UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática

Universidad Politécnica de Valencia

Proyecto Friend Scout

Proyecto Final de Carrera

Ingeniería Técnica de Informática de Gestión

Autor: Andrés Francisco Martínez Broseta

Director: Juan Carlos Ruiz García

Director ITT Dublín: Enda Lee

(School of Science & Computing)

11/07/2014

Resumen

El proyecto es una aplicación para sistemas operativos Android que permite compartir tu localización geográfica con tus amigos. Se compone de la aplicación Android, de una base de datos que se encuentra en un servidor contratado y de los mecanismos necesarios para la comunicación entre éstos.

Palabras clave: Android, geolocalización, java, aplicación.

Tabla de contenidos

1. Introducción

Como entorno de programación utilizamos Eclipse. Este software es un entorno de desarrollo integrado (IDE) multilenguaje, escrito mayoritariamente en Java, y altamente extensible mediante plugins.

Google, proveyendo el SDK, proporciona un conjunto de herramientas que permiten comenzar a desarrollar tu proyecto muy rápidamente. Este paquete también incluye un emulador que te da la posibilidad de testear tus apps de una manera fácil y cómoda aunque también tienes la opción de utilizar tu propio dispositivo. Además, también existe una gran comunidad que te da un gran soporte frente a las dudas y problemas que van apareciendo durante el proceso de desarrollo del proyecto.

La versión del SDK por la que optamos fue la 4.0.3 ICS, ya que era la más reciente, y para la versión mínima requerida 2.2 Froyo. El motivo de esta decisión fue porque así sería compatible en la gran mayoría de dispositivos que hay en el mercado.

Para los mapas decidimos que la mejor opción era Google Maps ya que es el API de mapas que está mejor integrada. Además no teníamos necesidad de utilizar opciones de enrutamiento, cosa que Google no permite en sus mapas.

Utilizamos el sistema de datos MySql porque es el que normalmente se utiliza en la mayoría de empresas y servidores del mercado.

2. Objetivos

Aun teniendo en cuenta lo que hemos aprendido durante los cursos en la Escuela de Informática, nosotros cometimos un error que ya nos comentaron que ocurría muy a menudo cuando desarrollas software. Estoy hablando de objetivos demasiado optimistas y que quedan fuera del alcance. Pues esto fue un gran problema ya que nosotros planteamos una aplicación más grande de la que finalmente desarrollamos.

En un principio nosotros queríamos que esta aplicación tuviera más contenido como el que podías encontrar en las redes sociales o en otros programas, ya sea un chat, alarmas, etc. Pensamos que sería una gran idea implementar una alarma que sonara cuando llegases a ciertos lugares y así, por ejemplo, poder recordar si tienes que comprar alguna cosa concreta cuando pasas cerca de un supermercado de tu elección.

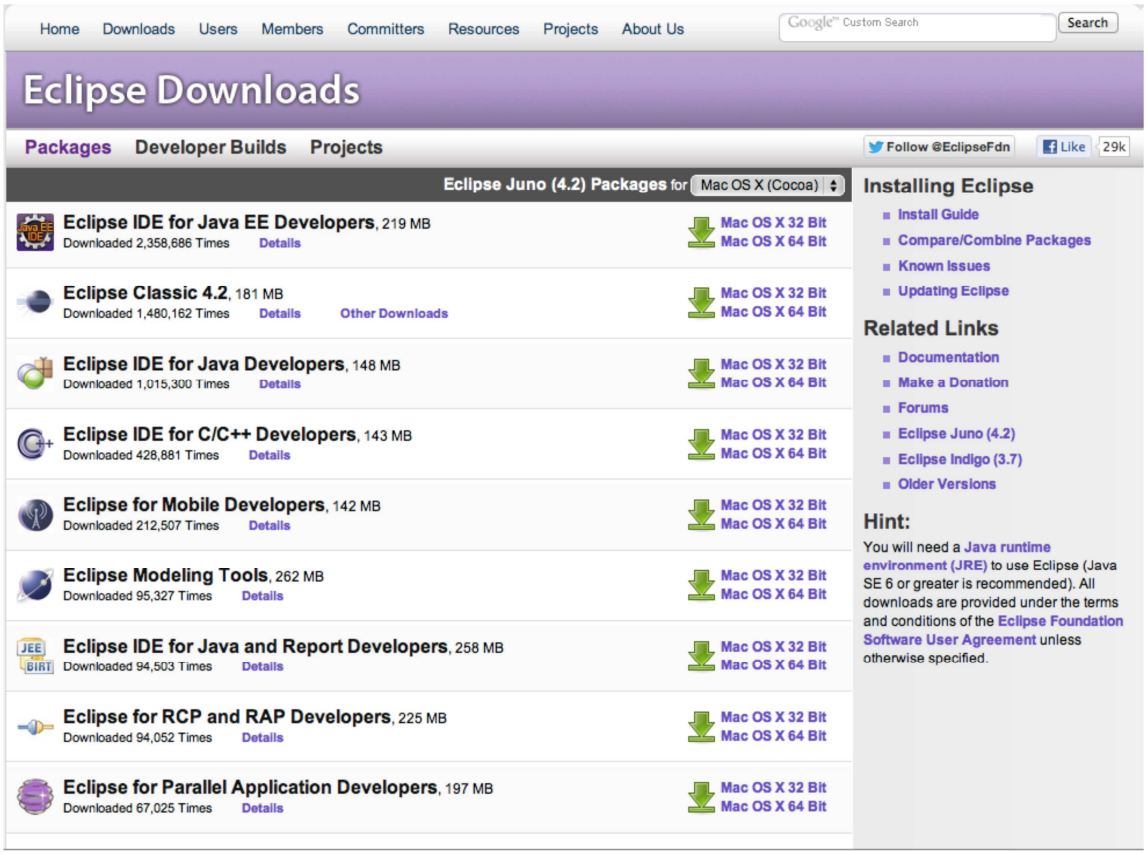
Finalmente nos dimos cuenta que estos objetivos eran demasiado optimistas y nos limitamos a hacer la aplicación básica con opción a una futura expansión. Los objetivos que nos planteamos fueron los siguientes:

- Una aplicación con un login, un menú donde podías elegir un mapa donde ver a tus amigos, una lista donde visualizarlos ordenados por cercanía a tu posición, un editor y visor de tu información y perfil, y un menú de opciones y ajustes.
- Un servidor con una base de datos para gestionar los diferentes usuarios de la app.

3. Pasos previos al desarrollo

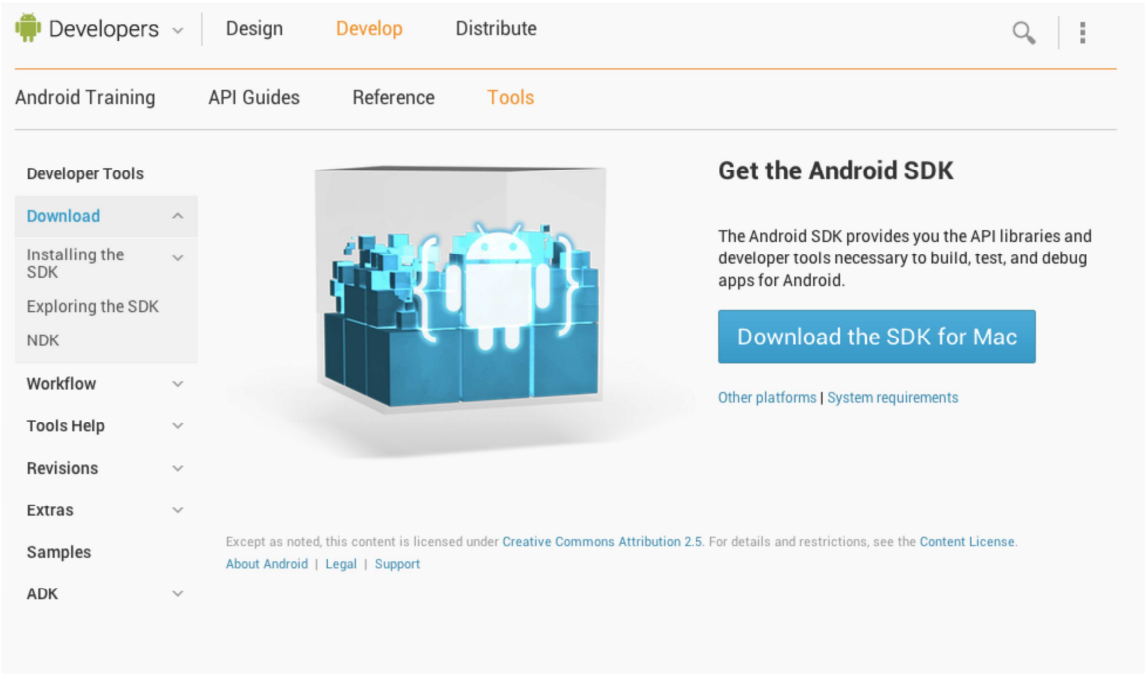
3.1 Instalación del entorno de trabajo

Aunque Eclipse no forma parte del proyecto es una parte fundamental para su desarrollo. El IDE (Classic) se descarga gratuitamente de su página web oficial (http://www.eclipse.org). Después de su descarga sólo es necesaria una instalación típica para comenzar a trabajar.



3.2 Instalación del SDK de Android

Como en el paso anterior sólo se necesita descargar el paquete de la web de desarrolladores de Android (http://developer.android.com) e instalarlo tal y como se indica en la página web.



3.3 Contratación del servidor

Contratamos un servidor web básico con soporte para MySql y PHP. Elegimos los servidores de NTC Hosting (http://www.ntchosting.com) ya que nos hicieron una buena oferta y cubría todas nuestras necesidades.

Aunque podíamos obtener opciones similares de manera gratuita, las garantías que nos daban por el bajo precio nos parecieron razonables.

