

Objetivos:

- Se describe como utilizar las librerías de Google Maps V2.

GOOGLE MAPS (API V2)

Google Maps nos proporciona un servicio de cartografía *online* que podremos utilizar en nuestras aplicaciones Android. Veamos las claves necesarias para utilizarlo. Estudiaremos la versión 2 del API que incorpora interesantes ventajas respecto a la versión anterior. Entre estas ventajas destaca el menor tráfico intercambiado con el servidor, la utilización de *fragments* y los gráficos en 3D. Como inconveniente resaltar que la nueva versión solo funciona en el dispositivo con Google Play instalado.

Conviene destacar que a diferencia de Android, Google Maps no es un software libre, por lo que está limitado a una serie de condiciones de servicio. Podemos usarlo de forma gratuita siempre que nuestra aplicación no solicite más de 15.000 codificaciones geográficas al día. Podemos incluir propaganda en los mapas o incluso podemos usarlo en aplicaciones móviles de pago (para sitios Web de pago es diferente). Una información más completa de este API la encontramos en:

<https://developers.google.com/maps/documentation/android/>

OBTENCIÓN DE UNA CLAVE GOOGLE MAPS

Para poder utilizar este servicio de Google, igual como ocurre cuando se utiliza desde una página web, va a ser necesario registrar la aplicación que lo utilizará. Tras registrar la aplicación se nos entregará una clave que tendremos que indicar en la aplicación.

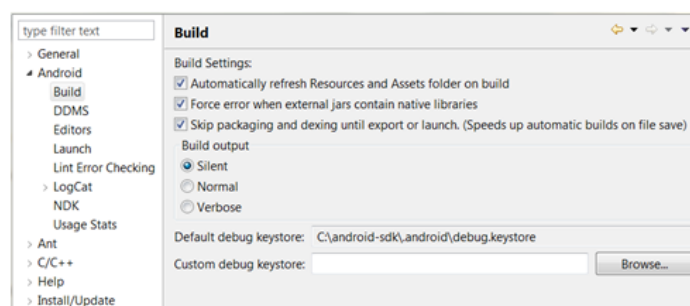
Realmente vamos a necesitar dos claves diferentes, una durante el proceso de desarrollo y otra para la aplicación final. La razón es que se genera una clave diferente en función del certificado digital con la que se firma la aplicación. En la fase de desarrollo las aplicaciones también han de ser firmadas digitalmente, pero en este caso el SDK utiliza un certificado especial (el certificado de depuración), utilizado solo en la fase de desarrollo.

Veamos cómo obtener la clave *Google Maps* para el certificado de depuración. En caso de querer distribuir tu aplicación, una vez terminada tendrás que firmarla con un certificado digital propio. Este proceso se explica en el último capítulo. Recuerda que será necesario reemplazar la clave *Google Maps*, por otra, esta última asociada al certificado digital usado en la fase de distribución.



Ejercicio: Obtención de una clave Google Maps

1. El primer paso va a consistir en descubrir donde está almacenado el certificado digital de depuración. Utilizando el entorno Eclipse accede al menú *Windows > Preferences > Android > Build*. Aparecerá el siguiente cuadro de diálogo:



En el cuadro informativo *Default debug keystore*: aparece la ruta del fichero donde se almacena el certificado digital de depuración.

Utilizando el entorno **Android Studio** accede a la carpeta `.android` que encontrarás en la carpeta de tu usuario. Dentro se almacena el fichero `debug.keystore` con el certificado digital de depuración. En Windows la ruta de este fichero podría ser `C:\Users\<Usuario>\.android\debug.keystore`. En Linux y Mac la ruta es `./android/debug.keystore`.

2. Copia esta ruta en el portapapeles.

3. Ahora necesitamos extraer la huella digital SHA1 de este fichero. Para extraer la huella digital puedes utilizar el programa `keytool`. En Windows este programa se encuentra en la carpeta `C:\Program Files\Java\jre7\bin\` o en una similar. Abre un intérprete de comandos y sitúate en la carpeta anterior (o similar).

