Utilización de Google maps y geolocalización para dispositivos móviles

1 Introducción

En este documento se explica cómo abrir un mapa Google y marcarlo con la posición actual en que nos encontramos, utilizando alternativamente para ello los servicios de localización que nos proporciona bien un proveedor de GPS (`LocationManager.GPS_PROVIDER`) o la red a la que estemos conectados (`LocationManager.NETWORK_PROVIDER`)

Se pretende desarrollar completamente un proyecto con Android Studio ©, que incluye las librerías adecuadas para un componente software de teléfonos móviles que tiene un `gradle.build` definido como sigue (sección Android):

```
compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "com.ejemplo.mcapel.pruebamapas3"
        minSdkVersion 19
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
```

El dispositivo móvil ejecuta Android (versión posterior a la 4.4, API 19) como sistema operativo, de acuerdo con el esquema general que muestra la siguiente figura:

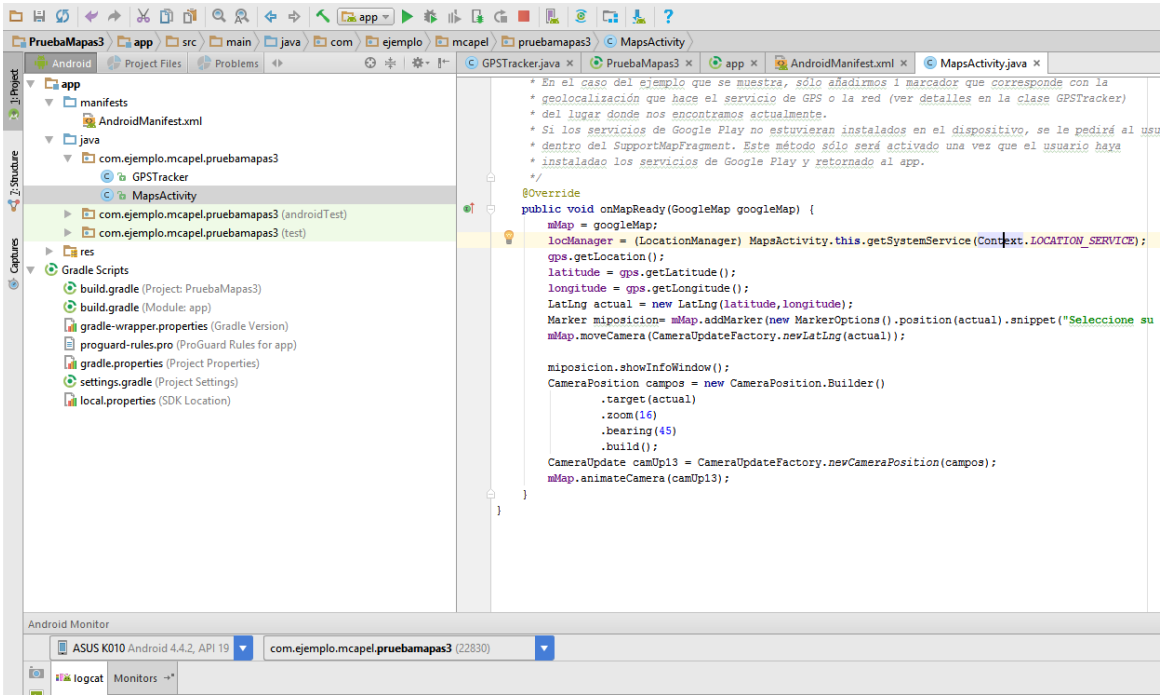

Figura 1: Estructura del proyecto que utiliza Google Maps con `LocationManager`

La estructura del *manifiesto*: `AndroidManifest.xml` es la que muestra la tabla 1.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ejemplo.mcapel.pruebamapas3">
    <!--
         The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
         Google Maps Android API v2, but you must specify either coarse or fine
         location permissions for the 'MyLocation' functionality.
    -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <!--
             The API key for Google Maps-based APIs is defined as a string resource.
             (See the file "res/values/google_maps_api.xml").
             Note that the API key is linked to the encryption key used to sign the APK.
             You need a different API key for each encryption key, including the release key that is used to
             sign the APK for publishing.
             You can define the keys for the debug and release targets in src/debug/ and src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".MapsActivity"
            android:label="@string/title_activity_maps">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Tabla 1: Archivo AndroidManifest.xml

Antes de poder ejecutar una aplicación Android que accede a mapas, necesitamos obtener una *clave de acceso*. El archivo dentro de los recursos (`\res\values`) más importante para que funcionen los mapas se denomina `google_maps_api.xml`, que ha de contener la clave obtenida de Google Maps API Manager `https://console.developers.google.com/apis/credentials?project=lunar-inn-88818`.

```
<resources>
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIza ...</string>
</resources>
```

Tabla 2: Archivo `google_maps_api.xml`

La forma más fácil de obtener una clave para que nuestra aplicación pueda acceder al API de Google Maps consiste acceder al enlace `https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=83:40:2E:EB:76:1E:05:78:1A:FF:E4:F8:17:C6:35:10:DD:7A:CB:63%3Bcom.ejemplo.mcapel.pruebamapas3` y presionar "Create" al final.

