, y ya que se trata de una lubuntu, teclearemos por orden los siguientes comandos:

-telnet localhost 5554 : nos conectamos al emulador de Android
-geo fix 1.0 1.0
-geo fix 2.0 1.0
-geo fix 4.0 3.0
-geo fix 6.0 1.0
-geo fix 1.0 1.0

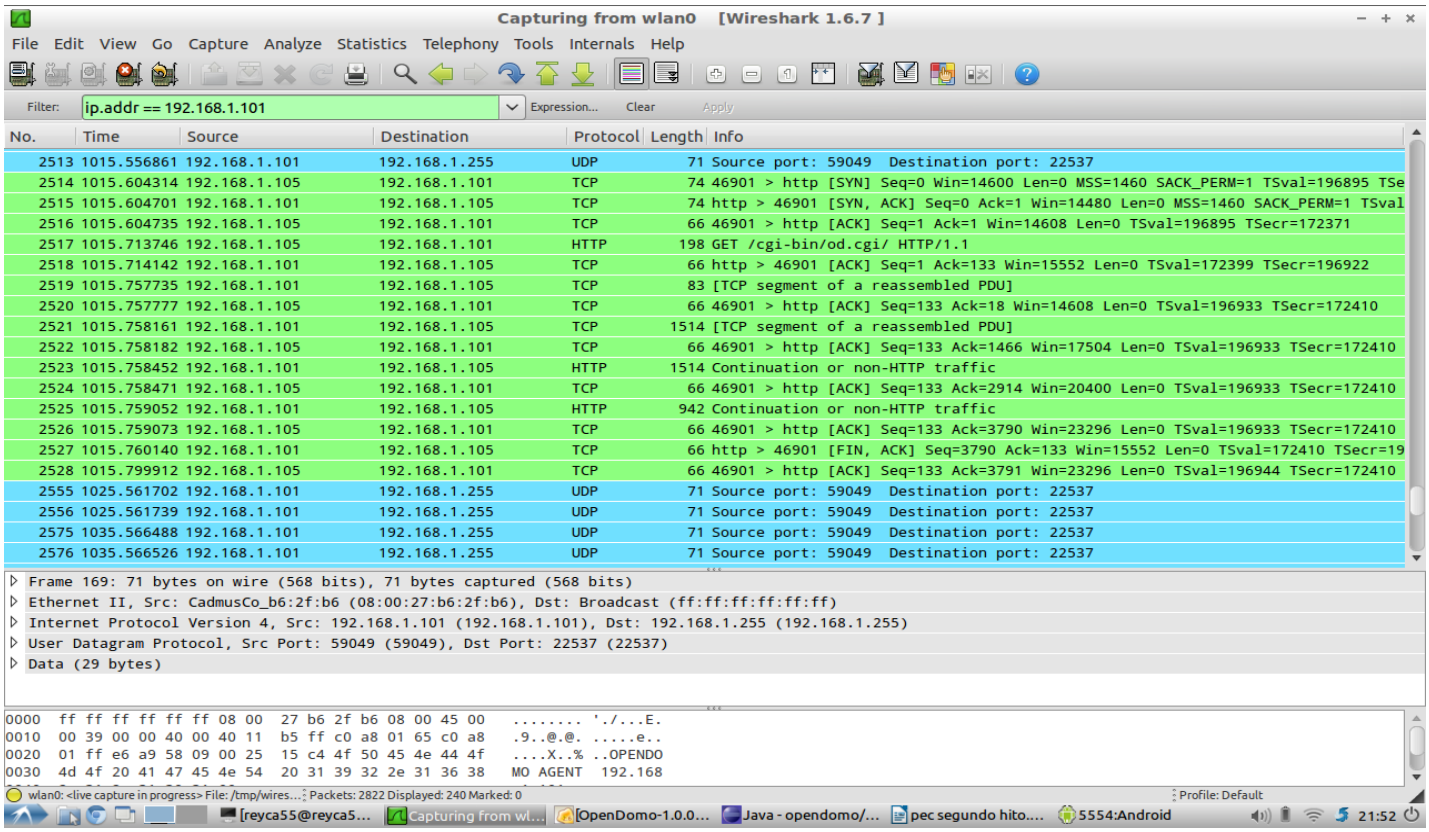
verificamos que con el primer comando del tipo "geo fix" se establecen las coordenadas GPS, y por lo tanto si el usuario tiene la opción de Geoposicionamiento a ON se lanzará una petición GET con las coordenadas. También se comprueba que únicamente se lanzarán las peticiones GET cuando haya expirando el tiempo de refresco GPS especificado por el usuario. Así pues, cuando se lance la petición, se hará con las coordenadas leídas en el momento que ha expirado este tiempo, despreciando todas las anteriores.