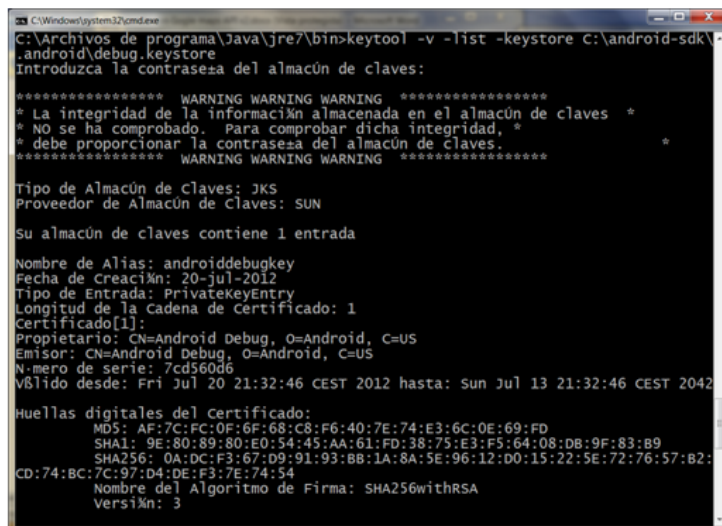
```
cd C:\Archivos de programa\Java\jre7\bin
```

4. Ejecuta el siguiente comando reemplazando el nombre del fichero por el que acabas de copiar en el portapapeles.

```
keytool -v -list -keystore [Tu debug.keystore path]
```

En nuestro ejemplo:

```
keytool -v -list -keystore C:\android-sdk\.android\debug.keystore
```



```
C:\Windows\system32\cmd.exe
C:\Archivos de programa\Java\jre7\bin>keytool -v -list -keystore C:\android-sdk\.android\debug.keystore
Introduzca la contraseña del almacén de claves:

***** WARNING WARNING WARNING *****
* La integridad de la información almacenada en el almacén de claves *
* NO se ha comprobado. Para comprobar dicha integridad, *
* debe proporcionar la contraseña del almacén de claves. *
***** WARNING WARNING WARNING *****

Tipo de Almacén de claves: JKS
Proveedor de Almacén de claves: SUN

Su almacén de claves contiene 1 entrada

Nombre de Alias: androiddebugkey
Fecha de Creación: 20-Jul-2012
Tipo de Entrada: PrivateKeyEntry
Longitud de la Cadena de Certificado: 1
Certificado[1]:
Propietario: CN=Android Debug, O=Android, C=US
Emisor: CN=Android Debug, O=Android, C=US
Número de serie: 7cd560d6
Válido desde: Fri Jul 20 21:32:46 CEST 2012 hasta: Sun Jul 13 21:32:46 CEST 2042

Huellas digitales del Certificado:
MD5: AF:7C:FC:0F:6F:68:C8:F6:40:7E:74:E3:6C:0E:69:FD
SHA1: 9E:80:89:80:E0:54:45:AA:61:FD:38:75:E3:F5:64:08:DB:9F:83:B9
SHA256: 0A:DC:F3:67:D9:91:93:BB:1A:8A:5E:96:12:D0:15:22:5E:72:76:57:B2:
CD:74:8C:7C:97:D4:DE:F3:7E:74:54
Nombre del Algoritmo de Firma: SHA256withRSA
Versión: 3
```

El programa te solicitará una contraseña para proteger el almacén de claves. Deja la contraseña en blanco. De toda la información mostrada nos interesa la huella digital del certificado en codificación SHA1. Como puedes ver en la captura anterior, para nuestro ejemplo está formado por los siguientes bytes:

```
9E:80:89:80:E0:54:45:AA:61:FD:38:75:E3:F5:64:08:DB:9F:83:B9
```

5. Copia en el portapapeles esta secuencia de dígitos. En Windows pulsa con el botón derecho sobre la barra superior de la venta y selecciona *Marcar*. Selecciona el área a copiar. Luego en este mismo menú selecciona *Editar/Copiar*.

6. Para obtener la clave *Google Maps* entra en la siguiente página Web:

<https://code.google.com/apis/console/>

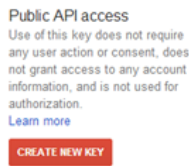
7. Tendrás que introducir un usuario de Google que realiza la solicitud.

8. Crea un nuevo proyecto en esta consola. Para ello pulsa el botón *Create Project...* Introduce como nombre "Ejemplo Google Maps" y pulsa en *Create*.