4. Equipo utilizado

4.1 Equipo de desarrollo

- MacBook Air 13” 2011
  - Mac OS X Lion 10.7.4
  - 1,7 GHz Intel Core i5
- MacBook Pro 13” 2011
  - Mac OS X Lion 10.7.4
  - 2,7 GHz Intel Core i7

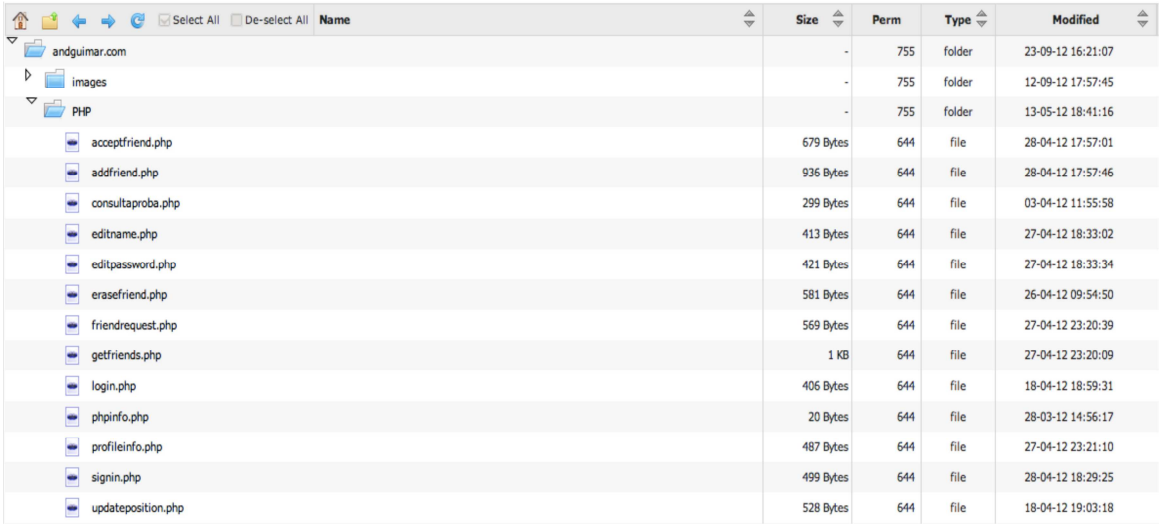
4.2 Equipo de testeo

- HTC Desire Android 2.3 Ginger Bread
- LG P-970 Optimus Black 2.3 Ginger Bread
- Samsung Galaxy SII 4.0.1 Ice Cream Sandwich
- Samsung Galaxy Mini 2.2 Frozen Yogurt

5. Implementación

5.1 Equipo del servidor

El servidor está compuesto por tres partes, los scripts PHP que gestionan la base de datos MySql, la propia base de datos y las imágenes de perfil de los usuarios.



Name	Size	Perm	Type	Modified
andguimar.com	-	755	folder	23-09-12 16:21:07
images	-	755	folder	12-09-12 17:57:45
PHP	-	755	folder	13-05-12 18:41:16
acceptfriend.php	679 Bytes	644	file	28-04-12 17:57:01
addfriend.php	936 Bytes	644	file	28-04-12 17:57:46
consultaproba.php	299 Bytes	644	file	03-04-12 11:55:58
editname.php	413 Bytes	644	file	27-04-12 18:33:02
editpassword.php	421 Bytes	644	file	27-04-12 18:33:34
erasefriend.php	581 Bytes	644	file	26-04-12 09:54:50
friendrequest.php	569 Bytes	644	file	27-04-12 23:20:39
getfriends.php	1 KB	644	file	27-04-12 23:20:09
login.php	406 Bytes	644	file	18-04-12 18:59:31
phpinfo.php	20 Bytes	644	file	28-03-12 14:56:17
profileinfo.php	487 Bytes	644	file	27-04-12 23:21:10
signin.php	499 Bytes	644	file	28-04-12 18:29:25
updateposition.php	528 Bytes	644	file	18-04-12 19:03:18

Los scripts PHP

acceptfriend.php

Este script escribe en la tabla friends de la base de datos los nombres de dos usuarios que acaban de aceptar una petición de amistad, y los borra de la tabla friendrequest donde se almacenan estas peticiones pendientes.

```php
<?php
// .$_REQUEST['serverpassword'].
// .$_REQUEST['username'].
      mysql_connect( "localhost","nikomac_fs",$_REQUEST['serverpassword']);

      mysql_select_db("nikomac_fs");



      $q=mysql_query("INSERT INTO friends (friend1,friend2)
                        VALUES ('".$_REQUEST['username']."','".$_REQUEST['friend']."')");

        if(!$q){
                die('error1');

        }


      $q1=mysql_query("DELETE FROM friendrequest
                        WHERE uname = '".$_REQUEST['friend']."' AND friend='".$_REQUEST['username']."'");

        if(!$q1){
                die('error2');

        }
      //echo mysql_result($result, 2);

    mysql_close();
?>
```

addfriend.php

En este script es donde se tramita la petición de amistad de un usuario. Primero se comprueba que estos dos usuarios no son amigos todavía y seguidamente, si no lo son, se hace una nueva entrada en la tabla friendrequest.

```php
<?php
// .$_REQUEST['serverpassword'].
// '".$_REQUEST['username']."'
     mysql_connect( "localhost","nikomac_fs",$_REQUEST['serverpassword']);

     mysql_select_db("nikomac_fs");

       $q1=mysql_query("SELECT * FROM friends WHERE '".$_REQUEST['username']."'= friends.friend1 AND friends.friend2 = '".$_REQUEST['friend']."'");

       $q2=mysql_query("SELECT * FROM friends WHERE '".$_REQUEST['username']."'= friends.friend2 AND friends.friend1 = '".$_REQUEST['friend']."'");


    if((mysql_num_rows($q1)!=0)||(mysql_num_rows($q2)!=0)){ die('error1'); }

       $q3=mysql_query("SELECT * FROM users WHERE '".$_REQUEST['friend']."'= users.username");

    if(mysql_num_rows($q3)==0){ die('error2');}

     $q=mysql_query("INSERT INTO friendrequest (uname,friend) VALUES ('".$_REQUEST['username']."','".$_REQUEST['friend']."')");

       if(!$q){
              die('error');
       }

   mysql_close();
?>
```

consultaproba.php

Este script fue una prueba para comprobar que el servidor funcionaba correctamente. Por motivos de seguridad no se muestra el script.

editname.php

Este script sirve para modificar el nombre y apellido del usuario.

```php
<?php
// .$_REQUEST['serverpassword'].
// .$_REQUEST['username'].
      mysql_connect( "localhost","nikomac_fs",$_REQUEST['serverpassword']);

      mysql_select_db("nikomac_fs");




      $q=mysql_query("UPDATE users SET name ='".$_REQUEST['name']."'
                        WHERE username ='".$_REQUEST['username']."'");

        if(!$q){
                die('error');

        }

    mysql_close();
?>
```

editpassword.php

Este script es como el anterior, pero en este caso modificamos el campo password del usuario.

```php
<?php
// .$_REQUEST['serverpassword'].
// .$_REQUEST['username'].
      mysql_connect( "localhost","nikomac_fs",$_REQUEST['serverpassword']);

      mysql_select_db("nikomac_fs");




      $q=mysql_query("UPDATE users SET password ='".$_REQUEST['password']."'
                        WHERE username ='".$_REQUEST['username']."'");

        if(!$q){
                die('error');

        }

    mysql_close();
?>
```

erasefriend.php

Este script borra un amigo de tu lista de amigos. Como en la tabla de amigos una amistad puede estar representada como amigo1 - amigo2 ó amigo2 - amigo1, la comprobación ha de hacerse desde ambos lados.

```php
<?php
// .$_REQUEST['serverpassword'].
// .$_REQUEST['username'].   $outputfinal = array_merge($output,$output2); }
      mysql_connect( "localhost","nikomac_fs",$_REQUEST['serverpassword']);

      mysql_select_db("nikomac_fs");

          $q=mysql_query("DELETE FROM friends
                        WHERE friend1 = '".$_REQUEST['friend']."' AND friend2='".$_REQUEST['username']."'
                         OR friend2 = '".$_REQUEST['friend']."' AND friend1='".$_REQUEST['username']."' ");

        if(!$q){
                die('error2');

        }

    print(json_encode($outputfinal));

    mysql_close();
?>
```

friendrequest.php

En este script recopilamos las peticiones de amistad que puede tener una persona pendientes.

```php
<?php
// .$_REQUEST['serverpassword'].
// .$_REQUEST['username'].
      mysql_connect( "localhost","nikomac_fs",$_REQUEST['serverpassword']);

      mysql_select_db("nikomac_fs");
      mysql_query("SET NAMES 'utf8'");

      $table=$_REQUEST['username'];

      $q=mysql_query("SELECT users.name, users.username FROM users,friendrequest WHERE '".$table."'= friendrequest.friend AND
                        friendrequest.uname = users.username");



    while($e=mysql_fetch_assoc($q))

          $output[]=$e;

     print(json_encode($output));

   mysql_close();
?>
```

getfriends.php

Este script busca todas las relaciones de amistad que puede tener un usuario y la información de estas amistades (nombre, nombre de usuario, latitud y longitud).

```php
<?php
// .$_REQUEST['serverpassword'].
// .$_REQUEST['username'].   $outputfinal = array_merge($output,$output2); }
      mysql_connect( "localhost","nikomac_fs",$_REQUEST['serverpassword']);

      mysql_select_db("nikomac_fs");
      mysql_query("SET NAMES 'utf8'");

      $table=$_REQUEST['username'];

        $q=mysql_query("SELECT users.name,users.username, users.latitude, users.longitude FROM users,friends
        WHERE '".$table."'= friends.friend1 AND friends.friend2 = users.username");

        $q2=mysql_query("SELECT users.name,users.username, users.latitude, users.longitude FROM users,friends
        WHERE '".$table."'= friends.friend2 AND friends.friend1 = users.username");

     while($e=mysql_fetch_assoc($q))

            $output[]=$e;

      //print(json_encode($output));

     while($e2=mysql_fetch_assoc($q2))

            $output2[]=$e2;


      if((mysql_num_rows($q)!=0)&&(mysql_num_rows($q2)!=0)){ $outputfinal = array_merge($output,$output2);
      }elseif(mysql_num_rows($q)!=0){  $outputfinal = $output;
      }elseif(mysql_num_rows($q2)!=0){ $outputfinal = $output2;
      }

      print(json_encode($outputfinal));

    mysql_close();
?>
```

login.php

En este script se hace una comprobación en la tabla de usuarios para ver si el usuario existe y si la contraseña encriptada es igual a la que hay en el servidor.

```php
<?php
// .$_REQUEST['serverpassword'].
// .$_REQUEST['username'].
      mysql_connect( "localhost","nikomac_fs",$_REQUEST['serverpassword']);

      mysql_select_db("nikomac_fs");

      $q=mysql_query("SELECT * FROM users WHERE username ='".$_REQUEST['username']."'");

      while($e=mysql_fetch_assoc($q))

              $output[]=$e;

           print(json_encode($output));

    mysql_close();
?>
```

phpinfo.php

Este script, cuando es invocado, devuelve la información de PHP del servidor. También fue implementado para hacer pruebas con el servidor.

