PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Servicios en red III
Servicios de Google
Google Maps
Publicación de una aplicación en la Play Store

Utilización de servicios externos

Es posible **utilizar un servicio externo** e integrarlo en nuestras Apps Android, como ya hemos visto.

Una de las mayores reglas en el desarrollo de software nos dice que si algo ya está inventado y funciona, no lo reinventes y utilízalo.

En las aplicaciones móviles, y en las de escritorio también, podremos utilizar **servicios de terceros** que ya tienen implementados, por ejemplo:

- APIS de inicio de sesión con redes sociales,
- mapas de Google Maps,
- envío de mensajes en tiempo real con Firebase,
- etc.

Es aconsejable buscar un poco antes de intentar crear nuestro propio servicio, ya que puede estar creado o puede haber algo que nos ayude a terminar de implementarlo.

Google Maps

En **2003**, Lars y Jens Rasmussen, junto a los australianos **Noel Gordon y Stephen Ma**, fundaron la empresa Where 2 Technologies, con la que empezaron a mapear la ciudad de Sídney, en **Australia**.

En **2004, Google se interesó por esta idea** y **se la compró** a sus creadores. Así nació Google Maps. Se supone que fue este año 2004 en el que **Google creó Google Maps**.

Oficialmente, Google Maps se anunció en **febrero de 2005** y lo soportarían los navegadores **Internet Explorer y Mozilla Firefox**, ya que, por aquel entonces, no existían los smartphones.

Google Maps utiliza un sistema de coordenadas mediante una latitud y una longitud, que son **positivas para el Norte y el Este** y negativas para el Sur y el Oeste.

El enlace a la web de Google Maps es el siguiente:

> https://www.google.es/maps/preview

Google Maps no solo ofrece servicios en la Tierra, sino que también podremos disfrutar de unas breves zonas en la Luna, mediante **Google Moon**; y en Marte, mediante **Google Mars**.

El enlace a Google Moon es el siguiente:

> https://www.google.com/moon/

El enlace a Google Mars es este:

> https://www.google.com/mars/

Google nos ofrece la posibilidad de usar su API de Google Maps en nuestros dispositivos de una forma relativamente fácil y gratuita.

Complementos de los navegadores

Una cosa a tener en cuenta, es que al estar ofreciendo **servicios HTTPS** en nuestras aplicaciones, puede ser necesario el **visualizar algún sitio web** en nuestro dispositivo móvil.

Es importante saber que los navegadores utilizan ciertos **complementos para visualizar los sitios web** y que éstos pueden estar desactualizados en nuestros dispositivos móviles, así que no hay que olvidar tener todas las **apps y complementos** siempre **actualizados**.

Creación de un proyecto de Google Maps

Para utilizar la API de Google Maps en nuestras aplicaciones Android, debemos realizar algunos **pasos previos**.

En **primer lugar**, para hacer uso de todos los **servicios disponibles de Google Maps**, debemos generar una Clave de API (o API Key) para **asociarla** de forma unívoca a nuestra **aplicación** en concreto. Esto se hace muy rápido: se puede hacer accediendo a la **Consola de Desarrolladores** de Google desde nuestra cuenta de Gmail en el siguiente enlace y, posteriormente, siguiendo los pasos que mostramos seguidamente:

https://console.developers.google.com/apis/dashboard

1. Cuando accedamos, tenemos que crear un proyecto nuevo a partir de la lista desplegable que nos aparecerá arriba a la derecha. Para ello, seleccionamos la opción Crear proyecto.

2. Una vez hecho esto, se nos mostrará una nueva ventana que nos solicitará el nombre del proyecto que vamos a crear. Una vez hayamos introducido algún nombre descriptivo, se va a generar automáticamente un **ID único**. Solo nos queda terminar el proceso pulsando Crear.

3. Cuando tengamos creado nuestro proyecto, lo seleccionamos y pulsamos el enlace **biblioteca de API**, donde podremos seleccionar todas las API que nos ofrece Google y que utilizaremos. En nuestro caso particular, vamos a seleccionar Maps SDK for Android.

4. Ahora necesitaremos crear las credenciales de nuestro proyecto; para ello, pulsamos en Credenciales, que está a la izquierda. Una vez ingresemos en esta pantalla, pulsaremos en Crear credenciales y seleccionaremos **Clave de API**. Esto nos **creará la API** de nuestra aplicación, que permitirá a nuestras aplicaciones Android **usar los mapas** de Google.

Deberemos **guardar la clave de API** que se genere para utilizarla más adelante, aunque siempre podremos **acceder** a ella desde nuestra **cuenta y proyecto**.