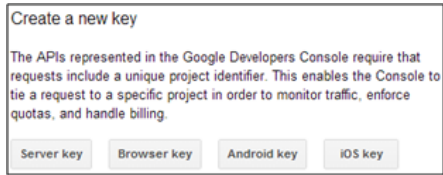
9. A la izquierda aparecerá un menú: selecciona *APIs & auth / APIs*. Localiza el elemento “Google Maps Android API v2” y pulsa en el botón *OFF*. El botón cambiará a *ON*, indicando que vamos a usar esta API.



10. Una vez activado, ve a la opción *APIs & auth / Credentials* y haz click en el botón *CREATE NEW KEY*.

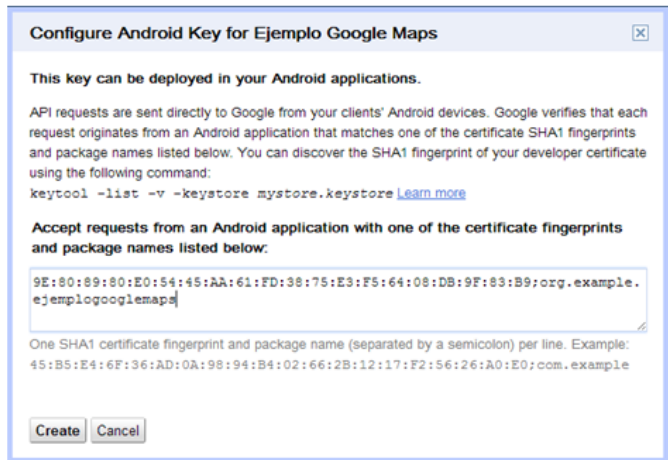


11. Aparecerá la siguiente ventana emergente. Selecciona *Android Key*.

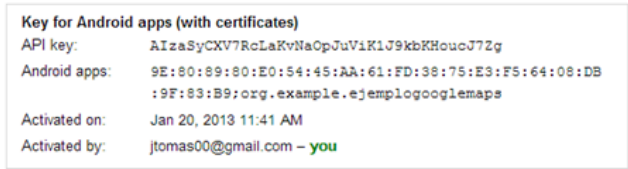


12. En la ventana que aparece, introduce la clave SHA1 obtenida en el paso anterior, seguida de un punto y coma, y del nombre de paquete de vuestra aplicación. En el ejemplo que realizaremos a continuación será:

`;org.example.ejemplogooglemaps`



13. Pulsa el botón *Create*. Aparecerá un nuevo elemento indicando la *API key* que tendrás que usar en la aplicación.



14. Copia en el portapapeles la clave obtenida, la necesitaremos más tarde.



Ejercicio: *Un ejemplo simple con Google Maps*

Veamos un sencillo ejemplo que nos permite visualizar un mapa centrado en las coordenadas geográficas detectadas por el sistema de posicionamiento.

1. Crea un nuevo proyecto con los siguientes datos:

Application Name: Ejemplo Google Maps

Package Name: org.example.ejemplogooglemaps

Minimum Required SDK: API 9: Android 2.3 (Gingerbread)

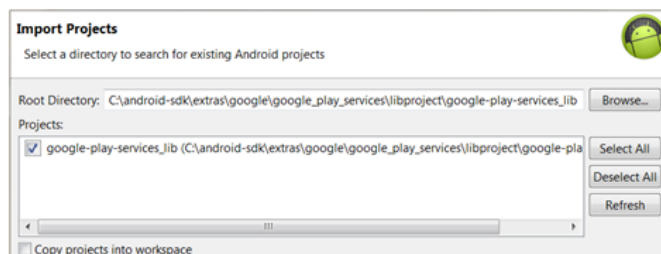
Nota: Aseguraté que el nombre de paquete coincida con el del ejercicio anterior.

2. Abre *Android SDK Manager* y asegúrate que el paquete *Google Play services* y *Google Repository* están instalados. Si hay una versión más reciente actualízala:

Name	Rev.	Status
Android Support Repository	11	Installed
Android Support Library	21.0.3	Installed
Google Play services for Froyo	12	Not installed
Google Play services	22	Installed
Google Repository	15	Installed
Google Play APK Expansion Library	3	Not installed
Google Play Billing Library	5	Not installed

3. Vamos a importar a nuestro proyecto las librerías de *Google Play Services*. Utilizando **Android Studio**, selecciona *File > Project Structure...* y en la columna izquierda pulsa sobre *app*. En la pestaña *Dependencies* y haz clic en el botón más (+) de arriba a la derecha y escoge *Library dependency*. Selecciona *play-services* (com.google.android.gms:play-services:X.X.X) y pulsa *Ok*.