profileinfo.php

Este script devuelve toda la información propia del usuario para mostrarla en su perfil.

```php
<?php
// .$_REQUEST['serverpassword'].
// .$_REQUEST['username'].
      mysql_connect( "localhost","nikomac_fs",$_REQUEST['serverpassword']);

      mysql_select_db("nikomac_fs");
      mysql_query("SET NAMES 'utf8'");

      $q=mysql_query("SELECT users.name, users.latitude, users.longitude FROM users
      WHERE username ='".$_REQUEST['username']."'");

      while($e=mysql_fetch_assoc($q))

              $output[]=$e;

           print(json_encode($output));

    mysql_close();
?>
```

signin.php

Este script sirve para incluir un nuevo usuario en la tabla users de la base de datos. Inicialmente el usuario se inscribe con la posición de latitud y longitud 0.

```php
<?php
// .$_REQUEST['serverpassword'].
// '".$_REQUEST['username']."'
     mysql_connect( "localhost","nikomac_fs",$_REQUEST['serverpassword']);

     mysql_select_db("nikomac_fs");

     $q=mysql_query("INSERT INTO users (username,name,password,latitude,longitude)
                     VALUES ('".$_REQUEST['username']."','".$_REQUEST['name']."','".$_REQUEST['password']."',0,0)");
       if(!$q){
              die('error');

       }

     //echo mysql_result($result, 2);

   mysql_close();
?>
```

updateposition.php

Finalmente, este script actualiza la posición de un usuario (su latitud y longitud) dentro de la tabla users.

```php
<?php
// .$_REQUEST['serverpassword'].
// '".$_REQUEST['username']."'
      mysql_connect( "localhost","nikomac_fs",$_REQUEST['serverpassword']);

      mysql_select_db("nikomac_fs");

      $q=mysql_query("UPDATE users SET latitude = '".$_REQUEST['latitude']."', longitude = '".$_REQUEST['longitude']."'
                     WHERE username = '".$_REQUEST['username']."' ");


        if(!$q){
                die('error');

        }

      //echo mysql_result($result, 2);

    mysql_close();
?>
```

La Base de Datos

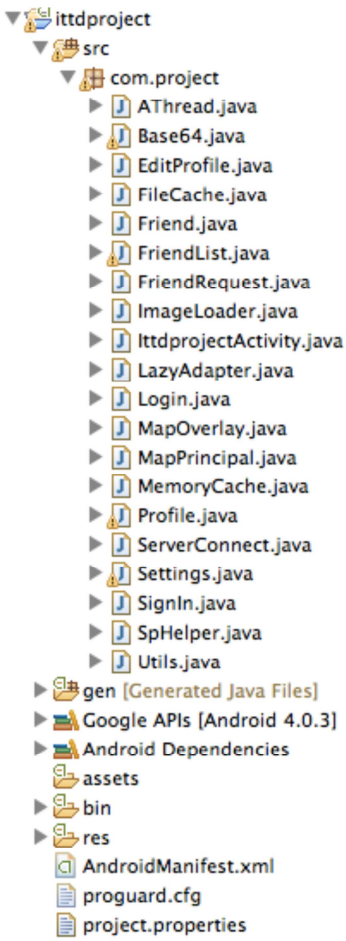
La base de datos está compuesta por 3 tablas: users, friends, friendrequest.

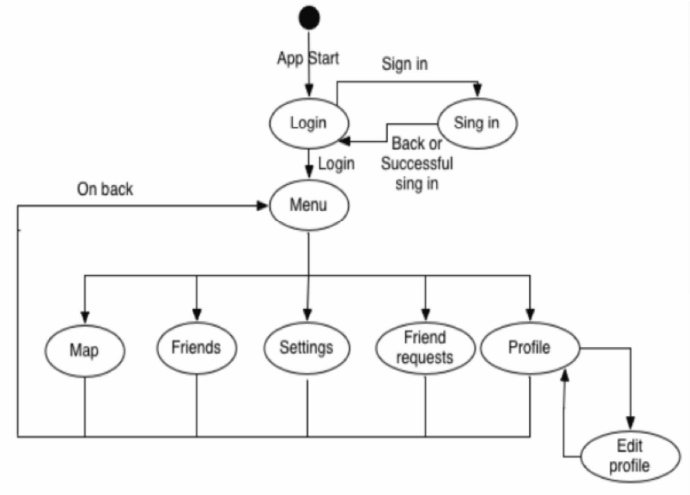
* User
  * Username (Primary key)
  * Name
  * Password
  * Latitude
  * longitude
* Friends
  * Friend1 (Foreing key -> User)
  * Friend2 (Foreing key -> User)
* Friend requests
  * Uname (Foreing key -> User)
  * Friend (Foreing key -> User)

Las Imágenes

Las imágenes de los usuarios están almacenadas en una carpeta denominada images accesible desde la aplicación, para mostrar los avatares de los usuarios que son amigos.

5.2 Implementación de la aplicación Android



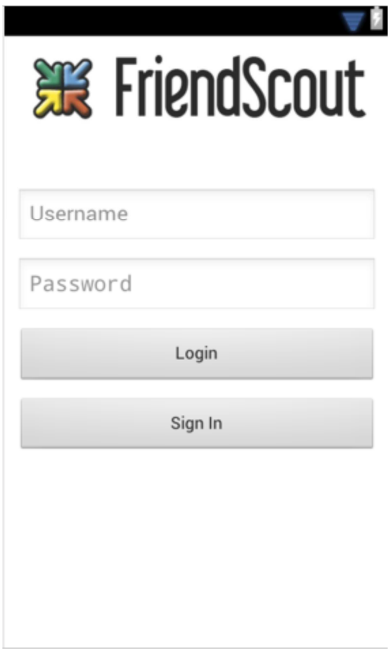


La aplicación consta de clases que tienen layout o principales de la aplicación (que serán descritas primero) y las clases de soporte (que serán descritas posteriormente).

Login

Principalmente es una activity con su correspondiente layout que describe su interfaz gráfica. A continuación describiremos las principales partes de estos archivos de código y cómo se relacionan con las otras partes de la aplicación.

login.xml



La interfaz gráfica consta de un ImageView con el logo de la aplicación, dos EditText (uno para el username y otro para el password) y finalmente dos botones (uno para loguearse y otro para registrar un nuevo usuario.

Login.java

En el método onCreate() de la activity accedemos a las diferentes Views descritas en la interfaz, registramos los onClickListener() para los botones y comprobamos si el usuario está logueado para pasar directamente al menú. Para esta transacción necesitamos acceder a las SharedPreferences mediante la clase SpHelper que será descrita más adelante.

```
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    //initialize the classes
    setContentView(R.layout.login);

    login = (Button)findViewById(R.id.bLogin);
    login.setOnClickListener(this);
    Signin = (Button)findViewById(R.id.bSignIn);
    Signin.setOnClickListener(this);
    sph = new SpHelper(this);
    Sc = new ServerConnect(this);

    //if there is a user logged in finish the login activity and go to the menu
    if(sph.checkSavedUser()){
           finish = true;
           startActivity(new Intent(getApplicationContext(),
                   IttdprojectActivity.class));
       }

    username = (EditText) findViewById(R.id.etLoginUser);
    pass = (EditText) findViewById(R.id.etLoginPassword);
}
```

El onClick() del botón de login hace una conexión al servidor mediante la clase ServerConnect para comprobar si la pareja usuario-contraseña es correcta. Por otro lado, el de Signin crea un nuevo Intent de Signin.

Signin

Esta activity es similar a la anterior pero, en lugar de intentar loguear al usuario, crea una nueva entrada en la base de datos si el nombre de usuario no está en uso.

signin.xml

Como ya hemos dicho la interfaz es muy similar con el logo, unos cuantos EditText para introducir los datos del usuario y un botón para registrarse.

Signin.java

```
public void onClick(View arg0) {
    // TODO Auto-generated method stub

    usern = username.getText().toString();
    nname = namee.getText().toString();
    passw = pass.getText().toString();
    repassw = repass.getText().toString();

    if(!(usern.equals(""))){
        if(!(nname.equals(""))){
                if((!(passw.equals("")))&&(passw.equals(repassw))){

                    pd = ProgressDialog.show(SignIn.this, "","Signing In...", true);
                    AThread at = new AThread(pd);
                    at.start();
                    error = Sc.trySignIn(usern,nname,passw);
                    at.interrupt();
                    if (error == 0){
                        finish();
                    }
                }else{
                    Toast to3 = Toast.makeText(this, "Passwords do not match or is empty.", Toast.LENGTH_SHORT);
                    to3.show();
                }
        }else{
            Toast to1 = Toast.makeText(this, "Name is empty.", Toast.LENGTH_SHORT);
            to1.show();
        }
    }else{
        Toast to2 = Toast.makeText(this, "UserName is empty.", Toast.LENGTH_SHORT);
        to2.show();
    }
}
```

El método onCreate() de esta activity es más o menos igual que el de la anterior, asignamos los layouts a las variables correspondientes e inicializamos los onClickListener(). En el método onClick() comprobamos que los campos no estén vacíos y que los dos campos de contraseña sean iguales. Si todo lo anterior es correcto se hace una petición al servidor para crear un nuevo usuario.

Menu

El menú es un activity donde podemos elegir y navegar por las diferentes opciones que nos ofrece la app ya sea el mapa, friends (lista de amigos), profile (perfil del usuario) o los ajustes. En la action bar podemos ver las peticiones de amistad y también hacer logout si preferimos entrar con otro usuario.

main.xml

El layout se compone de 4 ImageView para formar los botones y 4 EditText para los títulos de los botones. Además está la action bar que está implementada gracias a un paquete de código libre denominado Sherlock ActionBar (para más información sobre esta librería http://www.actionbarsherlock.com) gracias al cual conseguimos compatibilidad de actionbar en todas las versiones de Android.

lttdprojectActivity.java

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //Initialize the action bar and the classes
    setContentView(R.layout.main);

    ActionBar ab = getSupportActionBar();
    ab.setHomeButtonEnabled(true);
    ab.setTitle("FriendScout");
    ab.setIcon(R.drawable.logo1);

    Sc = new ServerConnect(this);
    sph = new SpHelper(this);

    frnumber = Sc.tryRequestNFriends(sph.getUser());

    //Call the auxiliary method to initialize the location manager
    initializeLM();

}
```

En el código del menú superior se inicializa el layout y además también se llama a los métodos de creación de la ActionBar. Una vez creada la barra con la clase ServerConnect (explicada más adelante) se llama al método oportuno para comprobar si hay alguna petición de amistad pendiente.

