Sistemas Operativos - Linux

Carlos E. Jiménez Pelayo[1]

Práctica de Android

Permisos en una App

Ciateq, Unidad Guadalajara

[1]pelayoce@mx1.ibm.com

22 Septiembre 2017

1. Objetivo de la práctica

Abrir la ventana de permisos de tu App (desde la App). Hacer una App con dos actividades, en la primera deben haber 2 botones, uno para ver los permisos y otro para ver un marcador en Google Maps (ejemplo realizado en clase).

2. Requisitos

El desarrollo de la App se lleva a cabo en Android Studio 2.3.3 bajo el S.O. Windows. Los elementos instalados son: Android 7.1.1 (Nougat) API Level 25 dentro del SDK Manager, así como Google Play services y Google USB Driver bajo SDK Tools.

Se utiliza un dispositivo móvil Android para hacer pruebas de Google Maps, que no pueden realizarse en el emulador.

Se requiere además un cable mini-USB conectado de la computadora al dispositivo móvil, el cuál debe tener habilitado el modo de 'USB debugging' dentro del menú 'Developer options', bajo la aplicación de 'Settings'.

3. Consideraciones para Google Maps

El dispositivo móvil requiere tener habilitado los siguientes servicios para utilizar el GPS, Wi-Fi y redes móviles, necesarios para una mejor determinación de la ubicación en Google Maps:

- Settings >WiFi >On
- Settings >Mobile networks >On
- Settings >Location >On
- Settings >Location >Mode >High Accuracy

4. Creación del proyecto

En Android Studio se crea un nuevo proyecto con las siguientes opciones, dando clic al botón Finish al final:

- New Project >Application name > *PERM_MAP*
- Target Android Devices >Phone and Table >Minimum SDK >API14: Android 4.0 (IceCreamSandwich)
- Add an Activity to Mobile >Empty Activity
- Customize the Activity >Activity Name >*MainActivity*, Layout Name >*activity_main*

El Proyecto nos mostrará la actividad MainActivity.java así como el layout activity_main.xml

Ahora agregaremos la segunda actividad, dando clic derecho sobre app, y seleccionando las siguientes opciones: app >New >Google >Google Maps Activity

En el panel 'Configure Activity' que aparece dejamos las opciones por defecto y damos clic en el botón Finish:

- Activity Name : *MapsActivity*
- Layout Name: *activity_maps*
- Title: *Map*

Aparecerá el archivo *app >res >values >google_maps_api.xml*, que se detalla a continuación.

5. Archivo de recursos google_maps_api.xml

El siguiente listado muestra el código contenido en el archivo *app >res >values >google_maps_api.xml*, el cual se generó automáticamente al agregar la actividad. En el archivo se detalla el proceso para conseguir la 'Google Maps API key'. El archivo original cuenta con el texto 'YOUR_KEY_HERE', el cual se reemplaza con la llave; la llave que se insertó en el código de abajo es la siguiente:
AIzaSyA1WYsX8PjauOpuR_QKV9NtDfcTvDWUqmQ
Una vez ingresada hay que guardar los cambios del archivo.

Nota : a la izquierda del código se indica con un cuadro en verde las líneas de texto que se necesita copiar y pegar, o ingresar manualmente.

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the
        end:

    https://console.developers.google.com/flows/enableapi?apiid=
        maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=51:89:62:C3:D5:79:AA:
        AA:33:5A:18:C8:ED:13:0B:2D:D3:D8:16:FB%3Bcom.example.user.perm_map

    You can also add your credentials to an existing key, using these values:

    Package name:
    51:89:62:C3:D5:79:AA:AA:33:5A:18:C8:ED:13:0B:2D:D3:D8:16:FB

    SHA-1 certificate fingerprint:
    51:89:62:C3:D5:79:AA:AA:33:5A:18:C8:ED:13:0B:2D:D3:D8:16:FB

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key
```

```
    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable=
         "false">
              AIzaSyA1WYsX8PjauOpuR_QKV9NtDfcTvDWUqmQ
        </string>
</resources>
```

6. Archivo MapsActivity.java

Al agregar la actividad de Google Maps también se generó el archivo *app >java>com.example.user.perm_map >MapsActivity.java*, el cual cuenta con algunas instrucciones y configura un marcador de la ciudad de 'Sydney' para cuando se abre la aplicación. En el método *onMapReady(GoogleMap googleMap)* hacia el final del archivo se agrega otro marcador para el Aeropuerto de la Ciudad de Guadalajara, así como la condición *ContextCompat.checkSelfPermission* para habilitar a Google Maps para que nos muestre la ubicación actual del dispositivo móvil.

Nota: puede que los textos ContextCompat y PackageManager se marquen en rojo, por lo cual hay que seleccionarlos de manera individual, presionar Alt+Enter y seleccionar 'Import Class' para que se incluyan en el encabezado del archivo.