Una vez hecho todo esto, tendremos listo nuestro proyecto de Google Maps para utilizarlo desde nuestra aplicación Android.

Cupo de proyectos

Tenemos que tener cuidado cuando creamos **proyectos de Google**, ya que sólo tendremos un cupo aproximado de **19 proyectos de forma gratuita**.

Cuando estamos desarrollando aplicaciones, es relativamente sencillo empezar a crear proyectos sin control para probar servicios, y posteriormente, vernos en la necesidad de tener que eliminar algunos de ellos.

Los registros como desarrollador

Cuando vamos a utilizar servicios de este tipo que ofrece un tercero, es muy normal que debamos registrarnos para obtener un ID de desarrollador. Esta práctica se da **por seguridad** y para poder tener **controlado el uso de los servicios** por parte de la compañía que los proporciona, ya que es un aspecto bastante serio y está **regulado por ley**.

El registro como desarrollador lo vamos a encontrar en todas las grandes empresas que ofrecen servicios de este tipo, como pueden ser Google, para utilizar sus servicios de **Google Maps o YouTube**, **Facebook** (para poder usar el inicio de sesión a través de su plataforma), **Twitter**, etc.

Configuración de un proyecto en Android Studio para utilizar Google Maps

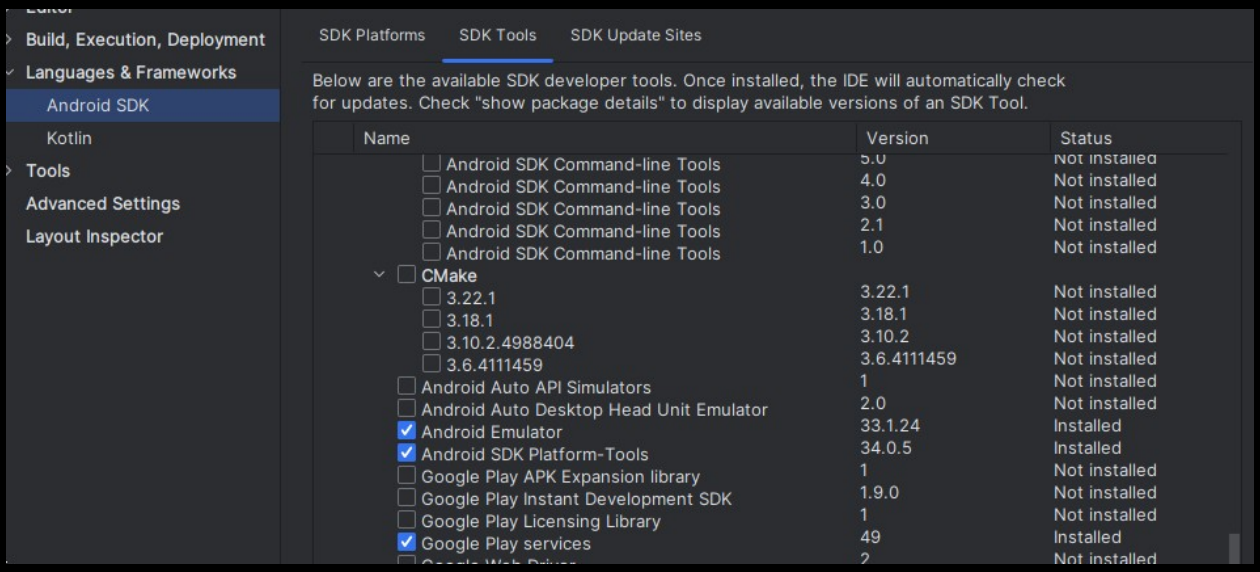
Una vez tenemos configurado nuestro proyecto de Google Maps en la consola de desarrolladores de Google, con la clave de API que nos permitirá su uso, estamos en condiciones de **crear un proyecto en Android Studio** para poder utilizar el servicio de Google Maps.

Lo primero que debemos hacer es instalar (si no la tenemos) la **SDK de Google Play Services** en nuestro Android Studio. Este es un paso importante, ya que, sin esta SDK, no aparecerán los mapas en las aplicaciones del emulador.

Para ello, nos vamos al **gestor de SDK de Android Studio (SDK Manager)** y, en la opción **Android SDK**, seleccionamos SDK Tools.

Aquí nos aparecerá la SDK de **Google Play Services** para instalar (si no está ya instalada).

Una vez hecho esto, estará **instalado para todos nuestros proyectos**, ya que se instala en la SDK de Android Studio.