```
//Auxiliary method to initialize the location manager. Position will be updated after
//5 minutes or 500 meters
private void initializeLM() {
    // TODO Auto-generated method stub
    lm = (LocationManager) getSystemService(LOCATION_SERVICE);

    // check if the location manager is working properly. Then if GPS is on
    // use it as the data provider. If not try to use the network. If both
    // are deactivated show a text to inform the user.
    if (lm == null) {
        Toast.makeText(this, "Service not avalible", Toast.LENGTH_SHORT)
                .show();
        return;
    }
    if (lm.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 300000,
                500f, this);

    } else if (lm.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {

        lm.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 300000,
                500f, this);

    } else {

        Toast.makeText(this, "Network and gps deactivated",
                Toast.LENGTH_SHORT).show();

    }
}
```

En el siguiente fragmento de código se muestra el método que inicializa el location manager para GPS y, si no puede, por la red móvil, además de hacer actualizaciones de estado de tanto en tanto, en este caso cada 5 minutos o 500 metros.

```
public void onLocationChanged(Location location) {
    // TODO Auto-generated method stub
    if (location != null) {
        Double latitude = location.getLatitude();
        Double longitude = location.getLongitude();
        currentLt = latitude;
        currentLn = longitude;
        Sc.tryUpdating(sph.getUser(), latitude, longitude);
    }

}
```

El método siguiente se dispara cuando la especificación requerida en el anterior (cuando pasan 5 minutos o 500 metros) se completa y se actualiza la posición actual mediante el método tryUpdating() de la clase ServerConnect.

```
//Method used to update the friend requests number. This method is called after an
//invalidateOptionsMenu().
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub

    menu.clear();
    MenuItem one = menu.add(0, 1, 0, "REQUEST(" + frnumber + ")");
    MenuItem two = menu.add(0, 2, 1, "LOGOUT");
    one.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
    two.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
    Log.i("log_tag", "NUMERO DE FRIENDREQUEST  " + frnumber);

    return super.onPrepareOptionsMenu(menu);
}

//Initialize action bar
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuItem one = menu.add(0, 1, 0, "REQUEST(" + frnumber + ")");
    MenuItem two = menu.add(0, 2, 1, "LOGOUT");

    one.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
    two.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
    return true;
}
```

De estos dos métodos que se muestran, en el segundo, onCreateOptionsMenu() se inicializa la ActionBar con los botones correspondientes y con el número de peticiones de amistad obtenido previamente. El primer método es el que actualiza la barra cada vez que se vuelve a la activity del menú, ya que podría darse el caso de que ocurriera una petición de amistad mientras se está mirando el mapa.

```
//Manage action bar interaction
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection

    switch (item.getItemId()) {
    case 1:
        if (frnumber != 0) {
            Toast to = Toast.makeText(this, "FRIEND REQUESTS",
                    Toast.LENGTH_SHORT);
            to.show();
            finish = false;
            startActivity(new Intent(getApplicationContext(),
                    FriendRequest.class));
        } else {
            Toast to = Toast.makeText(this, "YOU HAVE NO FRIEND REQUESTS",
                    Toast.LENGTH_SHORT);
            to.show();
        }
        return true;
    case 2:
        Toast to1 = Toast.makeText(this, "LOGOUT", Toast.LENGTH_SHORT);
        to1.show();
        confirmExit();
        return true;
    default:
        Toast to11 = Toast.makeText(this, "HOME", Toast.LENGTH_SHORT);
        to11.show();
        return true;
    }
}
```

Éste es el método que tramita las interacciones con la ActionBar cuando se clica en cualquiera de los botones.

```
//Manage dashboard interaction
public void onClickFeature(View v) {
    int id = v.getId();
    switch (id) {
    case R.id.map_btn:
        finish = false;
        startActivity(new Intent(getApplicationContext(),
                MapPrincipal.class));
        break;
    case R.id.friends_btn:
        finish = false;
        Intent iF = new Intent(getApplicationContext(), FriendList.class);
        iF.putExtra("latitude", currentLt);
        iF.putExtra("longitude", currentLn);

        startActivity(iF);
        break;
    case R.id.profile_btn:
        finish = false;
        startActivity(new Intent(getApplicationContext(), Profile.class));
        break;
    case R.id.settings_btn:
        finish = false;
        startActivity(new Intent(getApplicationContext(), Settings.class));

        break;
    default:
        break;
    }
}
```

Éste es el método que gestiona las acciones con los botones del menú como abrir el mapa, los amigos, el perfil o las opciones.

Request

En este caso no hay xml para definir el layout ya que se crea de manera activa en tiempo de ejecución.

RequestFriend.java

```
//method to create the list of current friend petitions
private void showList(){

    pd = ProgressDialog.show(this, "","Getting friend list...", true);
    AThread at = new AThread(pd);
    at.start();
    friendArray = Sc.tryRequestFriends(sph.getUser());
    Boolean asdf = (friendArray[0]==null);
    Log.i("log_tag",asdf.toString());
    at.interrupt();
    if(asdf){
        Log.i("log_tag","This activity will finish");
        finish();
    }else{
        Log.i("log_tag","Here fails");
        setListAdapter(new ArrayAdapter<String>(this, R.layout.list_item, friendArray ));
    //at.interrupt();
    }
}
```

Es una clase que se puede resumir en este método, que recupera del servidor la lista de las peticiones y te las muestra en una lista.

Map

map.xml

El layout del mapa se compone de un mapview. Para que el mapa pueda ser visto tienes que registrar el ordenador desde el que estés trabajando y obtener la clave de testeo y, en el caso de querer hacer la versión release de tu aplicación y subirla a Google Play, se debe adquirir una nueva clave que sólo sirve para este propósito.

MapPrincipal.java

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

    //Initialize classes
    initialize();
    sph = new SpHelper(this);
    Sc = new ServerConnect(this);
    mo = new MapOverlay(this,sph.getUser(),"Me",0);

    intializeLM();
    fixClassLoaderIssue();

}
```

Este método inicializa algunas clases e invoca a los métodos necesarios para que el mapa quede inicializado con sus overlays donde se dibuja la posición actual y también donde se encuentran los usuarios que son amigos.

```
//method to initialize the required variables
private void initialize(){

    setContentView(R.layout.map);
    //set needed variables
    map = (MapView)findViewById(R.id.mvPrincipal);
    bCenter = (Button)findViewById(R.id.bCenter);
    bCenter.setOnClickListener(this);

    map.setBuiltInZoomControls(true);
    map.setSatellite(false);
    mc = map.getController();
    mc.setZoom(15);

    //set the location manager

}
```

Este método inicializa el mapa además de los botones, como el botón de centrarse sobre tu posición y los botones de zoom.

```
//Auxiliary class to show center button
public class TouchHelper extends Overlay {
    private long downTime, upTime;
    public boolean onTouchEvent(MotionEvent e, MapView m){
        if(e.getAction() == MotionEvent.ACTION_DOWN){
            downTime = e.getEventTime();
        }
        if(e.getAction() == MotionEvent.ACTION_UP){
            upTime = e.getEventTime();
        }
        if(upTime-downTime > 10){
            animate = false;
            bCenter.setVisibility(View.VISIBLE);
            return true;
        }
        return false;
    }
}

//on click method for the center button
public void onClick(View arg0) {
    // TODO Auto-generated method stub
    animate = true;
    bCenter.setVisibility(View.GONE);
    mc.animateTo(current);
}
```

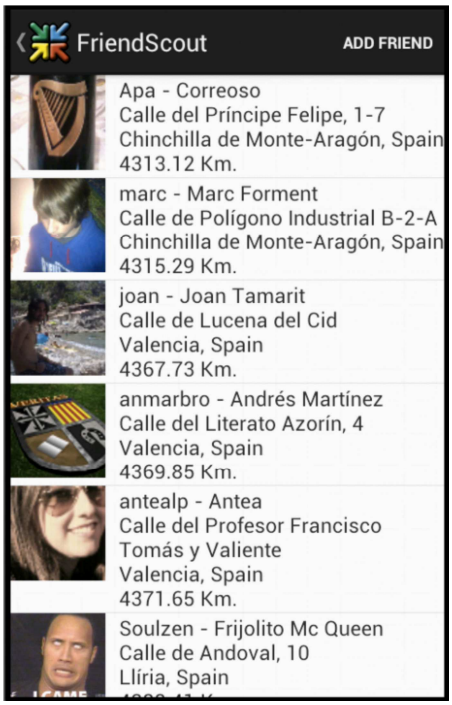
La clase mostrada es una clase que está dentro de la del mapa, y es un overlay que gestiona el botón, tanto cuando se ha de mostrar como cuando no, y también gestiona los eventos cuando éste es clicado.

```
//When the location is changed update the map overlays
public void onLocationChanged(Location arg0) {
    // TODO Auto-generated method stub
    if(arg0!=null){
        double latitude = arg0.getLatitude();
        double longitude = arg0.getLongitude();

        current = new GeoPoint((int) (latitude * 1E6), (int) (longitude * 1E6));
        mo.setPoint(current);
        MapOverlay mo1;
        ArrayList<Friend> Afriends = Sc.getServerData(latitude, longitude);

        DisplayMetrics metrics = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(metrics);
        Log.i("log_tag","Density pixel"+metrics.density);
        int i =0;
        for(Friend f : Afriends){
            if(i==sph.getNumber()){break;}
            mo1 = new MapOverlay(this,f.getUsername(),f.getName(),metrics.density);
            mo1.setPoint(new GeoPoint((int) (f.getLatitude() * 1E6), (int) (f.getLongitude() * 1E6)));
            AfriendsMo.add(mo1);
            i++;
        }

        if(animate){
            mc.animateTo(current);
        }
        List<Overlay> overlayList = map.getOverlays();
        overlayList.clear();
        for(MapOverlay mo2 : AfriendsMo){
            overlayList.add(mo2);
        }
        overlayList.add(mo);
        overlayList.add(th);


    }
}
```

Este método es como el ya explicado anteriormente al menú (no se muestra el método que inicializa el location manager ya que es exactamente igual en las dos clases). Se dispara cuando hay algún tipo de variación, ya sea temporal o espacial, de tu posición actual. En este caso el método tiene la función de actualizar los overlays donde estáis tú y tus amigos representados por si ha habido algún tipo de cambio.