```
package com.example.user.perm_map;

import android.content.pm.PackageManager;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import android.support.v4.content.ContextCompat;

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.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback
    {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready
            to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
            getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    /**
```

```
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the
         camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be
         prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once
         the user has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney and move the camera
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in
            Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));

        // Add a marker in GDL Airport and move the camera
        LatLng gdlAirport = new LatLng(20.5260, -103.3076); // N-W
        mMap.addMarker(new MarkerOptions().position(gdlAirport).title("Marker in
    GDL Airport"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(gdlAirport));

        if(ContextCompat.checkSelfPermission(this,
                android.Manifest.permission.ACCESS_FINE_LOCATION) ==
                PackageManager.PERMISSION_GRANTED) {
            mMap.setMyLocationEnabled(true);
        } else {
            // Show rationale and request permission.
        }
    }
}
```

7. Archivo de recursos strings.xml

Ahora editaremos el archivo *app >res >values >strings.xml*. En él ya deben aparecer definidos el nombre de la aplicación y el título de la actividad 'maps', por lo que agregaremos el título de la actividad 'main', de manera que el código se asemeje al siguiente:

```
<resources>
    <string name="app_name">PERM_MAP</string>
    <string name="title_activity_maps">Map</string>
    <string name="title_activity_main">Main</string>
</resources>
```

8. Diseño de la interfase (layout)

La siguiente imagen muestra el diseño de la interfase dentro del panel *app >res >layout >activity_main.xlm*:

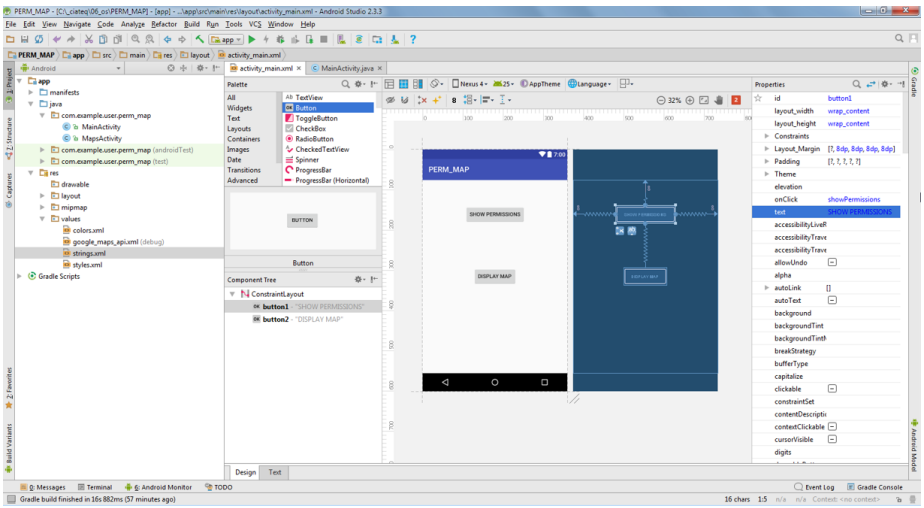

Figura 1: Vista del panel activity_main.xml

Para este diseño se debe eliminar el texto que aparece en la interfase (Hello World) y se arrastra la figura 'Button' desde la paleta de herramientas hasta la interfase, el cual se modifica en el panel de propiedades ubicado del lado derecho de la pantalla, para que tenga los siguientes valores (si no aparecen todos los parámetros, dar clic en 'View all properties'):

- Properties >ID > button1
- Properties >Button >onClick > showPermissions
- Properties >TextView >text > SHOW PERMISSIONS

Ahora se agrega el segundo botón arrastrándolo de manera similar desde la paleta de herramientas, centrándolo en la interfase debajo del primer botón, y fijándo ambos botones en relación a los bordes de la pantalla con los 'resortes' de cada lado; enseguida se configura el segundo botón en el panel de propiedades con los siguientes valores:

- Properties >ID > button2
- Properties >Button >onClick > showMap
- Properties >TextView >text > DISPLAY MAP

La propiedad *onClick* asigna a cada botón las funciones *showPermissions* y *showMap* respectivamente, las cuales serán declaradas en el archivo *MainActivity.java* a continuación.

9. Archivo MainActivity.java

El siguiente listado muestra el código contenido en el archivo *app >java>com.example.user.perm_map >MainActivity.java*. El archivo es generado automáticamente pero no contiene las actividades que se declararon en el layout, por lo que a la clase MainActivity le agregaremos las funciones *showPermissions* y *showMap*.

Nota: es necesario importar las librerías (Alt+Enter) para los textos que aparecerán en rojo al pegar o transcribir el código, tales como View, Toast, Intent, Settings, addFlags, Uri, setData.