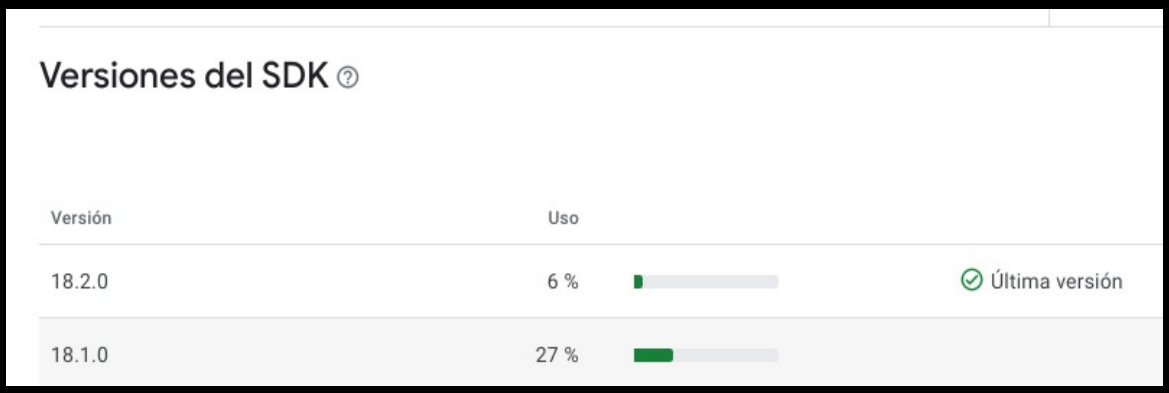
A continuación, en nuestro proyecto deberemos poner las **dependencias de Google Maps** en el fichero **build.gradle** de la siguiente forma:

```
implementation("com.google.android.gms:play-services-maps:18.2.0")
```

Nota: Android Studio marcará que no está actualizada la versión del SDK, deberemos ir a:

> *https://play.google.com/sdks/details/com-google-android-gms-play-services-maps?utm_source=ide&utm_medium=snapshot*

Y en "Versiones del SDK" ver cual es la última versión y cambiar a esa versión en la implementación de arriba. Entonces la advertencia subrayada en rojo se quitará.



Finalmente, modificaremos el fichero **AndroidManifest.xml** para añadir estos tres elementos nuevos:

Nota: gms.version y geo.API_KEY se agregarán dento del elemento <application> ... </application>.

Mientras que glEsVersion se agregará como hijo directo de <manifest>, quedando así la estructura:

Elementos nuevos por añadir

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

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
```

```
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.TestGoogleMapsMedac"
        tools:targetApi="31">
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyCO2Ll1f5ldqVvQIgeUUzJEAs2LKlk4Ep8" />
```

[...]

Ayuda:

https://developers.google.com/maps/documentation/android-sdk/start?hl=es-419

https://developer.android.com/guide/topics/manifest/uses-feature-element?hl=es-419

https://developers.google.com/maps/documentation/android-sdk/config?hl=es-419#kotlin

Agregar un mapa de Google Maps

Para agregar un mapa de Google Maps a nuestra actividad, simplemente tenemos que agregar el **control al Layout** de la actividad donde queramos mostrar el mapa.

Para poder añadirlo, necesitaremos agregar un fragment a la actividad de tipo com.google.android.gms.maps.MapFragment, el cual quedará como se muestra a continuación.

Fragment con un mapa

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.MapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</LinearLayout>
```

Puede ocurrir que no se muestre correctamente, pero puede deberse a algún error de renderizado en el editor visual de Layouts de Android Studio. Con cerrar la pantalla y volverla abrir, se solucionará.

Para poder **obtener la referencia al mapa** en nuestra actividad, vamos a crear, en primer lugar, un **atributo privado** de tipo GoogleMap en la actividad principal. Haremos, además, que nuestra actividad **implemente** la interfaz **OnMapReadyCallback**, lo cual nos permitirá **dar funcionalidad** a nuestro mapa.

Debemos modificar el método onCreate() de la actividad, obteniendo, en primer lugar, una **referencia al MapFragment** que hemos añadido a nuestro Layout a través del fragment manager; seguidamente, llamaremos a su método **getMapAsync**(), pasándole un **objeto que implemente la interfaz OnMapReadyCallback**, en nuestro caso la **propia actividad principal**.

