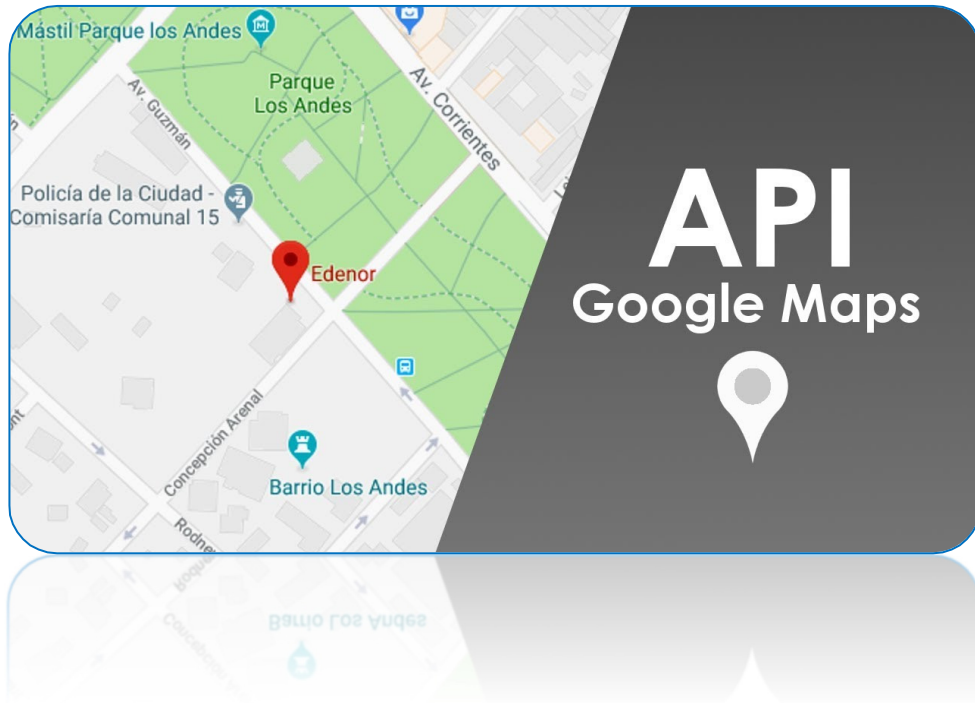


UNIDAD 5: GOOGLE MAPS API



Google Maps es una de las herramientas de cartografía y navegación más utilizada en todo el mundo, ofreciendo mapas actualizados, diferentes tipos de vistas (normal, satélite, terrestre, etc) y muchas funciones como gestión de marcadores, cálculo de rutas, etc.

Mediante la API de Google Maps para Android vamos a poder integrar estas funcionalidades en nuestras aplicaciones Android de una forma muy sencilla, ya que en Android Studio contamos con un tipo de proyecto específico para ello: *Google Maps Activity*.

Más información:

<https://developer.android.com/training/location?hl=es-419>

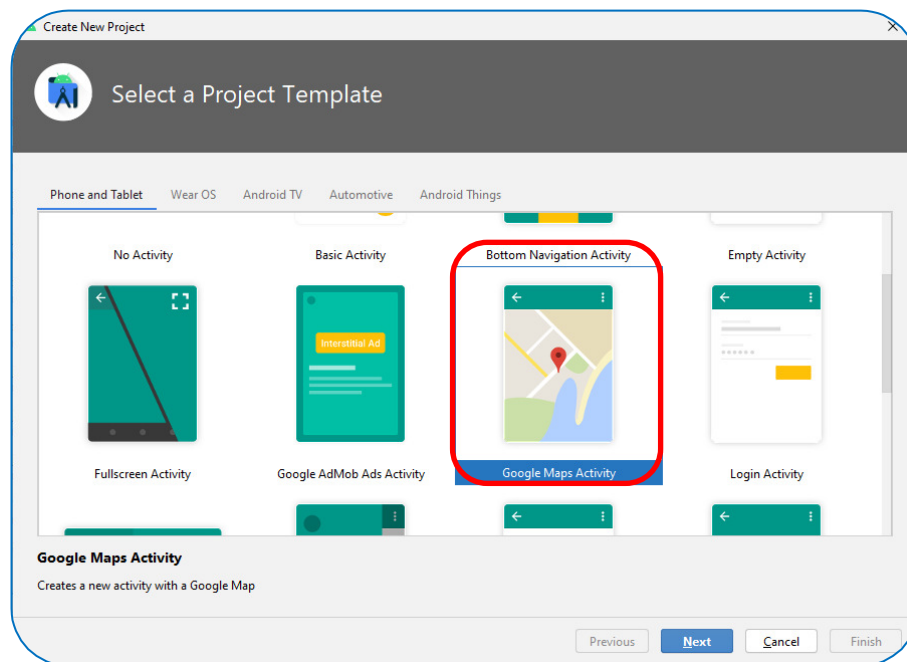
<https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap>