Friends

friendlist.xml, item.xml



Realmente el layout de friends es una composición de dos layouts. El primero es una lista del segundo y así, el segundo, está compuesto de un ImageView y de un TextView para el avatar del amigo y para la información, respectivamente.

La barra superior ya ha sido explicada y siempre se declara de la misma forma, aunque en este caso sólo tiene un botón que abre un cuadro de diálogo para agregar un nuevo amigo.

FriendList.java

```
@Override
public void onCreate(Bundle savedInstanceState) {

    //Initialize the components
    super.onCreate(savedInstanceState);
    setContentView(R.layout.lista);

    sph = new SpHelper(this);
    Sc = new ServerConnect(this);

    ActionBar ab = getSupportActionBar();
    ab.setDisplayHomeAsUpEnabled(true);
    ab.setTitle("FriendScout");
    ab.setIcon(R.drawable.logo1);

    currentPos = getIntent().getExtras();

    //Show the list on the screen
    showList();

}
```

Este método, como ya hemos visto, es el que se ejecuta al crearse la activity. Lo único que tiene de especial es que invoca al método showList() que mostrará la lista de amigos.

```
//Method to create an ArrayList of friends
public String[] friendsToArray(ArrayList<Friend> friends)
{
    String[] res = new String[friends.size()];
    int i = 0;
    for(Friend f: friends)
    {
        res[i]=f.toString();
        i++;
    }
    return res;
}
```

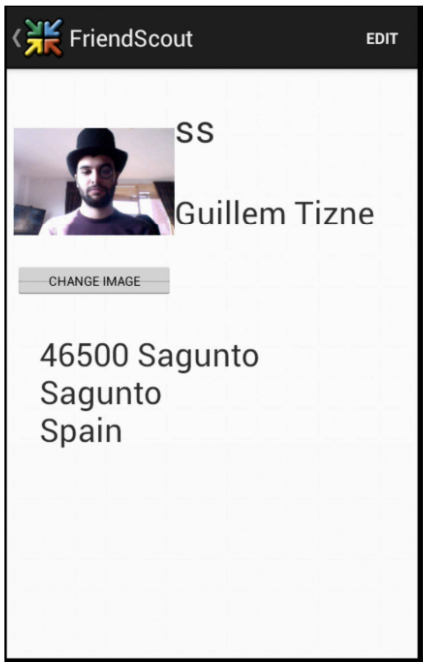
Este método recorre un ArrayList de la clase Friend que veremos más adelante y crea un array de strings con la información del amigo respecto al usuario como puede ser la distancia entre los dos, la calle en la que se encuentra, etc.

```
//Method used to  create and show the list
private void showList(){

    //Get the necessary information
    pd = ProgressDialog.show(this, "","Getting friend list...", true);
    AThread at = new AThread(pd);
    at.start();

    friendArray = friendsToArray(Sc.getServerData(currentPos.getDouble("latitude"),currentPos.getDouble("longitude")));
    String[] imageString = new String[friendArray.length];
    Scanner sc;

    for(int i=0;i<friendArray.length;i++){
        sc = new Scanner(friendArray[i]);
        String aux = sc.next();
        imageString[i] = "http://www.andguimar.com/images/"+aux+".png";
        Log.i("log_tag", imageString[i]);
    }

    //Use the LazyAdapter to format the list
    list=(ListView)findViewById(R.id.list);
    adapter=new LazyAdapter(this, imageString,friendArray);
    list.setAdapter(adapter);

    //onClickListener to manage the clicks on list items
    list.setOnItemClickListener(new OnItemClickListener(){
        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,long arg3){
            // TODO Auto-generated method stub
            Scanner sc = new Scanner(friendArray[arg2]);
            String aux = sc.next();
            deleteFriendDialog(aux);
        }
    });
    at.interrupt();
}
```

El método showList() básicamente crea un array con tus amigos gracias al método friendsToArray() explicado anteriormente. Después crea la url donde estará la imagen del avatar y utiliza unas clases del paquete LazyList (para más información sobre esta librería http://github.com/thest1/LazyList) para recuperar la imagen de la url y posteriormente crear la lista con la imagen y la correspondiente información sobre el amigo. También crea un disparador para gestionar el clic sobre algún objeto de la lista.

Profile

profile.xml



El layout del profile consta de la barra, como el menú, y la lista de amigos. En este caso tiene un botón que te lleva a una nueva activity donde se modifica la información de tu perfil.

El layout en sí se compone de un ImageView para el avatar, un botón para cambiar la imagen y 3 TextView para el resto de la información (el nombre de usuario, el nombre y la posición actual).

Profile.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.profile);

    //initialize action bar and classes
    iL = new ImageLoader(this);
    sph = new SpHelper(this);
    Sc = new ServerConnect(this);

    ActionBar ab = getSupportActionBar();
    ab.setDisplayHomeAsUpEnabled(true);
    ab.setTitle("FriendScout");
    ab.setIcon(R.drawable.logo1);

    usernameTV = (TextView) findViewById(R.id.txtVProf1);
    nameTV = (TextView) findViewById(R.id.txtVProf2);
    infoTV = (TextView) findViewById(R.id.txtVProf3);
    iView = (ImageView) findViewById(R.id.iProfile);

    String url = "http://www.andguimar.com/images/" + sph.getUser()
            + ".png";
    Log.i("log_tag", url);
    iL.DisplayImage(url, iView);

    Sc.getProfileInfo(usernameTV, nameTV, infoTV);
    Button b = (Button) findViewById(R.id.bProfile);
    b.setOnClickListener(listener);
}
```

Este método es el que se ejecuta al iniciar la activity. Respecto a los anteriores no cambia demasiado, simplemente se puede observar que la imagen la recupera del servidor mediante una de las clases del paquete LazyList.

```
//Listener for the profile edit button
public OnClickListener listener = new OnClickListener() {
    @Override
    public void onClick(View arg0) {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(
                Intent.createChooser(intent, "Select Picture"),
                SELECT_PICTURE);
    }
};
```

Éste es el Listener del botón de editar la imagen donde, en el caso de ser pulsado, se abrirá la galería de imágenes del dispositivo para seleccionar una imagen.

```
//after the intent to get a picture get the image path to call the image upload
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK) {
        if (requestCode == SELECT_PICTURE) {

            Uri selectedImageUri = data.getData();
            if (selectedImageUri.getPathSegments().get(0).equals("mnt")) {
                selectedImagePath = selectedImageUri.getPath();
                Log.i("log_tag", selectedImagePath);
                imageValidation(selectedImagePath);
            } else {
                selectedImagePath = getPath(selectedImageUri);
                Log.i("log_tag", selectedImagePath);
                imageValidation(selectedImagePath);

            }
        }
    }
}

//method to get a path from an uri (when the image is from the gallery)
public String getPath(Uri uri) {
    String[] projection = { MediaStore.Images.Media.DATA };
    Cursor cursor = managedQuery(uri, projection, null, null, null);
    int column_index = cursor
            .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
    cursor.moveToFirst();
    return cursor.getString(column_index);
}
```

Estos dos métodos gestionan el resultado del intent que se ejecuta en el Listener anterior. Verifican el path de la imagen y la gestionan de una forma u otra ya que es posible que, si tienes más de una galería de imágenes en el dispositivo, te dé a elegir entre ellas y no siempre el resultado de éstas tiene el mismo formato. Estos métodos formatean éste resultado para que sea igual. ImageValidation() simplemente pregunta si es la imagen correcta la que has seleccionado.

```
//method to rescale the image  and then upload it to the server
private void updateImage() {

    Bitmap bitmapOrg = BitmapFactory.decodeFile(selectedImagePath);
    if (bitmapOrg.getWidth() > bitmapOrg.getHeight()) {
        float prop = (float) bitmapOrg.getWidth()/ (float) bitmapOrg.getHeight();
        int newHeight = (int) (200 / prop);
        bitmapOrg = Bitmap.createScaledBitmap(bitmapOrg, 200, newHeight,
                false);
    } else {
        float prop = (float) bitmapOrg.getHeight()/ (float) bitmapOrg.getWidth();
        int newWidth = (int) (200 / prop);
        bitmapOrg = Bitmap.createScaledBitmap(bitmapOrg, newWidth, 200,
                false);
    }

    InputStream is;

    ByteArrayOutputStream bao = new ByteArrayOutputStream();
    bitmapOrg.compress(Bitmap.CompressFormat.PNG, 90, bao);
    byte[] ba = bao.toByteArray();
    String ba1 = Base64.encodeBytes(ba);
    ArrayList<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();
    nameValuePairs.add(new BasicNameValuePair("image", ba1));
    nameValuePairs.add(new BasicNameValuePair("username", sph.getUser()));
    try {
        HttpClient httpclient = new DefaultHttpClient();
        HttpPost httppost = new HttpPost(
                "http://nikomac.uk.cloudlogin.co/images/upload.php");
        httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
        HttpResponse response = httpclient.execute(httppost);
        HttpEntity entity = response.getEntity();
        is = entity.getContent();
        Log.i("log_tag", is.toString());
    } catch (Exception e) {
        Log.e("log_tag", "Error in http connection " + e.toString());

    }
    String url = "http://www.andguimar.com/images/" + sph.getUser()
            + ".png";
    Log.i("log_tag", url);
    iL.clearCache();
    iL.DisplayImage(url, iView);