Con esto, conseguimos que, en cuanto esté disponible la referencia al mapa incluido dentro de nuestro MapFragment, se llame automáticamente al método **onMapReady**() de la interfaz. Por el momento, la implementación de este método va a ser muy sencilla, ya que nos limitaremos a guardar el objeto GoogleMap recibido como parámetro en nuestro atributo mapa.

Una vez hecho esto, podremos ejecutar nuestra aplicación y tendremos un mapa funcional.

En el siguiente vídeo, podrás visualizar el proceso completo, así como la opción de cambiar de mapa. En la sección de «Recursos del tema» encontrarás todo el código generado.

Integración de un mapa de Google en nuestra aplicación cambiando el tipo de mapa

En primer lugar deberemos abrir el fichero build.gradle del módulo aplicación, para colocar esta implementación que es la biblioteca de Google Maps.

```
implementation 'com.google.android.gms:play-services-maps:18.2.0'
```

Una vez hagamos esto aquí nos aparecerá una opción de sincronización de nuestro proyecto, la pulsamos y se descargará la biblioteca para poder usarla.

Luego, en el Manifest deberemos de poner las siguientes configuraciones:

- uses-feature que deberá ir justo encima de application.
- Los dos elementos meta-data vistos antes, que uno indica nuestro número de Google Play Services y el otro indica nuestro código de la aplicación de Google Maps en la nube.

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

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

    <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">

        <meta-data
            android:name="com.google.android.gms.version"
```

```
        android:value="@integer/google_play_services_version" />

    <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyCO2Ll1f5ldqVvQIgeUUzJEAs2LKlk4Ep8" />

    <activity
        android:name=".MainActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Ahora vamos a ver la interfaz gráfica:

Podemos ver que tenemos:

- un fragment con un mapa,
- un RadioGroup con cuatro RadioButtons. Estos serán los cuatro tipos de mapas que podremos poner. Pulsando sobre cada uno se cambiará el tipo de mapa mostrado, y se descargará dicho mapa.

En nuestra aplicación, en la clase MainActivity, deberemos de:

1. crear un objeto de tipo Google Maps y otro objeto MapFragment a los que haremos un FindViewById.
2. implementar las interfaces OnMapReadyCallBack y GoogleMap.OnMapClickListener, además de la del RadioButton, para saber qué RadioButton está activo.

Clase MainActivity del proyecto

```
public class MainActivity extends AppCompatActivity implements OnMapReadyCallback,
RadioGroup.OnCheckedChangeListener, GoogleMap.OnMapClickListener{

    private GoogleMap mapa;
    private MapFragment mapFragment;
    private RadioGroup rgTipoMapa;

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

        // ***** Ligamos los recursos de la actividad *****
        mapFragment = (MapFragment) getFragmentManager().findFragmentById(R.id.map);
        rgTipoMapa = (RadioGroup) findViewById(R.id.rgTipoMapa);
        // ************************************************

        mapFragment.getMapAsync(this);
        rgTipoMapa.setOnCheckedChangeListener(this);
    }
```

En el método OnMapReady:

Deberemos **igualar nuestro** mapa al mapa que ya está **creado**, que es el que le llega, y deberemos indicar que el **listener** del click es **nuestra propia clase (this)**.

```
@Override
public void onMapReady(GoogleMap map)
{
    mapa = map;
    mapa.setOnMapClickListener(this);
}
```

Cuando cambiemos y pulsemos sobre los RadioButton haremos lo siguiente:

- Para cambiar a una vista normal, con el método SetMapType deberemos pasar la variable MAP_TYPE_NORMAL.
- Para una vista en satélite MAP_TYPE_SATELLITE.
- Para una vista en híbrido que será una combinación de normal y satélite será MAP_TYPE_HYBRID.
- Y para una vista **topográfica** será MAP_TYPE_TERRAIN.

Todas ellas son variables finales de la clase Google.Map.

```
    @Override
    public void onCheckedChanged(RadioGroup radioGroup, @IdRes int checkedId)
    {
        switch(radioGroup.getId())
        {
            case R.id.rgTipoMapa:
                switch(checkedId)
                {
                    case R.id.rbNormal:
                        mapa.setMapType(GoogleMap.MAP_TYPE_NORMAL);
                        break;
                    case R.id.rbSatelite:
                        mapa.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
                        break;
                    case R.id.rbHibrido:
                        mapa.setMapType(GoogleMap.MAP_TYPE_HYBRID);
                        break;
                    case R.id.rbTopografico:
                        mapa.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
                        break;
                }
                break;
        }
    }

    @Override
    public void onMapClick(LatLng latLng) {

    }
}
```

Agregar marcadores en Google Maps

Una de las acciones más comunes que podemos realizar en Google Maps es usar marcadores. Con los marcadores, podremos **dejar seleccionada una ubicación** en el mapa, con la que podremos interactuar.

