TUTORIAL: GEOLOCALIZACIÓN

Introducción

En este tutorial se va a explicar cómo hacer una aplicación sencilla que mostrará la latitud y la longitud dependiendo del lugar en el que se encuentre el dispositivo. También se mostrará un mapa proporcionado por google indicando la posición (dentro de un radio de error) actual de la persona que esté usando la aplicación.

Cabe destacar que es necesario tener el GPS del dispositivo activo y encontrarse en el exterior ya que la comunicación con el satélite se hace imposible dentro de edificios.

Permisos

Primero lidiaremos con los permisos necesarios para que la aplicación funcione correctamente. En el archivo AndroidManifest.xml tenemos que añadir la siguiente línea: <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

Layout

Ahora en layout añadimos el texto que queremos que se muestre por pantalla. Tenemos que añadir dos textos descriptivos: “Latitud” y “Longitud” y otros dos campos de texto vacíos donde aparecerá la posición dada por el satélite.

Cada campo de texto vacío en el documento activity_main.xml tiene que verse de la siguiente forma:

```
<TextView
        android:id="@+id/TextLat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dip"
        android:layout_marginRight="5dip"
        android:text=" "
        android:textSize="20dip" >
</TextView>
```

TextLat es el identificador que tendrá ese campo de texto (en este caso para la latitud) para poder asignarle valores más adelante. Como se puede apreciar, el texto queda vacío (“ “).

Actividad principal.

Ahora procederemos a hacer la actividad principal de la aplicación. Siguiendo el esquema que se usa por defecto, en el archivo de actividad principal se habrá generado una función principal. Dentro de esta creamos dos variables que nos servirán para guardar los valores de latitud y longitud.

```
TextView t1;
TextView t2;
```

Ahora tenemos que asociar estas variables con los textos que hemos creado anteriormente en el layout. Para ello, asignamos las variables de la siguiente forma:

```
t1=(TextView)findViewById(R.id.TextLat);
t2=(TextView)findViewById(R.id.TextLong);
```

Así, t1 queda asignado a la latitud y t2 a la longitud y cuando haya algún valor en dichas variables se mostrarán por la pantalla del dispositivo.

Para obtener la localización, es necesario comunicarse con el sistema operativo Android. Para ello usamos un "LocationManager" y un "LocationListener".

```
LocationManager lm =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);

LocationListener ll = new mylocationListener();
lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, ll);
```

El objeto LocationManager al que se ha llamado "lm" se comunica con el sistema para obtener la geolocalización actual a través del GPS. Los demás parámetros son tiempo mínimo, distancia mínima y el LocationListener.

Mientras que el objeto LocationListener llamado "ll" se encarga de "escuchar" los cambios que se produzcan y actuar en consecuencia.

Ahora tenemos que crear la clase "mylocationListener" que hemos usado anteriormente. Esta clase se encargará de gestionar las cosas que queremos que haga la aplicación cuando ocurran cambios.

```
class mylocationListener implements LocationListener{
```

Añadimos los métodos a la clase empezando por onLocationChanged que se llamará cada vez que la posición cambie. Primero se comprueba que la posición existe y es válida y después se obtienen la longitud y la latitud y se asignan a las variables que se encargarán de mostrar los resultados por pantalla.

```
@Override
public void onLocationChanged(Location location) {
	if(location !=null){
		double pLong=location.getLongitude();
		double pLat = location.getLatitude();

		t1.setText(Double.toString(pLat));
		t2.setText(Double.toString(pLong));
}
```

Mapa con Google Maps API v2.

Para trabajar con mapas en nuestra aplicación hemos recurrido a los servicios que nos proporciona *Google Maps* lo que nos otorga la posibilidad de recurrir a muchas funciones interesantes que podremos explorar mas adelante pero que en este tutorial no han sido introducidas.

En primer lugar para trabajar con los mapas de *Google Maps* debemos conocer la localización del certificado digital de depuración. Si trabajamos con Eclipse bastará con acceder al menú *Windows>preferences>Android>Build*, y obtendremos la ruta del fichero en la sección Default debug keystore.