}
```

Este método que es tan grande hace principalmente dos cosas: reescala la imagen (ya que podría ser demasiado grande) hasta un tamaño razonable y eficiente y, seguidamente, la sube al servidor con el nuevo nombre de usuario. Si ya había una imagen anteriormente, la nueva sobrescribe a la anterior.

Edit Profile

editprofile.xml

El layout del editprofile es un layout muy común, tal y como ya hemos visto anteriormente en el login, pues está compuesto por un TextView donde está el nombre de la activity (Edit Profile) y 3 EditText para cambiar el nombre y la contraseña que, por motivos de seguridad, ha de ser escrita dos veces para poder ser modificada. Finalmente un botón para hacer efectivos todos los cambios.

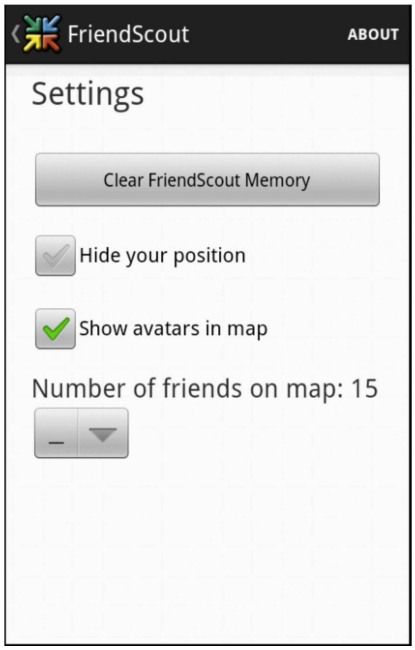
También incluye la ActionBar pero ésta sin botones.

EditProfile.java

Ésta también es una activity bastante sencilla donde simplemente está el gestor del botón como ya hemos mostrado en otras clases. Si ha habido cambios y éstos son correctos, serán subidos al servidor y se modificará la base de datos.

Settings

settings.xml



Este layout se compone de la barra (como casi todos) donde hay un botón que muestra los creadores de la aplicación, un TextView que muestra la palabra Settings, un botón para limpiar la memoria de la aplicación (básicamente borra las imágenes guardadas de los amigos), dos checkbox para diferentes opciones de la aplicación y finalmente una DropDownList para seleccionar el número de personas que se mostrarán en el mapa.

settings.java

Básicamente, en esta clase, todos los métodos son gestores de los distintos tipos de botones que hay en el layout además de guardar cada modificación que se haga en las Shared preferences, que son las variables permanentes de la aplicación, para que perduren aunque se cierre la aplicación.

Las clases que vienen a continuación no tienen layout.

AThread

AThread.java

```
//A simple thread to manage progress dialogs. That waits
package com.project;

import android.app.ProgressDialog;

public class AThread extends Thread implements Runnable {

    ProgressDialog pd;


    public AThread(ProgressDialog pd){
        this.pd= pd;
    }

    public void run(){
         try {
             synchronized(this) {
                 wait();
             }
          }
        catch (Exception e) {
            Log.e("tag",e.getMessage());
        }
        pd.dismiss();
     }
    public void interrupt(){
        synchronized(this) {
            notify();
        }
    }
}
```

Esta clase es una extensión de la clase Thread donde se crea un sencillo hilo de ejecución para gestionar un progress dialog. Cuando el hilo es ejecutado se espera hasta que el hilo padre le interrumpe, y entonces el progress dialog se cierra.

Friend

Friend.java

```
//Constructor. For the distance we use the method from Location and the coordinates from
//the user and his friend
public Friend(String name, String username,double latitude, double longitude,
        double latitudeF, double longitudeF, Context con)
{
    this.name = name;
    this.username = username;
    this.latitude = latitude;
    this.longitude = longitude;
    float[] results = new float[4];
    Location.distanceBetween(latitude, longitude, latitudeF, longitudeF, results);
    this.distanceToF = results[0];
    if(this.distanceToF>999){
        this.dist=String.format("%.2f", results[0]/1000);
        this.dist=this.dist+" Km.";
    }else{
        this.dist=String.format("%.0f", results[0]);
        this.dist=this.dist+" m.";
    }
    this.con = con;
}
```

Esta clase es una clase que crea un objeto de tipo Friend donde se almacena información sobre un amigo para su posterior tramitación. Éste trozo de código corresponde al constructor de la clase donde también se calcula la distancia entre el amigo y el usuario.

La clase también tiene implementados todos los get y set necesarios.

```
//toString method
public String toString()
{
    Geocoder gc = new Geocoder(con);
    try {
        List<Address> addr = gc.getFromLocation(latitude, longitude, 1);
        Address a = addr.get(0);
        return username+" - "+ name +"\n"+ a.getAddressLine(0) + "\n"+ a.getLocality() +", "
                + a.getCountryName() + "\n" + dist;
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
        return (username+" - "+ name +"\n"+"Unknown");

}

//Override compareTo to use the sort method from Collections
@Override
public int compareTo(Friend f) {
    // TODO Auto-generated method stub

    return (int)(this.getDistanceToF()-f.getDistanceToF());
}
```

El código anterior pertenece a los métodos toString() donde, a partir de la latitud y la longitud, extraemos con el Geocoder la dirección con la ciudad, la región e incluso la calle donde se encuentra el amigo, y además su nombre de usuario y nombre, que son introducidos en una string.

También tiene implementado un compareTo() para comparar la distancia entre dos amigos y así poder hacer una ordenación por distancia.

MapOverlay

MapOverlay.java

```
public MapOverlay(Context c,String user,String name,float density){
    //instances of the auxiliary classes that we need
    sph = new SpHelper(c);
    iL = new ImageLoader(c);

    //We check if the overlay that we are creating is not for the current logged user
    //to load the image from the server
    if((!(user.equals(sph.getUser())))&&sph.getAvatar()){
        btmp = iL.getImage("http://www.andguimar.com/images/" + user + ".png");
        if(btmp!=null){
            btmp = Bitmap.createScaledBitmap(btmp, (int)(37*density),(int)(37*density), false);
        }
    }
    this.currentContext = c;
    this.user = user;
    this.name = name;
    this.density = density;
}
```

La clase MapOverlay es la que se encarga de crear las capas del mapa donde están los avatares y los nombres de usuario de los amigos dibujados, y donde también está el logo de la aplicación que indica tu posición.

Este código corresponde al constructor donde se comprueba si el overlay será para el usuario o para un amigo. Si es para un amigo descargará la imagen del servidor y, si es para el usuario, simplemente cogerá el logo de los recursos.

El siguiente método es el draw() del overlay que se encarga de dibujar en el mapa, en la posición correspondiente, el usuario con el avatar y el nombre o simplemente el nombre si en el apartado de configuración está desmarcada la opción de mostrar avatares en el mapa.

```java
//Draw method
@Override
public boolean draw(Canvas canvas, MapView mapView, boolean shadow,
        long when) {
    super.draw(canvas, mapView, shadow);

    Point screenPts = new Point();
    mapView.getProjection().toPixels(pointD, screenPts);
    //If the overlay is for the current user we create a overlay with the logo
    if(user.equals(sph.getUser())){
        Bitmap bmp = BitmapFactory.decodeResource(currentContext.getResources(), R.drawable.logo);
        canvas.drawBitmap(bmp, screenPts.x-15, screenPts.y-15 , null);
    //If the overlay is for a friend we create an overlay with the image that we loaded from the server
    //on the constructor, his name and a red point
    }else{
        if(!(pointD.getLatitudeE6()==0&&pointD.getLongitudeE6()==0)){
            Scanner sc = new Scanner(name);
            String aux = sc.next();
            Paint paint = new Paint();
            if(btmp!=null){
                canvas.drawBitmap(btmp, screenPts.x, screenPts.y , null);
            }
            paint.setColor(Color.RED);
            canvas.drawCircle(screenPts.x, screenPts.y, 6*density, paint);
            paint.setColor(Color.BLACK);
            paint.setTextSize(16*density);
            paint.setShadowLayer(4, 0, 0, Color.YELLOW);
            canvas.drawText(aux,screenPts.x+6*density, screenPts.y, paint);
        }
    }
    return true;
}
```

ServerConnect

ServerConnect.java

```java
//Url for php files used to make the mysql querys
public static final String KEY_1 = "http://nikomac.uk.cloudlogin.co/login.php";
public static final String KEY_2 = "http://nikomac.uk.cloudlogin.co/signin.php";
public static final String KEY_3 = "http://nikomac.uk.cloudlogin.co/updateposition.php";
public static final String KEY_4 = "http://nikomac.uk.cloudlogin.co/friendrequest.php";
public static final String KEY_5 = "http://nikomac.uk.cloudlogin.co/getfriends.php";
public static final String KEY_6 = "http://nikomac.uk.cloudlogin.co/erasefriend.php";
public static final String KEY_7 = "http://nikomac.uk.cloudlogin.co/addfriend.php";
public static final String KEY_8 = "http://nikomac.uk.cloudlogin.co/acceptfriend.php";
public static final String KEY_9 = "http://nikomac.uk.cloudlogin.co/profileinfo.php";
public static final String KEY_10 = "http://nikomac.uk.cloudlogin.co/editpassword.php";
public static final String KEY_11 = "http://nikomac.uk.cloudlogin.co/editname.php";
```

La clase ServerConnect es la clase que hace todas las conexiones con el servidor. Tenemos todas las direcciones a los PHP a los que podemos conectarnos en variables para su posterior uso en la clase.

```
//method to get the friend list
public ArrayList<Friend> getServerData(double latitude, double longitude) {

    InputStream is = null;
    ArrayList<Friend> returnArray = new ArrayList<Friend>();
    String result = "";

    ArrayList<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();
    nameValuePairs
            .add(new BasicNameValuePair("serverpassword", "morvedre"));
    nameValuePairs.add(new BasicNameValuePair("username", sph.getUser()));

    try {
        HttpClient httpclient = new DefaultHttpClient();
        HttpPost httppost = new HttpPost(KEY_5);
        httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
        HttpResponse response = httpclient.execute(httppost);
        HttpEntity entity = response.getEntity();
        is = entity.getContent();

    } catch (Exception e) {
        Log.e("log_tag", "Error in http connection " + e.toString());
    }

    // convert response to string
    try {
        BufferedReader reader = new BufferedReader(new InputStreamReader(
                is, "iso-8859-1"), 8);
        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
        is.close();
        result = sb.toString();

    } catch (Exception e) {
        Log.e("log_tag", "Error converting result " + e.toString());
    }
```

Todos los métodos de acceso al servidor son similares así que se mostrará cómo funciona uno de ellos mientras que el resto se pueden ver en el documento adjunto con el código fuente. Casi todos los métodos pueden ser divididos en tres partes: primero la preparación de los valores que se enviarán al servidor como value pairs tal y como ocurre en las webs por las urls y la conexión y envío de los valores, el segundo la recepción de respuesta del servidor y el almacenamiento de ésta en una string, y la tercera la descodificación y almacenamiento ordenado de los valores devueltos por el PHP que están en formato JSON.