```
package com.example.user.perm_map;

import android.content.Intent;
import android.net.Uri;
import android.provider.Settings;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void showPermissions(View view){
        Toast.makeText(getApplicationContext(),"Checking Permissions",
                Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS)
    ;
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        Uri uri = Uri.fromParts("package", getPackageName(), null);
        intent.setData(uri);
        startActivity(intent);
    }

    public void showMap(View view){
        Toast.makeText(getApplicationContext(),"Showing Google Maps",
                Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(this, MapsActivity.class);
        startActivity(intent);
    }
}
```

El método *showPermissions* está ligado con el primer botón de la interfase; al ejecutarse el código, 'Toast' imprime brevemente el texto para la verificación de los permisos, para posteriormente llevar a cabo los siguientes pasos:

- Se envía el mensaje *Settings.ACTION_APPLICATION_DETAILS_SETTINGS*, que abre la pantalla de configuración de la App, donde el usuario puede asignar los permisos de forma manual.

- *intent.FLAG_ACTIVITY_NEW_TASK*. Esta es una bandera opcional, que al estar activada abre la ventana de configuración (Settings) como una nueva actividad independiente, de lo contrario se abrirá en la actividad que se está ejecutando actualmente.
- *Uri.fromParts("package", getPackageName(), null)*. Crea y prepara un URI (Uniform Resource Identifier), donde *getPackageName()* regresa el nombre de la aplicación.
- *intent.setData(uri)*. Es necesario ejecutar esta instrucción ya que Android espera el nombre para el 'intent' *Settings.ACTION_APPLICATION_DETAILS_SETTINGS*.
- *startActivity* inicia la actividad.

El método *showMap* muestra brevemente el texto para indicar que se abrirá Google Maps, y ejecuta la actividad MapsActivity de la API, mostrando el mapa con el marcador designado, así como la ubicación del dispositivo móvil.

10. Archivo AndroidManifest.xml

El siguiente listado muestra el código contenido en el archivo *app>manifests>AndroidManifest.xml*, el cual también se genera en forma automática:

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

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

En el código podemos resaltar el permiso para tener una ubicación fina mediante la localización GPS, así como el valor de la llave com.google.android.geo.API_KEY (android:value="@string/google_maps_key") que ingresamos previamente en el archivo de recursos *google_maps_api.xml*, ambos requeridos por Google Maps.

Solo es necesario agregar de forma manual el acceso a internet, mediante la línea *<uses-permission android:name="android.permission.INTERNET" />*.

11. Un vistazo a la App

Ahora es momento de construir la aplicación (Make Project = Ctrl+F9) y ejecutarla (Run ápp'= Shift+F10). Al intentar ejecutarla desde Android Studio, seleccionamos el dispositivo Android conectado al puerto USB, para que la App se instale. Ésta se iniciará de forma automática, desplegando la pantalla de la primer actividad, como en la siguiente imagen:

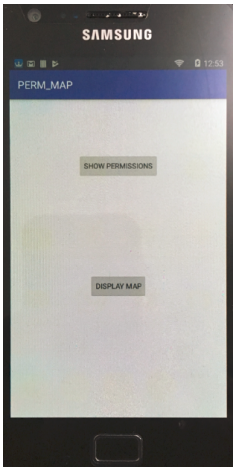

Figura 2: Actividad1

Al seleccionar el primer botón, se accesa al menú de propiedades y permisos de la aplicación, mostrada en la imagen de abajo. Con el botón 'atrás' se regresa al menú de la actividad 1.

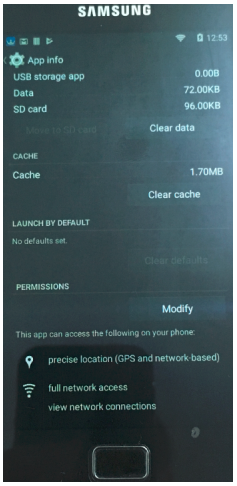

Figura 3: Menú de configuración y permisos de la App

Al seleccionar el segundo botón, se accesa a Google Maps, donde el marcador apunta a la ubicación del Aeropuerto de Guadalajara, mientras que el 'punto' azul muestra la ubicación GPS del dispositivo móvil.

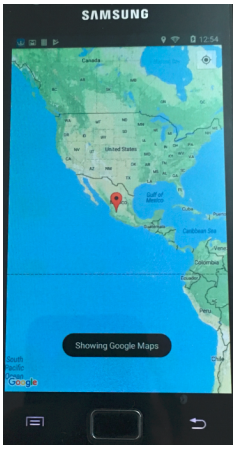

Figura 4: Actividad de Google Maps

Mediante el botón 'atrás' se regresa al menú de la actividad 1, y si se vuelve a presionar se cierra la aplicación, aunque esta queda residente en memoria.

Si la bandera *intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)* fue habilitada en el archivo *MainActivity.java* y se accesa el menú de propiedades y permisos de la aplicación, entonces se pueden observar un par de aplicaciones disponibles en el panel de aplicaciones residentes en memoria, que se accesa al mantener presionado el botón de Home; de lo contrario, solo deberá aparecer la App apuntando a la actividad 1, más no el menú de propiedades.

El o las actividades activas se pueden cerrar desde el panel de aplicaciones residentes en memoria.

Referencias

[1] STACKOVERFLOW.COM, 2015, *How to programmatically open the Permission Screen for a specific app on Android Marshmallow? [en línea]. 2015. Accesado: 09-Septiembre-2017. Disponible en: https://stackoverflow.com/questions/32822101/how-to-programmatically-open-the-permission-screen-for-a-specific-app-on-android*