Las coordenadas que usa Google Maps se basan en una latitud y en una longitud, que deberemos conocer para poder indicarle que coloque un marcador en dicha ubicación.

Si quieres averiguar las coordenadas GPS de cualquier lugar del mundo, pincha en el siguiente enlace:

http://www.coordenadas-gps.com/

Para **responder a los clics del usuario** sobre el mapa, definiremos el **evento** onMapClick(), llamando al **método** setOnMapClickListener() de nuestro mapa.

Dentro de este, recibiremos un **objeto LatLng** con la **posición geográfica**.

Para **obtener esta posición** usaremos el método toScreenLocation() del *objeto* Projection que podemos **obtener a partir del mapa** llamando a getProjection().

Una vez sabemos cómo podemos obtener la posición, podremos **agregar un marcador** a dicha posición, mediante el método addMarker, al que le deberemos pasar un objeto del tipo **LatLng** con las **coordenadas** y un **título** para identificar el marcador al pulsar en él. También podremos agregar un marcador en el lugar que deseemos, utilizando la página web que nos proporciona las coordenadas de una dirección cualquiera.

En el siguiente vídeo, podrás visualizar. En la sección de «Recursos del tema» encontrarás todo el código generado.

Cómo se agregan marcadores, paso a paso

Vamos a ver cómo podemos agregar marcadores a un mapa de Google.

Una vez que ya hemos **configurado nuestra aplicación** para que pueda utilizar la biblioteca de Google Maps y esté configurado el **fichero Manifest**, en nuestra interfaz gráfica tendremos lo siguiente:

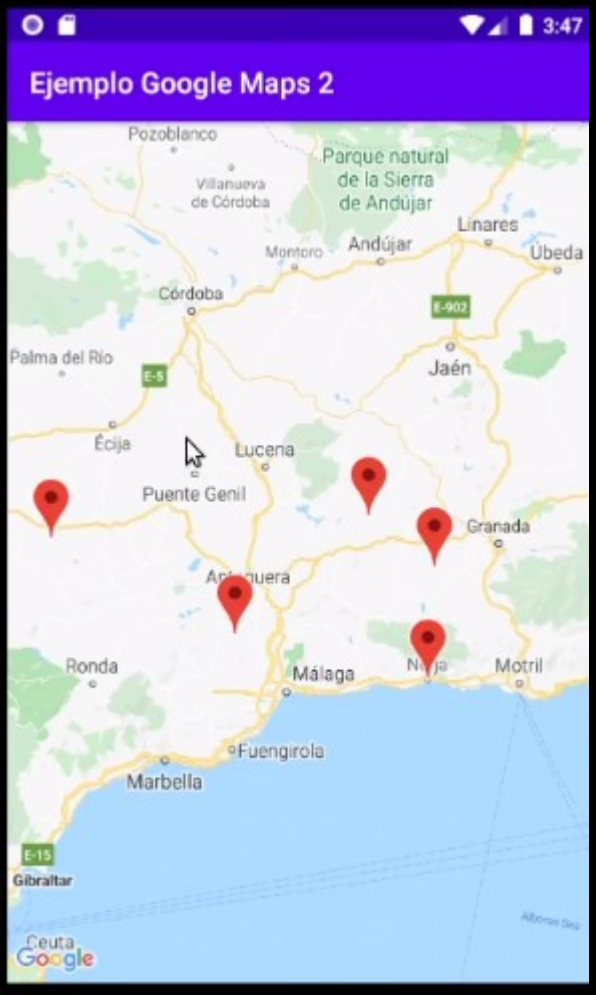
Un mapa de Google en el que lo que vamos a hacer es que cuando pulsemos sobre una localización se agregue automáticamente un marcador. Para ello:

En la clase ActivityMain haremos lo siguiente:

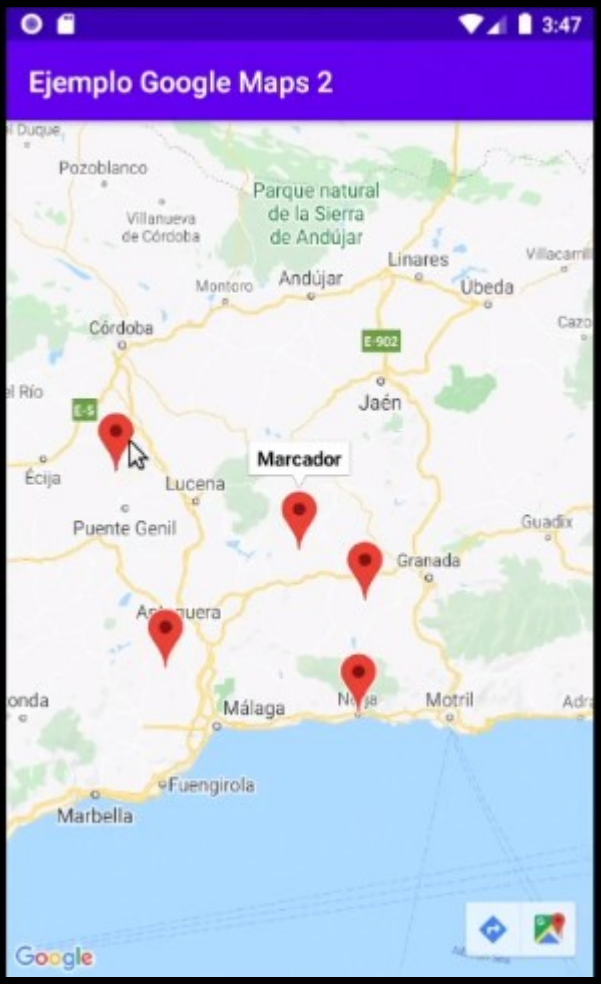
Crearemos un **Google Map** y un **MapFragment**. Hemos creado un **método insertarMarcador** que inserta un marcador en un punto dado con una latitud y una longitud las cuales son sus coordenadas. Para ello vamos a utilizar el **método addMarker** al que le pasaremos la posición en coordenadas y un título, que siempre será en este caso “Marcador”. Como hemos implementado la **interfaz onMapClickListener** de GoogleMaps, usaremos el

método sobrescrito **onMapClick** que hará que cuando hagamos clic sobre el mapa se ejecute una acción. Esa acción será la de **insertar el marcador** ya que cuando hagamos clic, por defecto se nos dará la latitud y longitud de donde hemos pulsado.

Vamos a probar nuestra aplicación. Aquí tenemos un mapa de Google que haciendo doble clic se nos da la latitud y longitud de donde hemos pulsado. Con un simple clic podremos acercarnos a una localización, por ejemplo a España. Ahora si hacemos clic sobre una localización se agregará un marcador donde hemos hecho el clic. Podremos agregar tantos marcadores como queramos.



Si pulsamos sobre un marcador seguirá el foco a dicho marcador y aparecerá el título. En este caso todos tendrán el mismo título.



MainActivity implementando la interfaz onMapClickListener

```
public class MainActivity extends AppCompatActivity implements OnMapReadyCallback,
GoogleMap.OnMapClickListener {

    private GoogleMap mapa;
    private MapFragment mapFragment;

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

        // ***** Ligamos los recursos de la actividad *****
        mapFragment = (MapFragment) getFragmentManager().findFragmentById(R.id.map);
        // ************************************************

        mapFragment.getMapAsync(this);
    }

    @Override
    public void onMapReady(GoogleMap map)
    {
        mapa = map;
        mapa.setOnMapClickListener(this);
    }

    @Override
    public void onMapClick(LatLng latLng)
    {
        insertarMarcador(latLng);
    }

    /**
     * Inserta un marcado en un Google Maps
     * @param coordenadas Coordenadas para insertar dicho marcador
     */
    private void insertarMarcador(LatLng coordenadas)
    {
        mapa.addMarker(new MarkerOptions()
                .position(coordenadas)
                .title("Marcador"));
    }
}
```

Nuevos servicios

Enviar mensajes instantáneos entre dispositivos móviles.

Uno de los servicios más amplios es el de FireBase. Con este servicio, podremos **enviar mensajes** entre diferentes dispositivos móviles, como lo hacen las aplicaciones de WhatsApp, Telegram, Messenger, etc. Podremos crear **bases de datos en tiempo real** en la nube, así como otras operaciones también en tiempo real, y muchos más servicios que pueden hacer que nuestras aplicaciones crezcan de forma exponencial.

Por ejemplo, mediante Firebase, podremos enviar mensajes de **texto y multimedia**, con imágenes, entre usuarios, tal y como podemos hacer con aplicaciones como WhatsApp o Telegram. Aquí deberíamos **dar permisos a nuestra aplicación**, tanto para poder **conocer el ID** de nuestro dispositivo como para poder **acceder a las fotografías** del mismo, en el caso de enviar mensajes multimedia.

