

# Desarrollo Android

---

Arkaitz Garro



# LBS

Location Based Services

# Location Based Services

Es un termino utilizado para describir diferentes tecnologías disponibles para determinar la localización de un dispositivo

Los dos principales componentes son:

**Location Manager:** gestiona los servicios basados en localización.

**Location Providers:** cada uno de los proveedores representa las diferentes maneras de geolocalizar al dispositivo.

```
String serviceString = Context.LOCATION_SERVICE;  
LocationManager locationManager;  
locationManager = (LocationManager) getSystemService (serviceString);
```

# Location Based Services

Antes de utilizar estos servicios, debemos solicitar los permisos de geolocalización

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

# Obtener un Location Provider

La mejor técnica es especificar nuestros requisitos, y que sea Android quien decida que tecnología utilizar.

Para ello utilizamos la clase Criteria, a la cual le indicamos nuestros requisitos.

```
Criteria criteria = new Criteria();  
criteria.setAccuracy(Criteria.ACCURACY_COARSE);  
//criteria.setHorizontalAccuracy(Criteria.ACCURACY_HIGH);  
criteria.setPowerRequirement(Criteria.POWER_LOW);  
criteria.setAltitudeRequired(false);  
criteria.setSpeedRequired(false);
```

```
String bestProvider = locationManager.getBestProvider(criteria, true);
```

```
LocationProvider provider = locationManager.getProvider(bestProvider);
```

# Obtener la posición actual

## Obtener la última posición conocida

```
String provider = LocationManager.GPS_PROVIDER;  
Location location = locationManager.getLastKnownLocation(provider);
```

# Obtener la posición actual

En la mayoría de los casos, obtener la última posición conocida no es suficiente.

Debemos registrar un escuchador de posición que nos notifique cuando se ha producido un cambio en la posición.

# Obtener la posición actual

```
String provider = LocationManager.GPS_PROVIDER;
int t = 5000; // milliseconds
int distance = 5; // meters

LocationListener locListener = new LocationListener() {
    public void onLocationChanged(Location location) {
        // Update application based on new location.
    }

    public void onProviderDisabled(String provider){
        // Update application if provider disabled.
    }

    public void onProviderEnabled(String provider){
        // Update application if provider enabled.
    }

    public void onStatusChanged(String provider, int status, Bundle extras){
        // Update application if provider hardware status changed.
    }
};
locationManager.requestLocationUpdates(provider, t, distance, locListener);
```



# Obtener la posición actual

```
String provider = LocationManager.GPS_PROVIDER;
int t = 5000; // milliseconds
int distance = 5; // meters

final int locationUpdateRC = 0;
int flags = PendingIntent.FLAG_UPDATE_CURRENT;

Intent intent = new Intent(this, MyLocationUpdateReceiver.class);
PendingIntent pendingIntent =
    PendingIntent.getBroadcast(this, locationUpdateRC, intent, flags);

locationManager.requestLocationUpdates(provider, t, distance, pendingIntent);
```

# Obtener la posición actual

```
public class MyLocationUpdateReceiver extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        String key = LocationManager.KEY_LOCATION_CHANGED;  
        Location location = (Location)intent.getExtras().get(key);  
        // TODO [... Do something with the new location ...]  
    }  
}
```

Ejercicio

# Alertas de proximidad

Es posible disparar alertas cuando el dispositivo se aproxima a un área definida por nosotros.

Esta alerta es lanzada tanto al “entrar” como al “salir” del área.

Android lanza un Intent definido junto con la alerta.

# Alertas de proximidad

```
private static final String TREASURE_PROXIMITY_ALERT = "com.paad.treasurealert";

private void setProximityAlert() {
    String locService = Context.LOCATION_SERVICE;

    LocationManager locationManager;
    locationManager = (LocationManager) getSystemService(locService);

    double lat = 73.147536;
    double lng = 0.510638;
    float radius = 100f; // meters
    long expiration = -1; // do not expire

    Intent intent = new Intent(TREASURE_PROXIMITY_ALERT);
    PendingIntent proximityIntent =
        PendingIntent.getBroadcast(this, -1, intent, 0);

    locationManager.addProximityAlert(lat, lng, radius, expiration,
        proximityIntent);
}
```

# Alertas de proximidad

```
public class ProximityIntentReceiver extends BroadcastReceiver {  
    @Override public void onReceive (Context context, Intent intent) {  
        String key = LocationManager.KEY_PROXIMITY_ENTERING;  
        Boolean entering = intent.getBooleanExtra(key, false);  
        // TODO [ ... perform proximity alert actions ... ]  
    }  
}
```

# Geocoder

El API de geolocalización proporcionado por Google permite realizar traducciones entre direcciones y coordenadas.