Con **Eclipse**, entra en *File > Import > Android > Existing Android Code Into Workspace*. Pulsa *Browse...* y ve a la carpeta donde está instalado *Android SDK Manager*. Desde esta, selecciona la carpeta */sdk/extras/google/google_play_services/libproject/google-play-services_lib*. Debería aparecer seleccionado un proyecto llamado *google-play-services_lib*. Pulsa *Finish*



Vamos a importar esta librería. Haz clic en el botón derecho y selecciona *Properties > Android*. Al final de la página, en la sección de librerías, pulsa *Add...* y selecciona el proyecto que acabas de importar.

4. En *AndroidManifest.xml*, dentro del elemento `<application>`, añade 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="AlzaSyCXV7RcLaKvNaOpJuViK1J9kbKHoucJ7Zg"/>
```

Reemplaza los caracteres marcados ("*Alza...*") por la *API Key* obtenido en el ejercicio anterior.

- 5 Añade los siguientes permisos dentro de *AndroidManifest* dentro del elemento `<manifest>`:

```

<permission
    android:name="org.example.ejemplogooglemaps.permission.MAPS_RECEIVE"
    android:protectionLevel="signature"/>

<uses-permission
    android:name="org.example.ejemplogooglemaps.permission.MAPS_RECEIVE"/>

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

```

Nota: Fíjate que en los dos primeros elementos se indica el nombre de paquete. En futuros proyectos reemplaza esta información.

6. En *AndroidManifest* añade a continuación este código para indicar que *Google Maps* requiere *OpenGL*:

```

<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true" />

```

7. Reemplaza el contenido del *layout* *activity_main.xml* por:

```

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <fragment
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="com.google.android.gms.maps.SupportMapFragment"/>
</RelativeLayout>

```

NOTA: La etiqueta *<fragment>* ha de escribirse en minúscula.

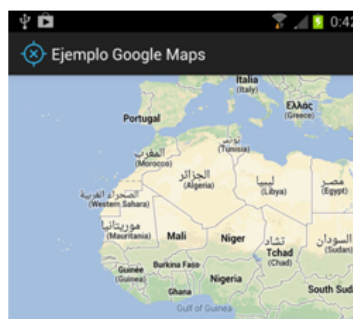
8. Abre *MainActivity.class* y haz que esta clase herede de *FragmentActivity*:

```

public class MainActivity extends FragmentActivity {

```

9. Ejecuta la aplicación en un dispositivo real. En un emulador no va a funcionar dado que la nueva versión de *Google Maps* solo funciona en el dispositivo con *Google Play* instalado. A continuación se muestra el resultado:



NOTA: Es posible que el dispositivo que estás utilizando no tenga instalados los servicios de *Google Play*. En tal caso, la aplicación te pedirá que los instales:



Ejercicio: Introduciendo código en Google Maps

En el ejercicio anterior hemos visto un ejemplo muy básico donde solo se mostraba un mapa con las opciones predeterminadas. En este ejercicio aprenderemos a configurarlo y añadir marcadores desde el código.

1. Abre el *layout activity_main.xml* y añade los siguientes tres botones dentro del `<RelativeLayout>` (tras el `<fragment ...>`):

```
<Button android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_toLeftOf="@+id/button2"
        android:onClick="moveCamera"
        android:text="ir a UPV" />
<Button android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:onClick="animateCamera"
        android:text="a mi posición" />
<Button android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_toRightOf="@+id/button2"
        android:onClick="addMarker"
        android:text="marcador" />
```

2. Sustituye el contenido de *MainActivity.java* por

```

public class MainActivity extends FragmentActivity
    implements OnMapClickListener {
    private final LatLng UPV = new LatLng(39.481106, -0.340987);
    private GoogleMap mapa;