```
        // parse json data
        try {
            // returnString = "";
            JSONArray jArray = new JSONArray(result);
            for (int i = 0; i < jArray.length(); i++) {
                JSONObject json_data = jArray.getJSONObject(i);

                returnArray.add(new Friend(json_data.getString("name"),
                        json_data.getString("username"), json_data
                                .getDouble("latitude"), json_data
                                .getDouble("longitude"), latitude, longitude,
                        context));
                Log.i("log_tag", "username " + json_data.getString("username")
                        + "name " + json_data.getString("name"));
            }
        } catch (JSONException e) {
            Log.e("log_tag", "Error parsing data " + e.toString());
        }
        Collections.sort(returnArray);
        return returnArray;
    }
```

SpHelper

SpHelper.java

```
    //Constructor
    public SpHelper(Context context){

        sp = context.getSharedPreferences("settings", 0);
        editor = sp.edit();
    }

    //Check if there is a saved user name
    public boolean checkSavedUser() {
        // TODO Auto-generated method stub
        String user = sp.getString("user","noUserDefined");

        if(!(user.equals("noUserDefined"))){

            return true;
        }
        return false;
    }
```

Esta clase es la encargada de gestionar las Shared preferences que son unos valores que perduran hasta que son borrados de forma manual, lo que quiere decir que perdurarán aunque cerremos la aplicación.

El primer método que se observa es el constructor. El segundo es un método que se utiliza en el login.java para saber si el usuario ya tenía una sesión abierta de nuestra aplicación.

En la clase también se encuentran los get y set necesarios para poder configurar las variables desde otras clases.

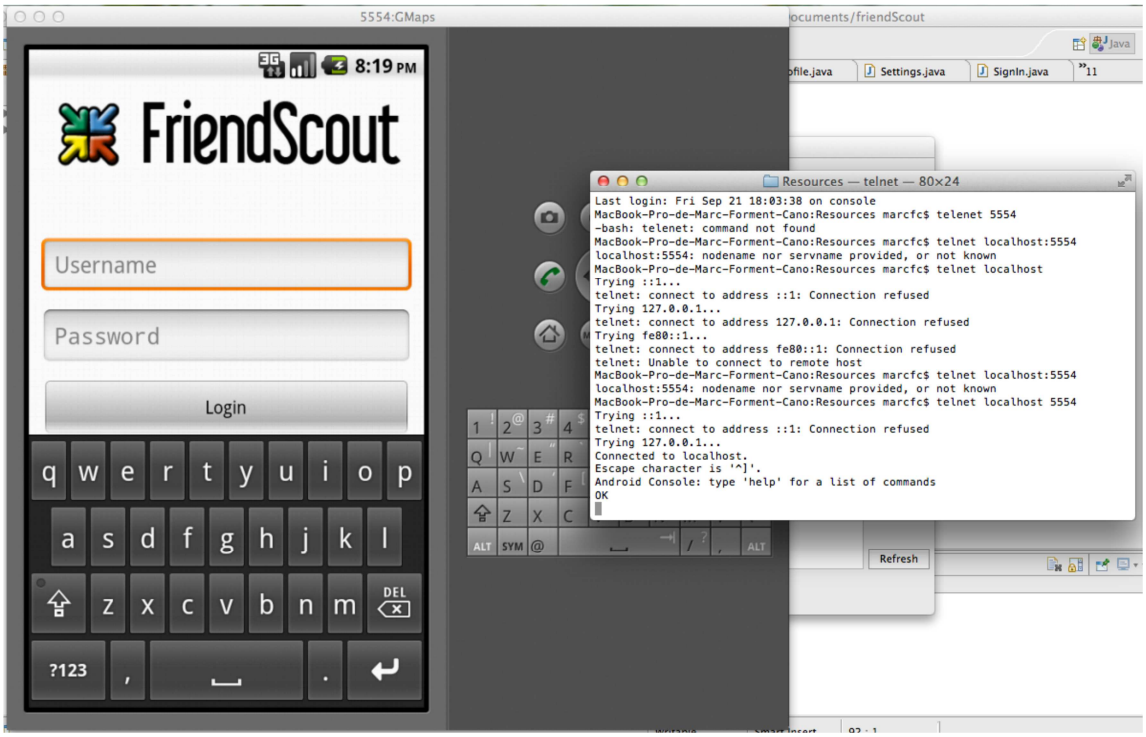
```
//Method to delete a saved user name (on logout)
public void cleanUser() {
    // TODO Auto-generated method stub
    editor.putBoolean("avatar", true);
    editor.putBoolean("visibility", false);
    editor.putString("user", "noUserDefined");
    editor.putInt("number", 15);

    editor.commit();