Por otro lado, si el usuario tuviese configurado el Geoposicionamiento a OFF la petición se lanzará una única petición en el momento que se le dé al botón "Salir". Además se ignoran los datos de GPS, ya que el formato de la petición como se ha visto en el desarrollo del código, es diferente.

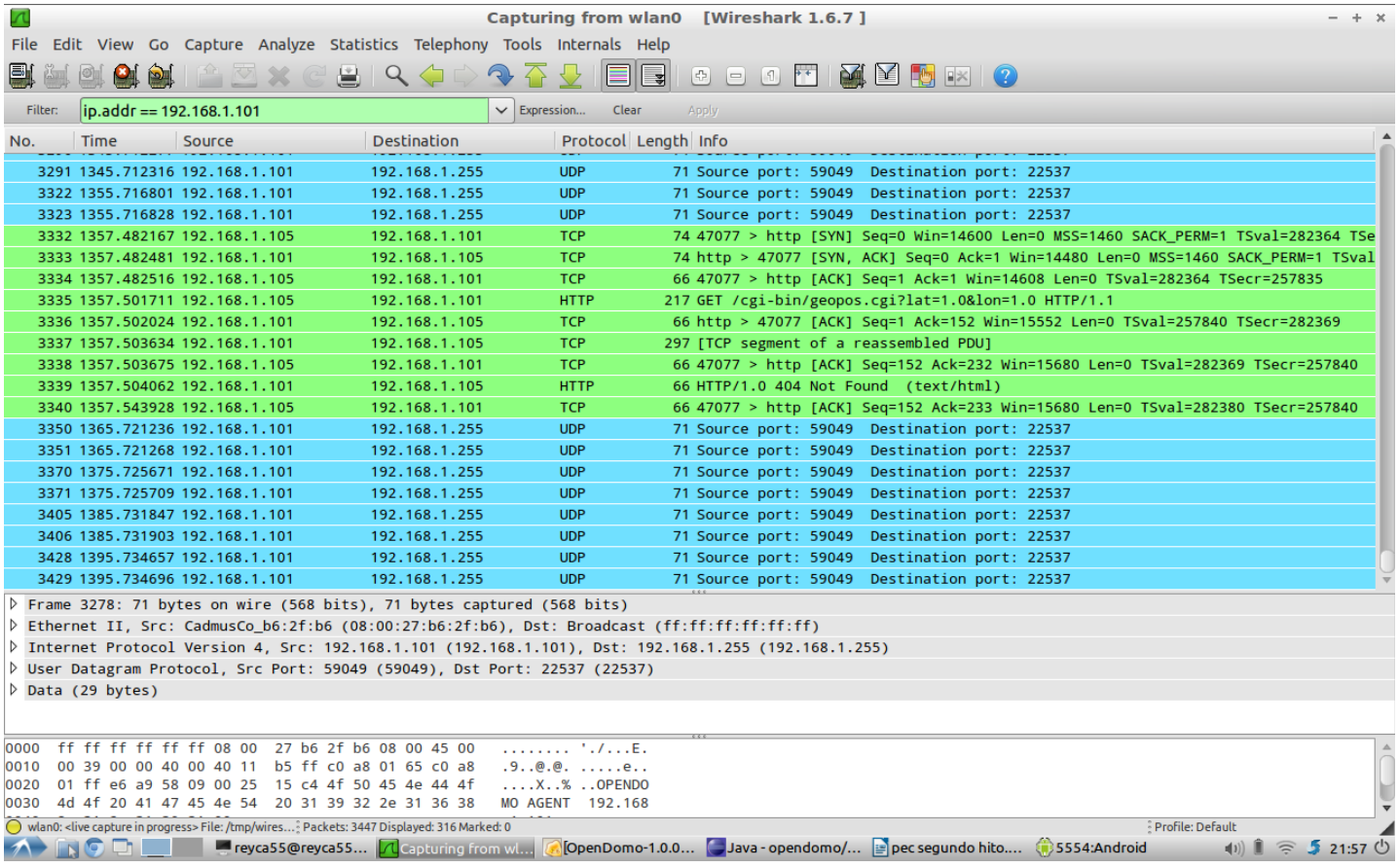
Todas las peticiones son capturadas por el WireShark, y en él pueden también verse los formatos de las peticiones.