    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mapa = ((SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map)).getMap();
        mapa.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
        mapa.moveCamera(CameraUpdateFactory.newLatLngZoom(UPV, 15));
        mapa.setMyLocationEnabled(true);
        mapa.getUiSettings().setZoomControlsEnabled(false);
        mapa.getUiSettings().setCompassEnabled(true);
        mapa.addMarker(new MarkerOptions()
            .position(UPV)
            .title("UPV")
            .snippet("Universidad Politécnica de Valencia")
            .icon(BitmapDescriptorFactory
                .fromResource(android.R.drawable.ic_menu_compass))
            .anchor(0.5f, 0.5f));
        mapa.setOnMapClickListener(this);
    }

    public void moveCamera(View view) {
        mapa.moveCamera(CameraUpdateFactory.newLatLng(UPV));
    }


    public void animateCamera(View view) {
        if (mapa.getMyLocation() != null)
            mapa.animateCamera(CameraUpdateFactory.newLatLngZoom(
                new LatLng( mapa.getMyLocation().getLatitude(),
                           mapa.getMyLocation().getLongitude()), 15));
    }

    public void addMarker(View view) {
        mapa.addMarker(new MarkerOptions().position(
            new LatLng(mapa.getCameraPosition().target.latitude,
            mapa.getCameraPosition().target.longitude)));
    }

    @Override
    public void onMapClick(LatLng puntoPulsado) {
        mapa.addMarker(new MarkerOptions().position(puntoPulsado).
            icon(BitmapDescriptorFactory
                .defaultMarker(BitmapDescriptorFactory.HUE_YELLOW)));
    }
}

```

Comenzamos declarando dos objetos: UPV hacer referencia a la posición geográfica de la Universidad Politécnica de Valencia y mapa, que nos permitirá acceder al objeto GoogleMap que hemos insertado en un *fragment* de nuestro *Layout*. Este objeto es inicializado al comienzo de `onCreate()` y a continuación se utilizan una serie de métodos para configurarlo. `setMapType()` permite seleccionar el tipo de mapa (normal, satélite, híbrido o relieve). Para averiguar las constantes correspondientes te recomendamos que utilices la opción de autocompletar (escribe `GoogleMap.` y podrás seleccionar las constantes de esta clase). El método `moveCamera()` desplaza el área de visualización a una determinada posición (UPV) a la vez que define el nivel de zoom (15). El nivel de zoom ha de estar en un rango de 2 (continente) hasta 21 (calle). El método `setMyLocationEnabled(true)` activa la visualización de la posición del dispositivo por medio del típico triángulo azul. El método `getUiSettings()` permite configurar las acciones del interfaz de usuario. En este ejemplo se han utilizado dos, desactivar los botones de zoom y visualizar una brújula. Puedes usar autocompletar para descubrir otras posibles configuraciones. El método `addMarker()` permite añadir los típicos marcadores que habrás visto en muchos mapas. ejemplo se indica como posición UPV, un título, una descripción, como icono se

utiliza el mismo *drawable* usado como icono de la aplicación y el punto del icono que haremos coincidir con el punto exacto que queremos indicar en el mapa. Un valor de (0, 0) corresponde a la esquina superior izquierda del icono y (1, 1) a la esquina inferior derecha. Como nuestro icono tiene forma de círculo , hemos indicado el valor (0.5, 0.5) para que coincida con su centro. Finalmente, hemos registrado un escuchador de evento para detectar pulsaciones sobre la pantalla. El escuchador vamos a ser nosotros mismos (this), por lo que hemos implementado el interfaceOnClickListener y añadido el método onMapClick().

A continuación se incluyen los tres métodos que se ejecutarán al pulsar sobre los botones añadidos al *layout*. El primero, moveCamera(), desplaza el punto de visualización a la UPV. A diferencia del uso anterior, sin cambiar el nivel de zoom que el usuario tenga seleccionado.

El segundo, animateCamera(), nos desplaza hasta nuestra posición actual por medio de una animación (similar a la que a veces utilizan en el Tele Diario para mostrar un punto en conflicto). Observa como el método getLocation() permite obtener la posición del dispositivo sin usar el API Android de posicionamiento. Si usas este método verifica siempre que ya se dispone de una posición (!= null) y que has pedido permisos de localización.

El tercero, addMarker(), añade un nuevo marcador en el centro del mapa que estamos observando (getCameraPosition()). En este caso usaremos el marcador por defecto, sin información adicional.

Como hemos indicado onMapClick() será llamado cuando se pulse sobre el mapa. Se pasa como parámetro las coordenadas donde se ha pulsado, que utilizaremos para añadir un marcador. Esta vez el marcador por defecto, es de color amarillo.

3. Ejecuta la aplicación. Se muestra el resultado:



Show Discussion

 New Post

© All Rights Reserved

© 2015 edX Inc.

EdX, Open edX, and the edX and Open edX logos are registered trademarks or trademarks of edX Inc.

POWERED BY
OPENedX