Dispone de las siguientes funciones:

**Forward geocoding:** busca la latitud y longitud de una dirección.

**Reverse geocoding:** busca la dirección para una latitud y longitud dada.

```
<uses-library android:name="com.google.android.maps"/>  
<uses-permission android:name="android.permission.INTERNET"/>
```

# Reverse Geocoding

```
bool geocoderExists = Geocoder.isPresent();

private void reverseGeocode(Location location) {
    double latitude = location.getLatitude();
    double longitude = location.getLongitude();

    List<Address> addresses = null;
    Geocoder gc = new Geocoder(this, Locale.getDefault());
    try {
        addresses = gc.getFromLocation(latitude, longitude, 10);
    } catch (IOException e) {
        Log.e(TAG, "IO Exception", e);
    }
}
```



# Forward Geocoding

```
bool geocoderExists = Geocoder.isPresent();

private void reverseGeocode(streetAddress) {
    // String streetAddress = "160 Riverside Drive, New York";

    List<Address> locations = null;
    Geocoder gc = new Geocoder(this, Locale.US);
    try {
        locations = gc.getFromLocationName(streetAddress, 5);
    } catch (IOException e) {
        Log.e(TAG, "IO Exception", e);
    }
}
```

# Mapas

## Algunos componentes a tener en cuenta:

**MapView:** la interfaz de usuario que muestra el mapa.

**MapActivity:** clase que muestra el **MapView**. Gestiona el ciclo de vida y los servicios asociados.

**Overlay:** clase utilizada para dibujar sobre el mapa.