Se muestran unas capturas:

- con el GPS a OFF:



– con el GPS a ON:



VII. DOCUMENTACIÓN - DOXYGEN

Como ya se ha visto la documentación del código fuente se ha ido creando al mismo tiempo que se éste se desarrollaba. Ahora ha llegado el turno de hacer la documentación de la aplicación.

Dicho lo anterior, y dado que entre sus ventajas está el formato en que genera la documentación, vamos a utilizar Doxygen para realizar este apartado. Para ello el primer paso es instalar los paquetes necesarios, cosa que haremos desde consola con el comando "sudo apt-get install doxygen". Este proceso puede llevar tiempo ya que es probable que necesite resolver muchas dependencias.

Para continuar, es necesario crear el fichero de configuración de Doxygen. Para ello usaremos el comando "doxygen -g", que nos creará el archivo "Doxyfile". Pasamos a modificarlo. No tendremos problemas a la hora de entender las opciones ya que hay multitud de comentarios. Por ejemplo, cambiaremos:

- línea 29: El nombre del proyecto (Por ejemplo PROJECT_NAME = "OpenDomo")
- línea 35: La versión (PROJECT_NUMBER = 0.1)
- línea 55: Directorio de salida (OUTPUT_DIRECTORY = ./doc)
- línea 77: El idioma (OUTPUT_LANGUAGE = Spanish)
- línea 651: Los ficheros de entrada (INPUT = …'ficheros para los que se genera la documentación")
- línea 778: Si queremos poder navegar por el código (SOURCE_BROWSER = YES)
- línea 855: Si queremos generar la documentación HTML (GENERATE_HTML = YES por defecto)
- línea 1213: Si queremos generar la documentación LaTeX (GENERATE_LATEX = YES por defecto)
- línea 1354: Si queremos generar la documentación MAN (GENERATE_MAN = YES)
- línea 1383: Si queremos generar la documentación XML (GENERATE_XML = YES)

Podemos tardar un buen rato en configurar la documentación a nuestro gusto. Una vez que ya lo tenemos simplemente ejecutamos el comando "doxygen" que nos creará la documentación en los formatos especificados.

Sin embargo existe una manera mucho más fácil de crear la documentación usando Doxygen y Eclipse. Se trata de usar un plugin. Una vez tenemos instalado Doxygen, cosa que se hace como se ha visto en el parrafo anterior, se instala el plugin Eclox de Doxygen para Eclipse como se indica a continuación:

1. En Eclipse, ir a "Help → Install New Software... " y añadir el lugar "http://download.gna.org/eclox/update" dándole el nombre eclox.
2. Luego hay que seleccionar todos los paquetes encontrados y aceptar la licencia. También hay que reiniciar Eclipse. Fijarse que hecho lo anterior, aparece un botón con el símbolo @ en la barra de herramientas.

Para construir la documentación una ver comentado el código fuente, se accederá a "File → New → Other..." y seleccionamos el desplegable "Other → Doxyfile". Luego se pedirá rellenar dos campos que son:

- Parent Folder: opendomo
- Doxyfile name: opendomo.doxyfile

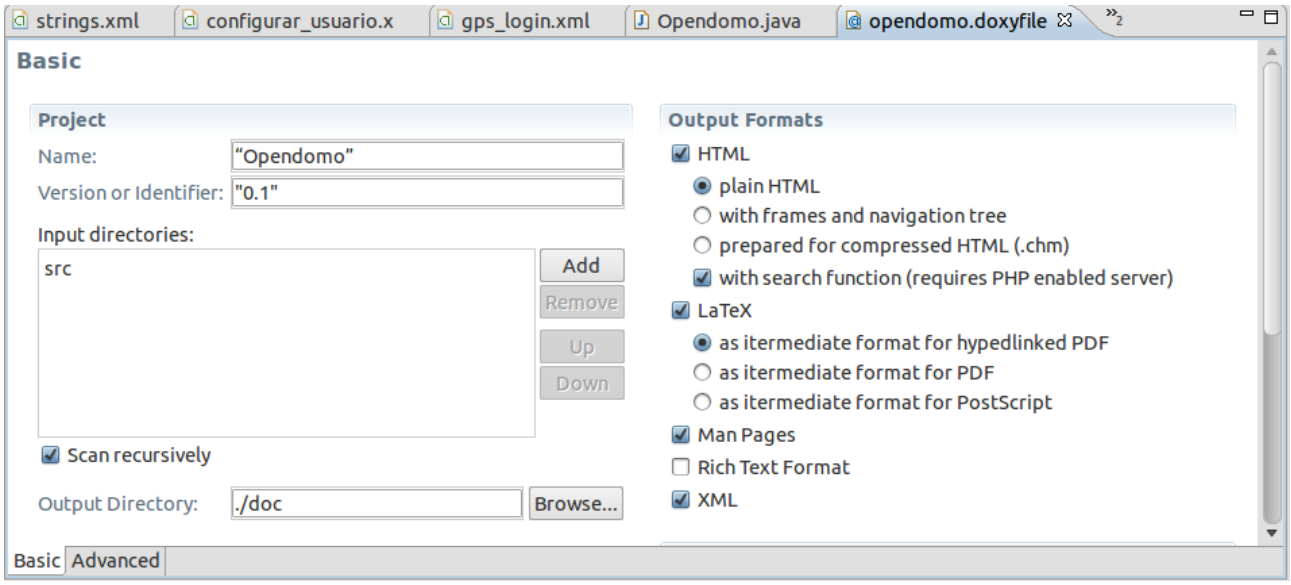
Para continuar vamos a configurar unos parámetros abriendo el fichero

“opendomo.doxyfile” que cuelga directamente del raíz en Eclipse. Así en la pestaña “Basic” se modificarán los parámetros:

- Project Name = “Opendomo”
- Version or Identifier = “0.1”
- Input directories. A través del botón “Add” se añadirá el directorio “src” que es donde se hallan todas las clases java del proyecto Opendomo. Además se marcará la casilla “Scan recursively”, para que las añada todas de golpe.
- Output Directory = “./doc”
- Y marcaremos las casillas “Man Pages” y “XML”.

mientras que en “Advanced” se pueden cambiar:

- Output Language = “Spanish”
- Source Browser = “YES”
- Extract Private = “YES” (Este parámetro se cambia para poder ver la documentación completa)
- Otros muchos parámetros al gusto.



Para finalizar, pinchando en el botón “@” de la barra de herramientas se generará la documentación correspondiente.

Si quisieramos ver la documentación de la aplicación, tendríamos que acceder al directorio “/opendomo/doc/”. Dentro de éste, se encuentra en los formatos elegidos.

VIII. TRABAJO FUTURO

Como hemos visto que lo que se está haciendo es desarrollar una nueva aplicación, y dado que ésta todavía está en desarrollo, seguramente se hagan cambios en ella. Por ello hay que tener en cuenta que más adelante seguramente haya algunas modificaciones, sobre todo en las áreas siguientes:

- Logueo en el sistema OpenDomo. (Es probable que se cambie el sistema por el que se hace el Logueo)

- Búsqueda del servidor Opendomo. (Es probable que se haga alguna modificación para que la aplicación detecte directamente el servidor OpenDomo)

IX. CONCLUSIONES Y LECCIONES APRENDIDAS

Como punto final de la memoria se mostrarán las conclusiones y lecciones aprendidas. En referencia al primer punto no ha mucho que decir. Tal como se mostraba en el punto II, la aplicación consta basicamente de tres pantallas. Dos de ellas (“Configuración de Usuario” y “Pantalla inicial”), han sido relativamente fáciles de desarrollar. Con respecto a la tercera (“Login en el Servidor”) hay que diferenciar dos partes, para mencionar algún problema que ha habido:

- Parte 1): Comprende el código que ha sido necesario implementar para llevar a cabo la función de Login en el Servidor. Realmente y aunque esta funcionalidad es la principal del programa, las líneas de código que se le han dedicado en este pantalla son pocas. Por ello y por que la funcionalidad a implementar era bastante sencilla, no hubo problemas en su desarrollo (Unicamente había que contruir un String con datos de la configuración de usuario, y usarlo como una URL accediendo a ella a través de un navegador).