También podemos añadir las *credenciales* de nuestro app:(`com.ejemplo.mcapel.pruebamapas3`) a una de las claves que tengamos creadas ya en Google Maps API Manager, utilizando la siguiente línea: `83 : 40 : 2E : EB : 76 : 1E : 05 : 78 : 1A : FF : E4 : F8 : 17 : C6 : 35 : 10 : DD : 7A : CB : 63; com.ejemplo.mcapel.pruebamapas3`

Se puede encontrar más información acerca de cómo consegir una clave en la siguiente página `https://developers.google.com/maps/documentation/android/start#get-key`.

Una vez que tengamos nuestra clave (que suele ser un acadena de 40 caracteres que comienza por "AIza"), hay que sustituir la cadena "`google_maps_key`" en el archivo anterior. `google_maps_api.xml`.

El aspecto final de la pantalla de ejecución del app que sirve de ejemplo en este tutorial es el que se muestra en la figura 2

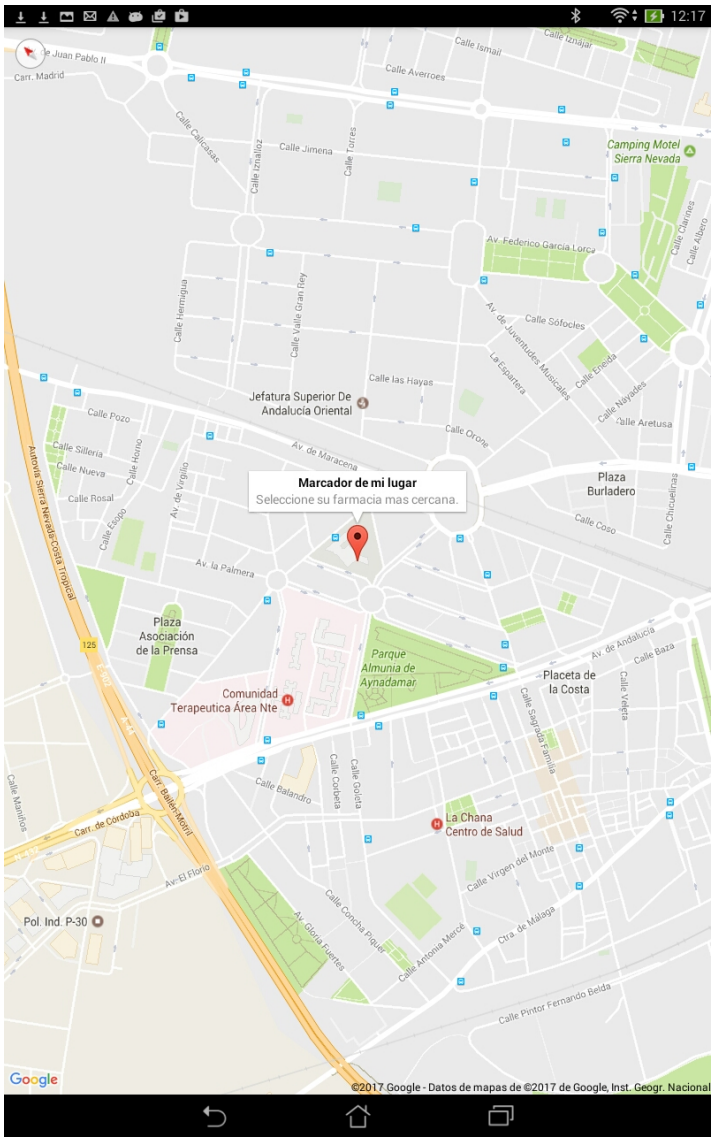

Figura 2: Pantalla correspondiente la vista del mapa que abre el app-ejemplo

1.1 Clases de la aplicación

La base de la detección de la localización en el mapa la realiza la clase–Java `GPSTracker` de nuestro proyecto, utilizando para ello la instrucción `gps.getLocation();` dentro de un método *callback* denominado `onMapReady(GoogleMap googleMap)` de `MapsActivity`.

La clase Java `MapsActivity` manipula el mapa una vez que esté cargado.