Adjuntaremos nuestras fuentes al final del tutorial, donde también se podrá ver como se realizan estos pasos en Android Studio.

El siguiente paso es obtener la huella digital SHA1 del fichero en cuestión. Para ello abriremos el intérprete de comandos y nos situaremos en la siguiente carpeta (o similar) *C:/Archivos de programa\Java\jre\bin*

Ejecutaremos el siguiente comando añadiendo la ruta obtenida en el primer paso, al final de la instrucción:

Keytool –v –list –keystore ruta

El programa solicita una contraseña, la dejamos en blanco y nos presentara diversa información. Nos interesa la huella digital del certificado en codificación SHA1. Es importante copiar la secuencia de dígitos.

Ahora iremos a la página https://code.google.com/apis/console para obtener la clave de *Google Maps*.

Introduciremos nuestro usuario de Google para realizar la solicitud.

Creamos un nuevo proyecto en nuestra consola. Pulsamos el botón *Create Project* y lo nombramos. Aceptamos términos y condiciones y pulsamos en *Create*.

En la parte izquierda aparecerá un menú. Seleccionamos *APIs & auth / APIs*. Localizamos el elemento *Google Maps Android API v2* y pulsamos en el botón OFF que pasara a ON.

Ahora vamos a la opción *APIs & auth / Credentialsy* y creamos una nueva llave, seleccionando *Android Key*.

En la ventana que aparece introducimos la huella SHA 1, seguida de ";" y del nombre del paquete de la aplicación que va a usar el mapa. Creamos la clave y la copiamos, dado que nos hará falta a continuación.

Ahora vamos a nuestro proyecto. Debemos de asegurarnos de tener el paquete de *Google Play* instalado, podemos verlo en el *Android SDK Manager*.

Importamos las librerías de *Google Maps* al proyecto, tal y como si fuera un proyecto existente de Android. La ruta a seguir debería de ser:

Android-sdk/extras/google/ google_play_services/libproject/google-play-services_lib partiendo de la carpeta donde está instalado *Android SDK Manager*.

Finalizamos la importación mediante Propiedades del Proyecto > Android > Add y seleccionamos la librería.

Ahora en el manifest debemos añadir las siguientes líneas:

```
<meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version"/>
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyBiJjYklUmMEA3_AI3akz2B5JOiZPxsvng"/>
```

Donde en la última línea introducimos nuestra clave. Además añadiremos los siguientes permisos, cambiando el nombre por el de vuestro paquete, el nuestro se llama geolocalización:

```
<permissio
n
    android:name="com.GPS.geolocalizacion.permission.MAPS_RECEIVE"
    android:protectionLevel="signature"/>


    <uses-permission
    android:name="com.GPS.geolocalizacion.permission.MAPS_RECEIVE"/>


    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVI
CES"/>

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

```
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

En el layout de la aplicación deberemos añadir lo siguiente para visualizar el mapa:

```
<LinearLayout
        android:id="@+id/linearLayout3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
            <fragment
                android:id="@+id/map"
                android:layout_width="match_parent"
                android:layout_height="match_parent"

class="com.google.android.gms.maps.SupportMapFragment"/>
    </LinearLayout>
```

Por ultimo tenemos que hacer que nuestra clase utilice los métodos de Fragment Activity, para ello cambiaremos la clase principal:

Public class MainActivity extends FragmentAtivity {

Hecho esto ya podremos utilizar el mapa en nuestra aplicación. Solo necesitaremos crear el mapa y asociarlo a nuestro xml:

Declaramos el mapa del tipo Google Map.

```
GoogleMap mapa;
```

Lo cargamos en el método OnCreate.

```
mapa=((SupportMapFragment)getSupportFragmentManager().findFragmentById(R.id.map)).get
Map();
```

Posteriormente utilizamos algunas de las funciones en el método OnLocationchanged que implementamos para obtener la latitud y la longitud vía GPS.

Antes de finalizar merece la pena contar que Google Maps no es software libre, podremos usarlo de forma gratuita siempre y cuando nuestra aplicación no supere las 15.000 codificaciones geográficas al día.

Enlaces consultados.

https://www.youtube.com/watch?v=7-n6p6RxSS8

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