- Parte 2): Ésta ha sido la parte más complicada del diseño de la aplicación con diferencia. Aunque en principio el lenguaje de programación se parece a otros (de hecho es realidad es una API de JAVA), nunca había implementado nada que tuviese las siguientes funcionalidades, y que han dado sus quebraderos de cabeza:
  - Módulo para enviar peticiones GET. No ha sido lo más complicado, pero nunca había hecho nada parecido. Estaba un poco perdido al principio.
  - Funcionalidad de Geolocalización. Aunque no es una tarea complicada implementar esta funcionalidad, el hecho que nunca se hubiera trabajado con coordenadas GPS ha implicado el tener que repasar mucha documentación. Además el tener que realizar esta tarea en background, tiene cierta complejidad añadida.
  - El uso se la clase Asynctask. Igual que en los dos casos anteriores, nunca se había implementado ningún servicio que corriese en segundo plano. Conseguir leer las coordenadas GPS en segundo plano, y enviar las peticiones GET correspondientes ha sido complicado.
    Sin embargo la tarea más complicada de todo el desarrollo, fue conseguir terminar las tareas en segundo plano que fueron lanzadas en la ejecución anterior de la aplicación. Al final buscando por foros, se encontró que la solución pasaba por usar el método “`requestSingleUpdate`” para “`myLocationManager`” usando una intrucción a mayores para el retardo de lectura de coordenadas GPS, en lugar de usar el método “`requestLocationUpdates`”.

Como punto final estaría bien mencionar las lecciones aprendidas en el desarrollo de este proyecto. Se podría decir que éstas guardan relación con los problemas sufridos en el desarrollo, aunque realmente hay más cosas que ver. Además de aprender como desarrollar los puntos anteriores, se han aprendido varias cosas más, y entre las más destacables se pueden citar:

- El uso de determinadas herramientas de red entre las que destacan WireShark, herramienta necesaria para el desarrollo del proyecto. Aunque ya se conocía este programa, se ha adquirido más destreza en su manejo.

- El uso de herramientas para la generación automática de la documentación. Ya vistas estas herramientas en otras asignaturas del máster, el uso de Doxygen en particular ha sido de gran ayuda en el desarrollo del proyecto. Además se ha visto como se puede integrar facilmente con Eclipse, a través de Eclox.

En resumen, las asignaturas del máster, y el proyecto en particular, han ofrecido unos conocimientos muy interesantes, tanto en el área de sistemas como en programación. Así pues, creo que el temario dado además de interesante, aporta conocimiento sobre bastantes áreas. Es por ello que me ha resultado un máster muy interesante, si bien también creo que se debería incluir algo de virtualización en las asignaturas relacionadas con el área de sistemas. La virtualización supone un ahorro en la implantación de las infraestructuras informáticas, y actualmente se está extendiendo el uso de esta tecnología, impulsada también en parte por la crisis económica.

X. REFERENCIAS

En este apartado se presentarán algunas referencias que han ayudado al desarrollo de la aplicación. También se mencionará en qué punto han ayudado.

- Para la preparación del entorno de desarrollo
  http://www.nosinmiubuntu.com/2011/10/instalacion-del-sdk-de-android-en.html

- Para el desarrollo general de las pantallas de la aplicación
  http://www.sgoliver.net/blog/?p=1467

- Para el guardado en fichero de la configuración de usuario
  http://www.javaya.com.ar/androidya/detalleconcepto.php?codigo=143&inicio

- Para el apartado donde se realiza la captura de coordenadas GPS
  http://www.elandroidelibre.com/2010/08/aprendiendo-android-v-inicializacion-a-la-api-del-gps.html

- Para buscar soluciones de alguna duda surgida en la códificación de algún módulo, en el siguiente portal los desarrolladores de Android aportaron algo de ayuda
  http://stackoverflow.com/

- Para la documentación de la aplicación
  http://www.disca.upv.es/aperles/mola_doxygen/mola_doxygen.html
  http://www.stack.nl/~dimitri/doxygen/manual/config.html#cfg_tab_size