Página web:

https://firebase.google.com/?hl=es

Huawei, información y entornos de pruebas:

Push: https://developer.huawei.com/consumer/en/hms/huawei-pushkit/

Analytics: https://developer.huawei.com/consumer/en/hms/huawei-analyticskit/

Location: https://developer.huawei.com/consumer/en/hms/huawei-locationkit/

Maps: https://developer.huawei.com/consumer/en/hms/huawei-MapKit/

Anuncios: https://developer.huawei.com/consumer/en/hms/huawei-adskit

Account: https://developer.huawei.com/consumer/en/hms/huawei-accountkit

Publicación de una aplicación en la Play Store de Google

Para poder publicar nuestras aplicaciones en Google Play, deberemos registrarnos en la **Consola para Desarrolladores de Google**. Esto supone un coste de unos **25 dólares** que deberemos pagar mediante tarjeta de crédito y tendremos una **licencia de por vida** para poder subir todas las aplicaciones que queramos. Para esto, solo necesitamos una **cuenta en Gmail** y acceder a la siguiente página web, para registrarnos y **realizar el pago**:

https://play.google.com/apps/publish

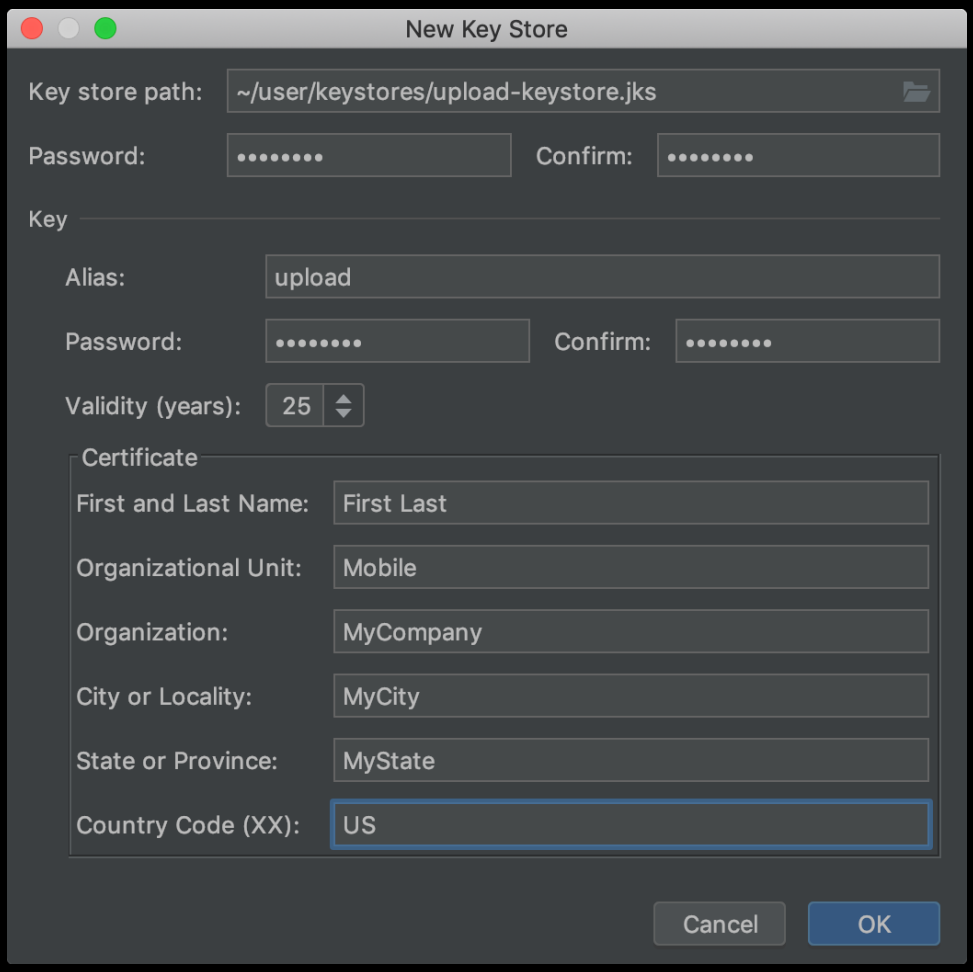
Tendremos que seguir **cuatro pasos**:

1. Iniciar sesión con nuestra **cuenta de Gmail**, la cual quedará vinculada y **será** la **cuenta de desarrollador**.
2. **Aceptar** el acuerdo.
3. Realizar el **pago**.
4. Rellenar la **información** pertinente.