}
```

Además del método cleanUser() que se utiliza para dejar las variables tal y como estaban inicialmente, este método se utiliza cuando se hace logout de la aplicación.

Hay algunas clases que no han sido descritas. No lo hemos hecho porque ya estaban hechas al formar parte de clases de código libre descargadas y utilizadas sin haber sido modificadas. Se ha hecho referencia a ellas en los respectivos apartados (Sherlock ActionBar y LazyList).

6. Pruebas

En cuanto a las pruebas, primeramente comenzamos con el emulador que proporciona el SDK de Google. Mientras la aplicación no empieza a tener un tamaño mayor y unas tareas que requieren más capacidad de cálculo, ésta es una opción bastante interesante. Una de las características más interesantes del emulador es que puedes probar la mayoría de las versiones de Android disponibles (en un principio no teníamos disponibles dispositivos con todas las principales versiones del sistema operativo).



Además, la simulación básica de coordenadas es muy asequible a través de telnet. Por otro lado, la perspectiva DDMS en Eclipse permite una simulación de rutas bastante útil que para hacer pruebas mientras estás trabajando es incluso más útil que la simulación en el dispositivo.

Al poco tiempo de trabajar con el emulador empezamos a hacer tests en dispositivos. Primero con los nuestros, que desgraciadamente utilizan la misma versión (2.3) como se indicó en el apartado de equipo utilizado. Seguidamente probamos en otras versiones de dispositivos a través de amigos y de la Universidad, y fue a través de un amigo donde descubrimos la importancia de este tipo de pruebas multiplataforma. Descubrimos que las conexiones de red debían hacerse mediante hilos a partir de la versión 3.0 (el dispositivo en concreto funcionaba con la 4.0.3) y aparentemente la aplicación continuaba funcionando ya que no se producía ningún crash.

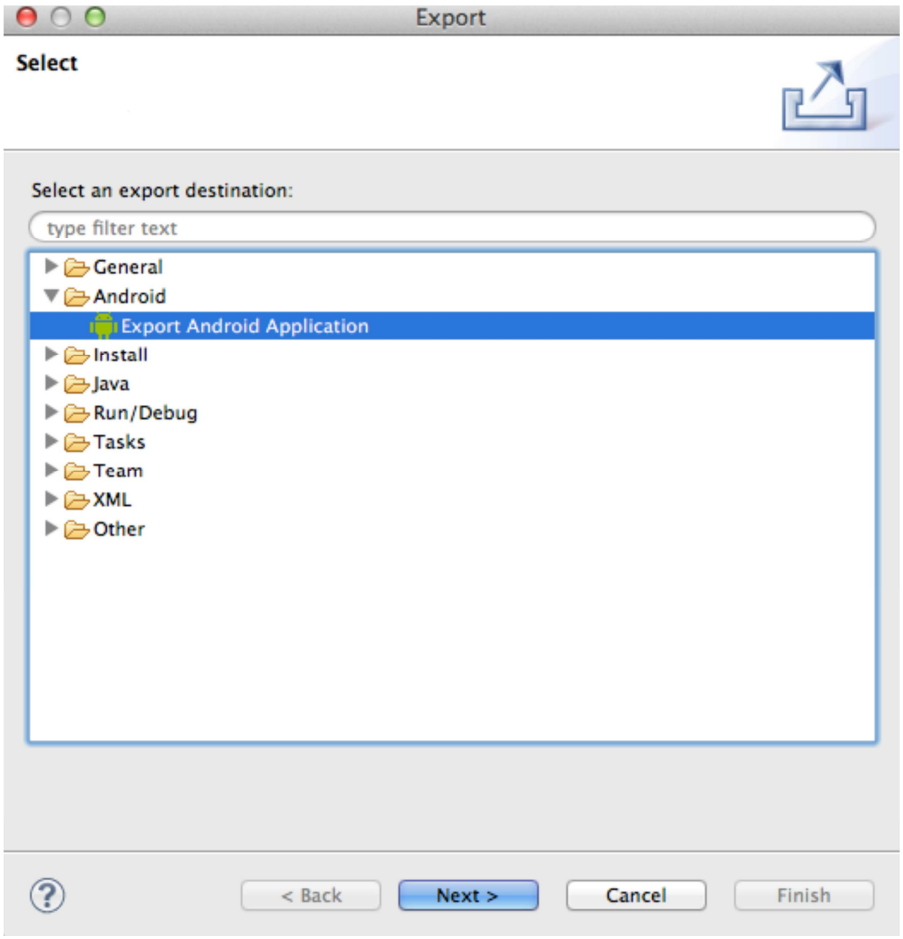
7. Publicación en la Play Store

La publicación en la Play Store de Google comprende diversos pasos que describiremos a continuación. Principalmente consta de cuatro pasos:

- Obtención de la clave para firmar el apk.
- Obtención de la clave de Google Maps.
- Creación de la cuenta de Android Developer.
- Subir el apk firmado a la Play Store a través de la cuenta de Developer.

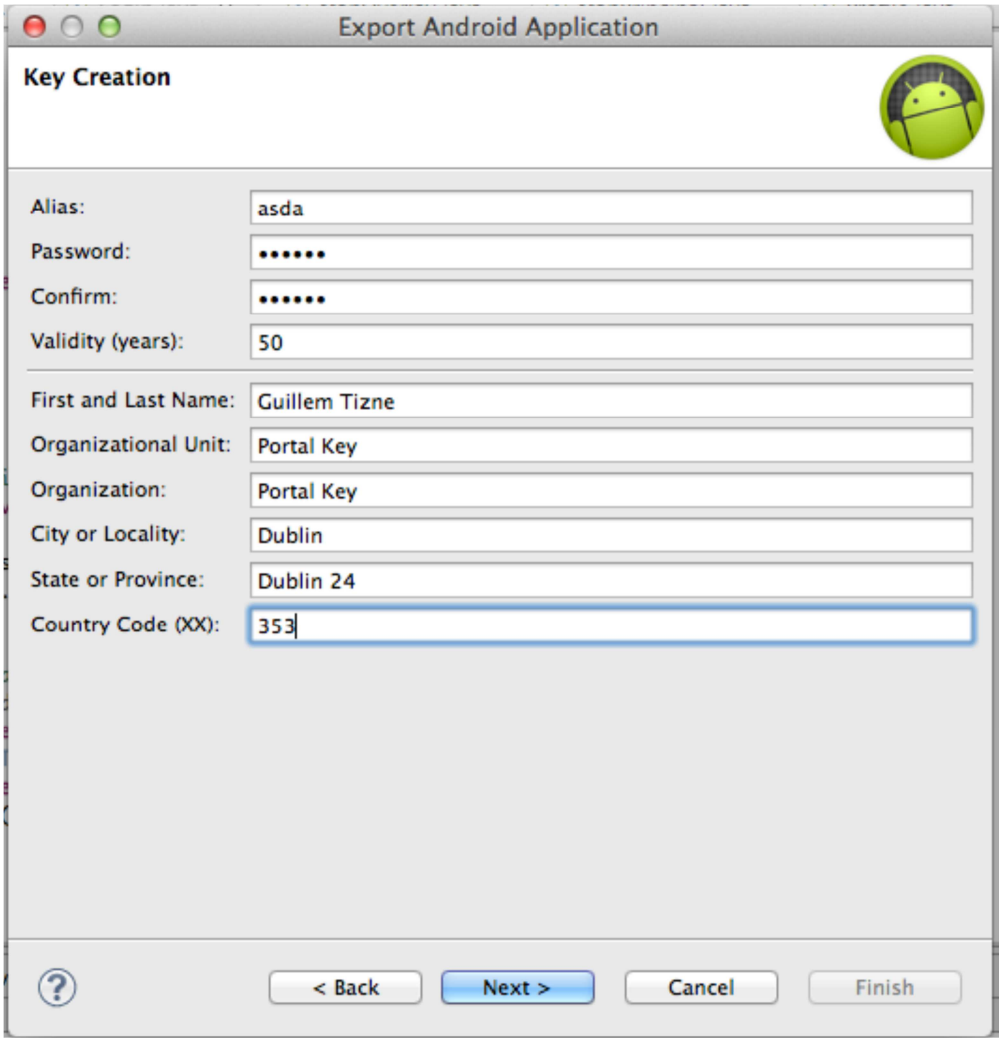
El primer paso es el más sencillo ya que el SDK proporciona un asistente para completar este proceso.

Así que primeramente seleccionamos File > Export > Android > Export Android Application.



Seguidamente seleccionamos el proyecto que queremos firmar e indicamos que queremos crear una keystore. Indicamos el nombre y la contraseña que queremos utilizar para ésta y clicamos Next.

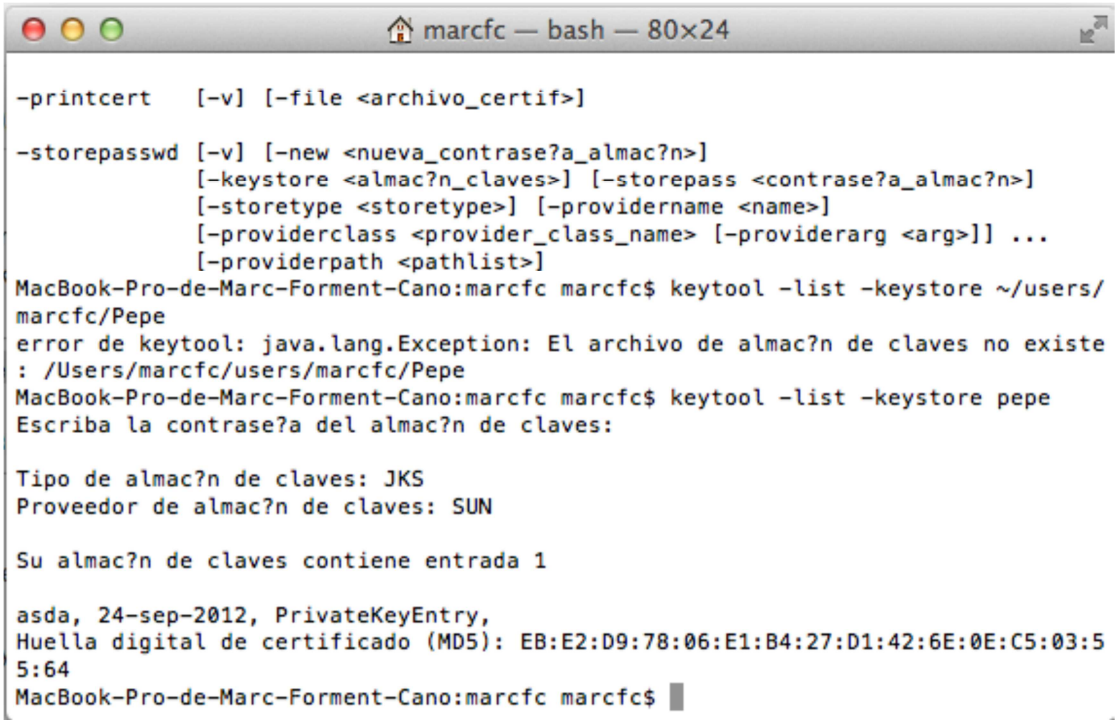
En la siguiente pantalla indicamos los datos que nos piden y una validez de al menos 50 años y clicamos Next.



Indicamos el destino del apk y, si todo ha ido bien, habremos creado un apk firmado pero en el que no funcionan los mapas. La razón es que necesitamos la keystore para conseguir la clave para los mapas, así que este paquete no nos servirá para el upload final.

Para obtener la clave de Google Maps primero necesitamos la fingerprint MD5 de la keystore que acabamos de crear. Para conseguirla hemos de utilizar la utilidad keytool de Java. Así que si tenemos el SDK de Java sólo hemos de abrir un terminal y ejecutar la siguiente instrucción:

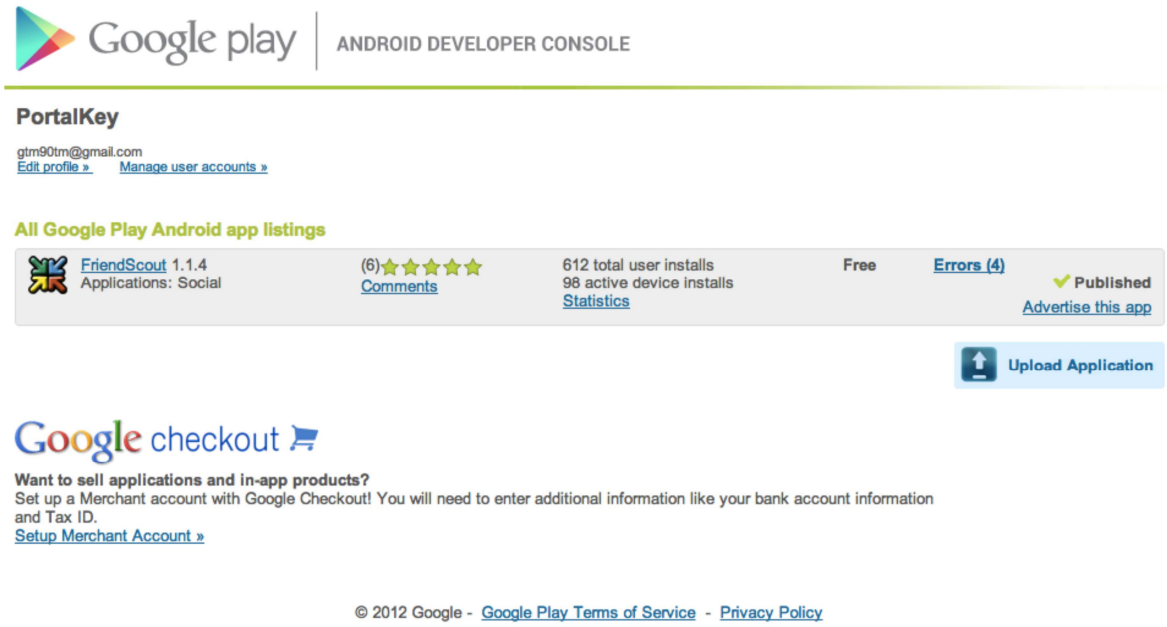
keytool -list -keystore “ruta de la keystore”



Con esta clave iremos a la siguiente dirección web http://developers.google.com/android/maps-api-signup, introduciremos la clave MD5 en el lugar correspondiente, aceptaremos las condiciones de uso y clicaremos en Aceptar. Si todo ha ido bien veremos una pantalla similar a ésta con la clave correspondiente:

Volveremos a firmar la aplicación como ya indicamos en el paso anterior, pero esta vez seleccionando la opción de utilizar una keystore existente y procederemos a hacer los preparativos para publicar finalmente la aplicación en Google Play.

Primero, para subir la aplicación tendremos que registrarnos como Developers en el siguiente enlace http://play.google.com/app/publish/, se habrá de pagar la cuota de desarrolladores de Google y, una vez sea haya verificado todo, puedes comenzar a subir la aplicación.



Si ya tienes la aplicación instalada y publicas versiones nuevas de la aplicación en Google Play, aparecerán como actualizaciones.

8. Conclusión

Nuestro proyecto, compuesto por una aplicación Android, una base de datos online y los mecanismos que permiten la comunicación entre éstos, ha permitido que aumentemos nuestros conocimientos tanto técnicos como de trabajo en equipo además de muchas otras capacidades.

Hemos aumentado de manera considerable nuestro conocimiento del lenguaje de programación Java. Los problemas que se han presentado durante el desarrollo del proyecto nos han dado la capacidad de resolver los problemas que se presenten de una forma mucho más fluida. Hemos experimentado el aprendizaje de nuevos SDK tanto de Android como de Google Maps. Esto nos será muy útil en el futuro por la gran utilidad que tienen estos dos SDKs ya que están muy extendidos y por ladestreza obtenida de cara a aprender otros nuevos.

Por la parte de la base de datos y dada nuestra especialidad, hemos obtenido experiencia en SQL que probablemente no habríamos conseguido de otra manera a causa de la necesidad de otros conocimientos en un proyecto de la rama de informática industrial. También ha sido muy interesante aprender un poco sobre PHP, una herramienta muy útil en los servidores web en la actualidad.

También hemos experimentado lo que es trabajar en un grupo durante largos periodos de tiempo, los periodos de entrega en un proyecto más grande de lo que estábamos acostumbrados, la experiencia de tener un director de proyecto de forma similar a un jefe en una situación real... Y todo esto en Irlanda, lo que nos ha permitido mejorar nuestro nivel de inglés y obtener una experiencia diferente a la que teníamos en Valencia.

Finalmente, este proyecto nos ha permitido mejorar como Ingenieros Informáticos y como personas. Todos estos conocimientos nos han permitido encontrar trabajo y estamos seguros que nos proveerá de otras cosas que aún no hemos descubierto.