El *callback* `onMapReady(GoogleMap googleMap)` es activado cuando el mapa está listo para ser utilizado. Aquí es donde podemos añadir marcadores o líneas, añadir escuchadores(listeners) o mover la cámera. En el caso del ejemplo que se muestra, sólo añadirmos 1 marcador que corresponde con la geolocalización que hace el servicio de GPS o la red (ver detalles en la clase GPSTracker 4) del lugar donde nos encontramos actualmente.

La cámara la movemos hasta la posición en la que nos encontramos y ampliamos la vista con la instrucción `CameraUpdateFactory.newCameraPosition(campos);`

Créditos

- `http://www.vogella.com/tutorials/AndroidGoogleMaps/article.html`
- Android developer homepage `http://developer.android.com/index.html`
- Google Maps API Manager `https://console.developers.google.com/apis/credentials?project=lunar-inn-8881`
- Información acreca de cómo conseguir una clave para acceder a Google Maps API `https://developers.google.com/maps/documentation/android-api/start#get-key`
- Google API for Android (Location services) `https://developers.google.com/android/reference/com/google/android/gms/location/LocationServices`

```
1  public class GPSTracker extends Service implements LocationListener {
2      private final Context mContext;
3      boolean isGPSEnabled = false;
4      boolean isNetworkEnabled = false;
5      boolean canGetLocation = false;
6      Location location = null; // localizacion
7      double latitude; // latitud
8      double longitude; // longitud
9      private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10; // 10 meters
10     private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1; // 1 minute
11
12     protected LocationManager locationManager;
13     public GPSTracker(Context context) {
14         this.mContext = context;
15         getLocation();
16     }
17     public Location getLocation() {
18         try {locationManager = (LocationManager) mContext
19                 .getSystemService(LOCATION_SERVICE);
20             isGPSEnabled = locationManager
21                 .isProviderEnabled(LocationManager.GPS_PROVIDER);
22             isNetworkEnabled = locationManager
23                 .isProviderEnabled(LocationManager.NETWORK_PROVIDER);
24             if (!isGPSEnabled && !isNetworkEnabled) {
25             } else { this.canGetLocation = true;
26                 if (isNetworkEnabled) {
27                     locationManager.requestLocationUpdates(
28                             LocationManager.NETWORK_PROVIDER,
29                             MIN_TIME_BW_UPDATES,
30                             MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
31                     Log.d("Network", "Network␣Enabled");
32                     if (locationManager != null) {
33                         location = locationManager
34                                 .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
35                         if (location != null) {
36                             latitude = location.getLatitude();
37                             longitude = location.getLongitude();
38                         }
39                     }
40                 }
41                 if (isGPSEnabled) {
42                     if (location == null) {
43                         locationManager.requestLocationUpdates(
44                                 LocationManager.GPS_PROVIDER,
45                                 MIN_TIME_BW_UPDATES,
46                                 MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
47                         Log.d("GPS", "GPS␣Enabled");
48                         if (locationManager != null) {
49                             location = locationManager
50                                     .getLastKnownLocation(LocationManager.GPS_PROVIDER);
51                             if (location != null) {
52                                 latitude = location.getLatitude();
53                                 longitude = location.getLongitude();
54                             }
55                         }
56                     }
57                 }
58             }
59         } catch (Exception e) {
60             e.printStackTrace(); }
61         return location;
62     } ...
63 }
```

Tabla 3: La clase `GPSTracker`

```java
import android.content.Context;
import android.location.LocationManager;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import android.location.Location;
import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
    private GoogleMap mMap;
    double latitude = 0;
    double longitude = 0;
    Location location;
    private GPSTracker gps;
    private LocationManager locManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        gps = new GPSTracker(((Context) this));
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
 // Obtener el fragmento: SupportMapFragment y ser notificado cuando el mapa este listo para ser utilizado.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        locManager = (LocationManager) MapsActivity.this.getSystemService(Context.LOCATION_SERVICE);
        location= gps.getLocation();
        latitude= location.getLatitude();
        longitude= location.getLongitude();
        LatLng actual = new LatLng(latitude,longitude);
        Marker miposicion= mMap.addMarker(new MarkerOptions().position(actual).snippet("Seleccione su farmacia ma
        mMap.moveCamera(CameraUpdateFactory.newLatLng(actual));
        miposicion.showInfoWindow();
        CameraPosition campos = new CameraPosition.Builder()
                .target(actual)
                .zoom(16)
                .bearing(45)
                .build();
        CameraUpdate camUp13 = CameraUpdateFactory.newCameraPosition(campos);
        mMap.animateCamera(camUp13);
    }
}
```

Tabla 4: La clase `MapsActivity`