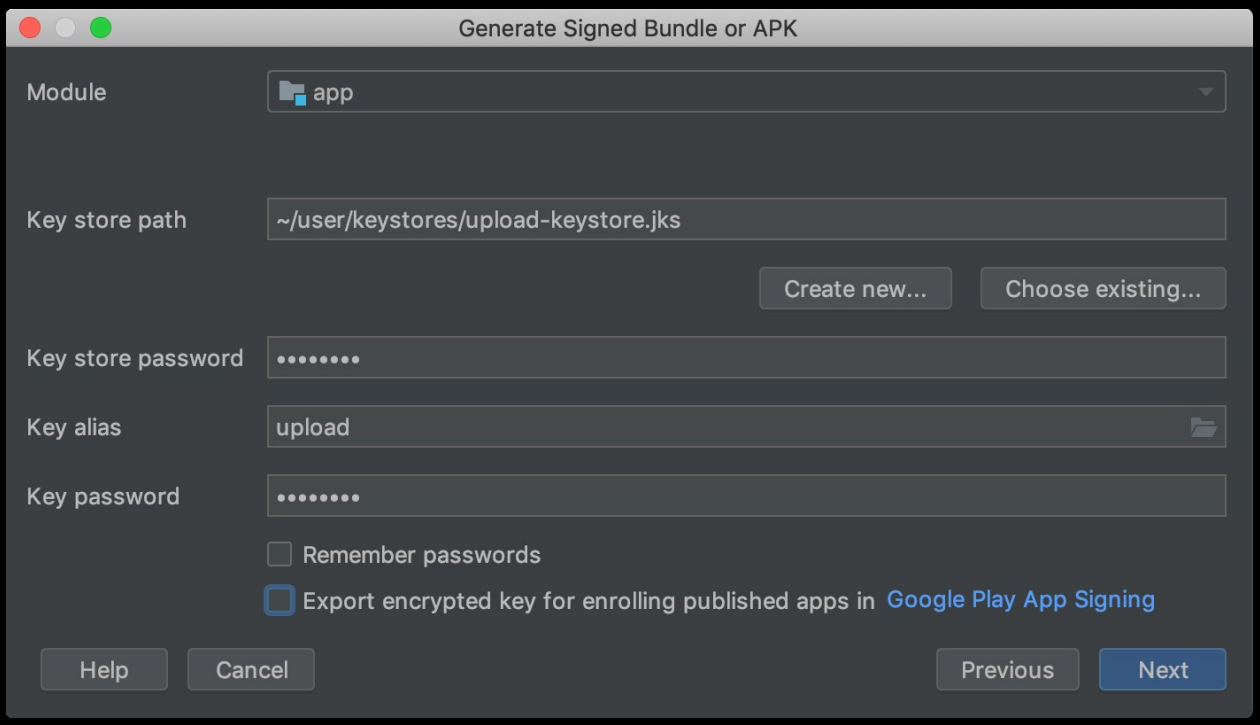
Para poder subir las aplicaciones, deberemos crear un APK desde Android Studio. Previamente, deberemos crearnos una firma para nuestras aplicaciones, lo cual podremos hacer también desde Android Studio. Para esto último:

https://developer.android.com/studio/publish/app-signing?hl=es-419

1. Pulsaremos en el **menú Run** (menú Build en actual versión Hedgehog -Erizo- | 2023.1.1), y **Generate Signed Bundle/APK**. Elegiremos la **opción APK** y, en la siguiente pantalla, podremos crear una Key Store o firma pulsando en **Create New**. **Si ya disponemos** de una, pulsamos **Choose existing**. Esta firma nos identificará como el propietario de la aplicación (estas solo se podrán subir si están firmadas).



(Crea una clave de carga y un almacén de claves nuevos en Android Studio)



(Firma la app con la clave de carga)

2. Terminamos de **crear nuestra APK** y, una vez la tengamos, nos dirigimos a la Consola de Desarrollador. Pulsaremos en **Crear una aplicación** y nos aparecerá una pantalla para elegir el **idioma** por defecto y el **título** de la misma.
3. Una vez **creemos la aplicación**, podremos **subir nuestro APK**, el cual pasará a revisión por parte del equipo de Google y, si todo está bien, se publicará en unas horas.
4. Dentro de la configuración de nuestra aplicación, podremos **traducirla** a otros idiomas, subir **capturas** de pantalla, subir el **icono** de la misma, etc.

Podremos tener todas las aplicaciones que queramos, siempre y cuando no infrinjan ninguna ley actual.