# Manejo de Google Maps en Android Studio con Java

## 1. Configuración Inicial del Proyecto

Antes de escribir código Java, debes configurar tu proyecto y obtener una clave de API.

### A. Agregar la Dependencia

En el archivo app/build.gradle, agrega la dependencia de Google Play Services para Maps.

Groovy

```
dependencies {
    // ... otras dependencias
    implementation 'com.google.android.gms:play-services-maps:18.2.0' // Usa la versión estable actual
}
```

Sincroniza el proyecto con Gradle.

### B. Obtener y Configurar la Clave API

1. **Obtener la Clave:** Ve a Google Cloud Console, crea un nuevo proyecto, habilita la **"Maps SDK for Android"** y genera una **Clave API**.
2. **Restringir la Clave:** Por seguridad, restringe la clave para que solo funcione con tu aplicación (usando el nombre del paquete y la huella digital SHA-1).
3. **Configurar en AndroidManifest.xml:** Agrega la clave API dentro de la etiqueta <application>.

XML

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

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

        <uses-library android:name="com.google.android.maps" android:required="false" />

        </application>
</manifest>
```

---

## 2. Implementación de la Interfaz (XML)

El mapa se carga en una Fragment especial proporcionada por el SDK.

En tu archivo activity_main.xml (o donde quieras mostrar el mapa):

XML

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

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</FrameLayout>
```

---

## 3. Manejo del Mapa en Java (Activity)

En tu Activity, debes implementar la interfaz **OnMapReadyCallback** para asegurarte de que el mapa esté completamente inicializado antes de interactuar con él.

Java

```java
// MainActivity.java
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
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 MainActivity extends AppCompatActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

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

        // Obtener el SupportMapFragment y notificar cuando el mapa esté listo.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
```

```
    /**
     * Este método se llama cuando el mapa está listo para ser utilizado.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // 1. Definir una ubicación (ej. Caracas, Venezuela)
        LatLng caracas = new LatLng(10.5000, -66.9000);

        // 2. Agregar un marcador a la ubicación
        mMap.addMarker(new MarkerOptions()
                .position(caracas)
                .title("Marcador en Caracas"));

        // 3. Mover la cámara a la ubicación (con un nivel de zoom)
        // Zoom 15 es un buen nivel para calles.
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(caracas, 15));

        // 4. Opcional: Configurar tipo de mapa
        // mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
    }
}
```

---

## 4. Componentes Clave de Google Maps

Para interactuar con el mapa, utilizas principalmente estas clases:

| Clase Java | Descripción | Ejemplo de Uso |
|---|---|---|
| **GoogleMap** | El objeto central. Permite interactuar con la cámara, dibujar formas, añadir marcadores, y cambiar el tipo de mapa. | mMap.setMapType(...) |
| **LatLng** | Representa un punto geográfico en la Tierra, definido por **latitud y longitud**. | new LatLng(10.5, -66.9) |
| **MarkerOptions** | Define las propiedades de un marcador (posición, título, icono) que se añadirá al mapa. | mMap.addMarker(new MarkerOptions()...) |
| **CameraUpdateFactory** | Proporciona métodos para mover la cámara (zoom, desplazamiento) de forma suave o inmediata. | mMap.moveCamera(...) |
| **SupportMapFragment** | El fragmento de la UI que aloja y muestra el mapa. | Obtenerlo con getSupportFragmentMana |

| | | |
|---|---|---|
| | | ger() |

---

## 5. Interactividad: Marcadores y Clics

Puedes añadir *listeners* para responder a las interacciones del usuario, como hacer clic en un marcador o en un punto del mapa.

Java

```
// Dentro del método onMapReady(GoogleMap googleMap)

// Listener para cuando el usuario hace clic en el mapa
mMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {
    @Override
    public void onMapClick(LatLng latLng) {
        // Elimina el marcador anterior
        mMap.clear();

        // Agrega un nuevo marcador en la posición del clic
        mMap.addMarker(new MarkerOptions()
                .position(latLng)
                .title("Nueva Ubicación"));

        // Mueve la cámara hacia el punto
        mMap.animateCamera(CameraUpdateFactory.newLatLng(latLng));
    }
});

// Listener para cuando el usuario hace clic en un marcador
mMap.setOnMarkerClickListener(new GoogleMap.OnMarkerClickListener() {
    @Override
```

```
    public boolean onMarkerClick(Marker marker) {
        // Muestra un Toast con el título del marcador
        Toast.makeText(MainActivity.this, marker.getTitle(), Toast.LENGTH_SHORT).show();
        // Devolver 'false' permite que se ejecute el comportamiento por defecto (mostrar ventana de info)
        return false;
    }
});
```

---

## 6. Mostrar la Ubicación del Usuario

Para mostrar el punto azul que indica la ubicación actual del usuario, necesitas dos cosas:

1. El permiso ACCESS_FINE_LOCATION en el *Manifest* (ya incluido en el paso 1).
2. Verificar y solicitar el permiso en **tiempo de ejecución** (si aún no lo has hecho).

Java

```
// Dentro del método onMapReady(GoogleMap googleMap)

if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
    // Si el permiso está concedido, habilita la capa de ubicación
    mMap.setMyLocationEnabled(true);
} else {
    // Solicitar permiso de ubicación (usando la lógica de runtime de la presentación anterior)
    // ActivityCompat.requestPermissions(...)
}
```

Con estos pasos, tendrás un mapa funcional en tu aplicación Android, listo para ser enriquecido con marcadores, polilíneas y servicios de localización.