<https://developer.android.com/reference/android/location/Geocoder>

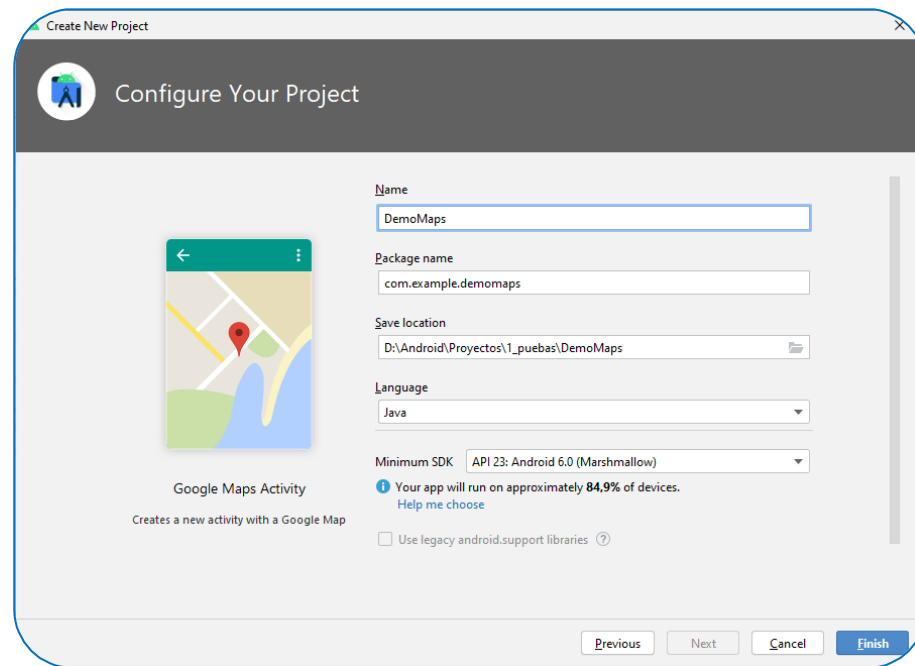
<https://www.codota.com/code/java/classes/android.location.Geocoder>

CREAR UN PROYECTO CON GOOGLE MAPS

Desde la pantalla de inicio de *Android Studio* seleccionamos "*Create New Project*", y en el tipo de proyecto seleccionamos "*Google Maps Activity*".



Por último, establecemos el nombre de nuestro proyecto y la ubicación donde queremos guardarlo.



GENERAR GOOGLE MAPS API KEY

La conexión de nuestro proyecto a *Google Maps* requiere de una clave personal (API key) asociada a una cuenta de Google, que se utilizará para la facturación, si bien el uso de la SDK de Android es gratuito.

Precios			
Recibe 200 USD de crédito gratis todos los meses, que puedes gastar en Maps, Routes y Places			
MAPS	ROUTES	PLACES	
Static Maps Muestra mapas como imágenes		GRATIS PARA MÓVILES	2 USD FOR 1000 SOLICITUDES
APIs	PRECIO POR 1000 SOLICITUDES	USO	COSTE MENSUAL
SDK de Maps para Android	GRATIS	Cargas ilimitadas	0 USD
API Maps Static	2 USD	Usd	-

Tras crearse el proyecto automáticamente se mostrará el archivo de configuración de la API key: "google_maps_api.xml".

En dicho archivo obtendremos el enlace al que tendremos que acceder para obtener la clave, que tendrá la siguiente forma:

https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=SHA-1_Certificate_Fingerprint%3BNombre_Paquete

Tras abrir dicho enlace en el navegador e iniciar sesión con nuestra cuenta de Google, accederemos al panel de Google APIS, donde podremos crear un proyecto existente si ya disponemos de uno con clave API, pero como no es el caso crearemos uno nuevo:

Registra tu aplicación para utilizar Maps SDK for Android en Consola de APIs de Google

La Consola de APIs de Google te permite administrar tu aplicación y supervisar el uso de la API.

Selecciona un proyecto en el que registrar la aplicación

Puedes utilizar un proyecto para gestionar todas tus aplicaciones o crear un proyecto diferente para cada una de ellas.

Crear proyecto

Continuar

Una vez creado el proyecto, se nos mostrará una pantalla desde la que podremos crear la clave:


La API está habilitada

El proyecto se ha creado y Undefined parameter - API_NAMES se ha habilitado.

A continuación, tienes que crear una clave de API para llamar a la API.

Crear clave de API

Por último, copiamos la clave generada y la pegamos en el archivo "google_maps_api.xml", en el espacio donde pone "YOUR_KEY_HERE"

Claves de API				
<input type="checkbox"/>	Nombre	Fecha de creación ↓	Restricciones	Clave
<input checked="" type="checkbox"/>	Clave de API 1	8 feb 2021	Aplicaciones para Android	AIzaSyAmsv...S4yYYKn2zU 

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

MAPSACTIVITY

El archivo "MapsActivity" generado automáticamente cuenta con dos métodos:

- *onCreate*: se encarga de obtener el mapa y mostrarlo en la vista dentro de un fragmento.
- *onMapReady*: permite realizar operaciones sobre el mapa una vez que éste se ha generado.

En el código se crea una localización (LatLng) de ejemplo con las coordenadas de Sydney, se añade un marcador en el mapa y se mueve la cámara hacia esa posición.

```
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);  
    }  
  
    @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));  
    }  
}
```

Podemos fijar la posición que queramos en el mapa sustituyendo las coordenadas del ejemplo por las el punto o ciudad que queramos mostrar, por ejemplo Paris.



Configuración del aspecto del mapa

Podemos modificar las propiedades del mapa para adaptarlo a nuestras necesidades.

Por ejemplo, podemos establecer el tipo de mapa utilizando el método *setMapType* con una de las siguientes constantes:

- MAP_TYPE_NONE
- MAP_TYPE_NORMAL
- MAP_TYPE_SATELLITE
- MAP_TYPE_HYBRID
- MAP_TYPE_TERRAIN

```
mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
```

También podemos establecer el nivel de zoom, utilizando los métodos *setMinZoomPreference* y *setMaxZoomPreference*:

```
mMap.setMinZoomPreference(5);  
mMap.setMaxZoomPreference(10);
```



Más información sobre la clase GoogleMap:

<https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap>

Mostrar nuestra ubicación en el mapa

Para poder mostrar nuestra ubicación debemos tener permiso, para lo que deberemos añadir a nuestro archivo de manifiesto:

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

Y antes de intentar de acceder a dicha localización deberemos asegurarnos de que disponemos del permiso:

```
public void mostrarUbicacion() {  
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)  
        != PackageManager.PERMISSION_GRANTED &&  
        ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION)  
        != PackageManager.PERMISSION_GRANTED) {  
        ActivityCompat.requestPermissions(MapsActivity.this,  
            new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_LOCATION);  
    } else {  
        mMap.setMyLocationEnabled(true);  
    }  
}  
  
public void onRequestPermissionsResult(int rc, @NonNull String[] per, @NonNull int[] res) {  
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
    switch (rc) {  
        case REQUEST_LOCATION:  
            if (per.length>0 && res[0] == PackageManager.PERMISSION_GRANTED) {  
                mostrarUbicacion();  
            }  
            break;  
    }  
}
```



Obtener una posición a partir de su dirección

La clase *GeoCoder* nos permite transformar una dirección en sus coordenadas y viceversa.

En este ejemplo vamos a obtener las coordenadas a partir de una dirección y vamos a crear un marcador en la posición obtenida.

Para ello utilizaremos el método *getFromLocationName*, al que hay que facilitarle un *String* con la dirección buscada y el número de resultados que queremos obtener, y nos devolverá una lista de objetos *Address*, a partir de los cuales podremos obtener las coordenadas: *getLatitude* y *getLongitude*.

```
// Añadir un marcador en una dirección concreta
String direccion = "Avenida el Romeral 12, Villabalter, Spain";
String nombre = "IES San Andres";
LatLng pos = obtenerPosicion(this, direccion);
if (pos != null) {
    mMap.addMarker(new MarkerOptions().position(pos).title(nombre));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(pos));
    mMap.setMinZoomPreference(12);
}

public LatLng obtenerPosicion (Context context, String strAddress) {
    Geocoder coder = new Geocoder(context);
    List<Address> address;
    LatLng pos = null;
    try {
        address = coder.getFromLocationName(strAddress, 5);
        if (address != null) {
            Address location = address.get(0);
            location.getLatitude();
            location.getLongitude();
            pos = new LatLng(location.getLatitude(), location.getLongitude() );
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return pos;
}
```