**MapController:** controlar la vista del mapa (zoom, centrado...).

**MyLocationOverlay:** **Overlay** especial utilizado para mostrar la posición actual.

**ItemizedOverlays y OverlayItems:** utilizados conjuntamente para mostrar marcas y textos asociados.

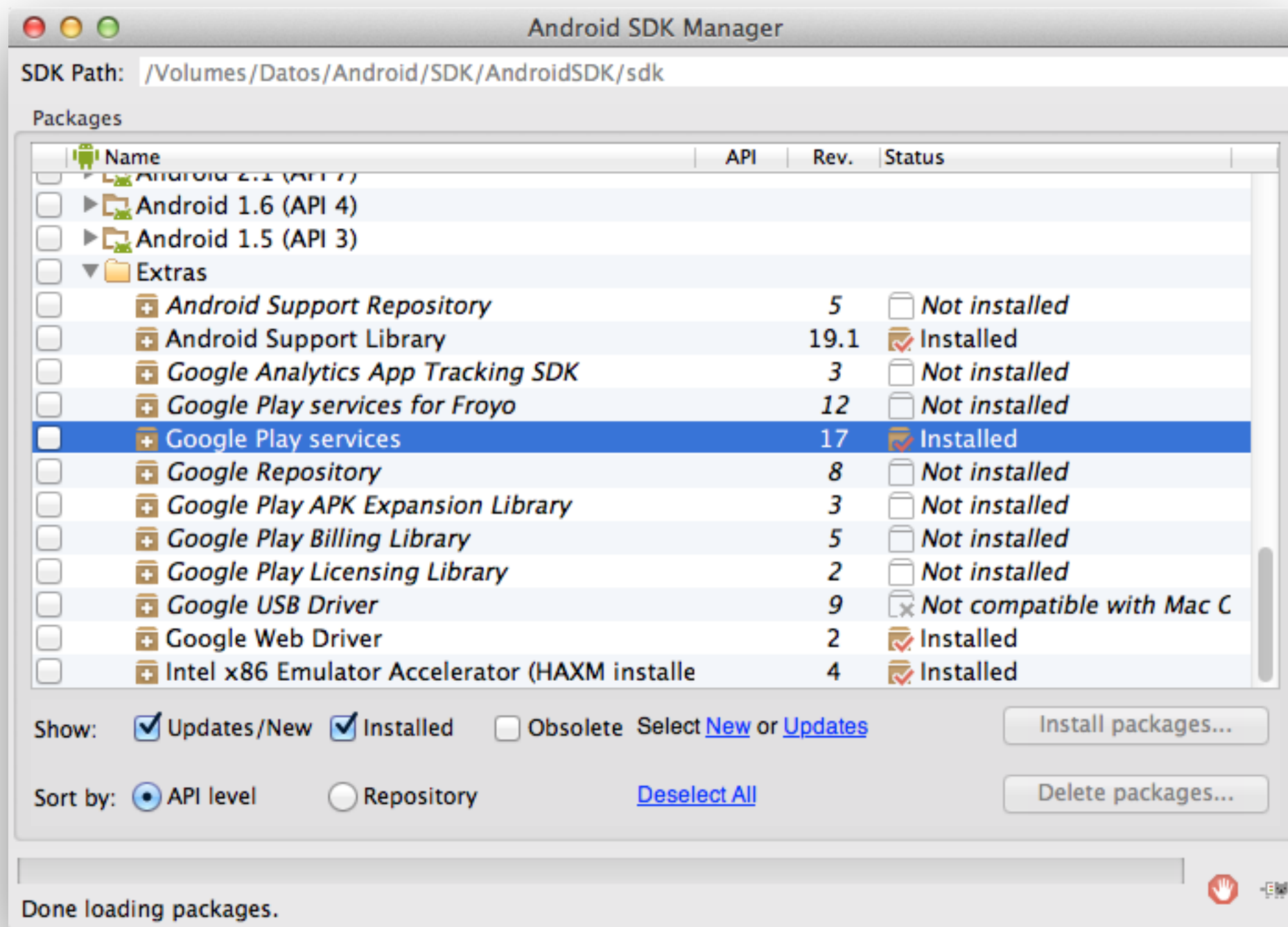
# Google Maps API Key



No es posible utilizar el servicio de mapas sin disponer de la API Key

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

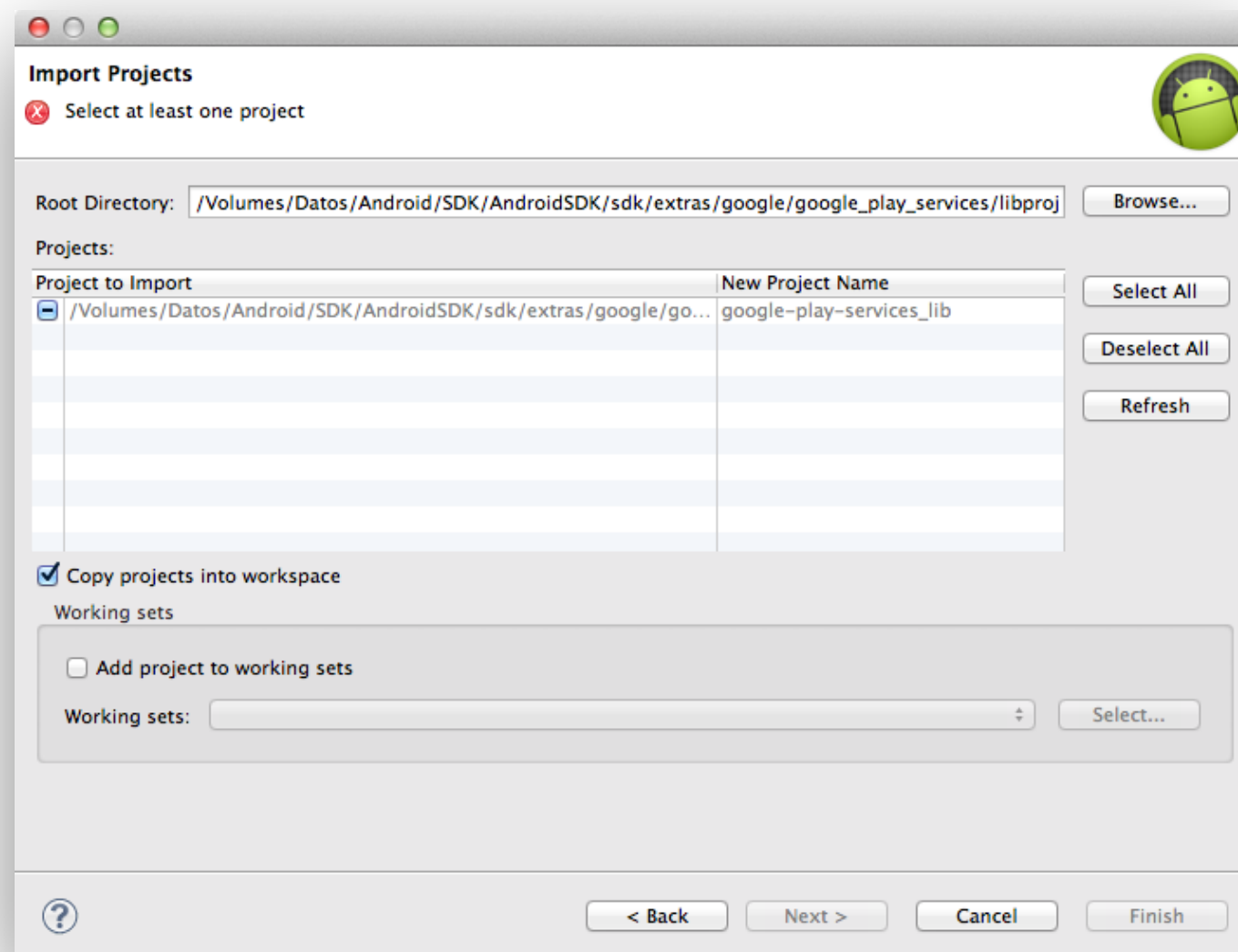
# Instalar Google Play Services



# Instalar Google Play Services

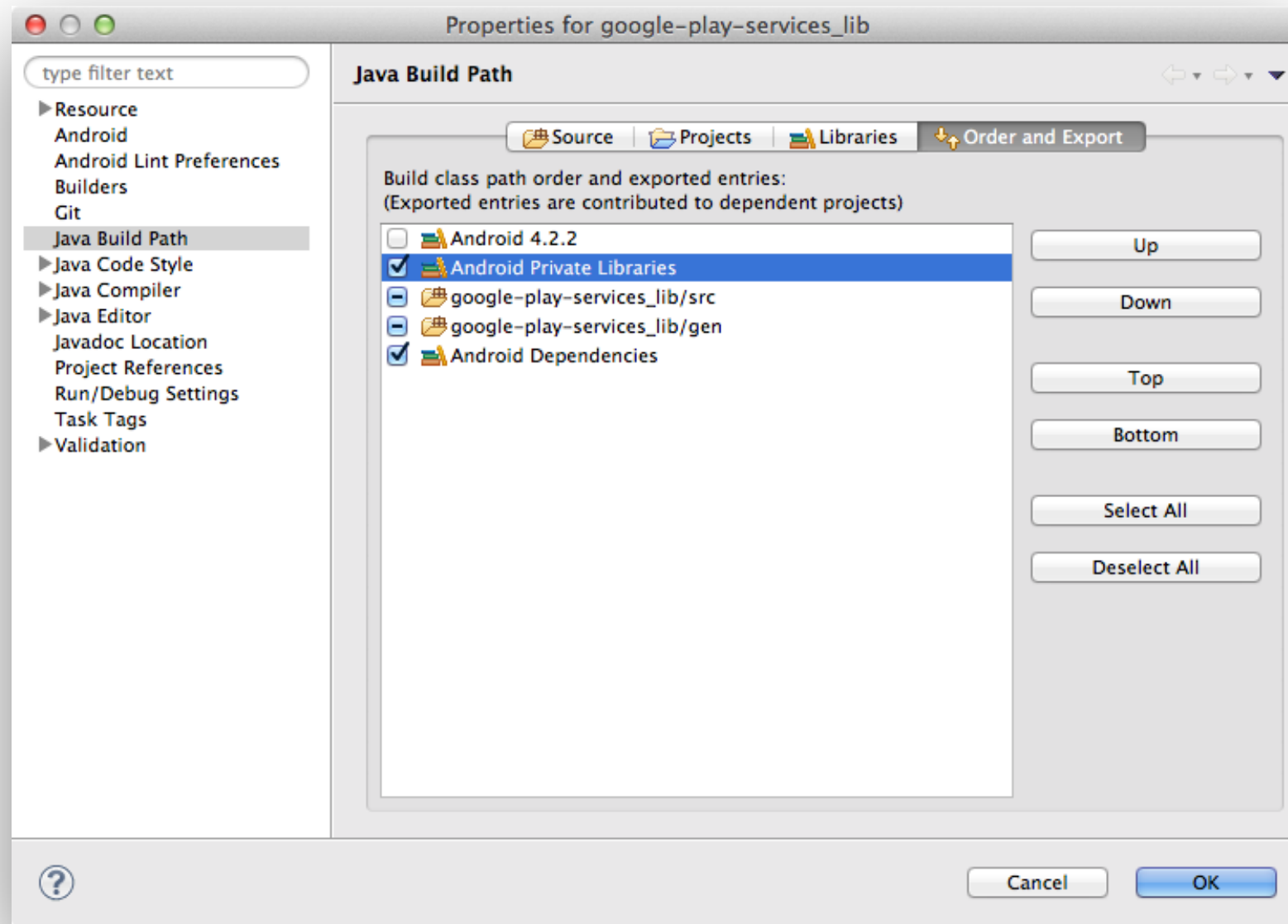
## Importar el proyecto al Workspace

<ruta-sdk>\extras\google\google\_play\_services\libproject\google-play-services\_lib



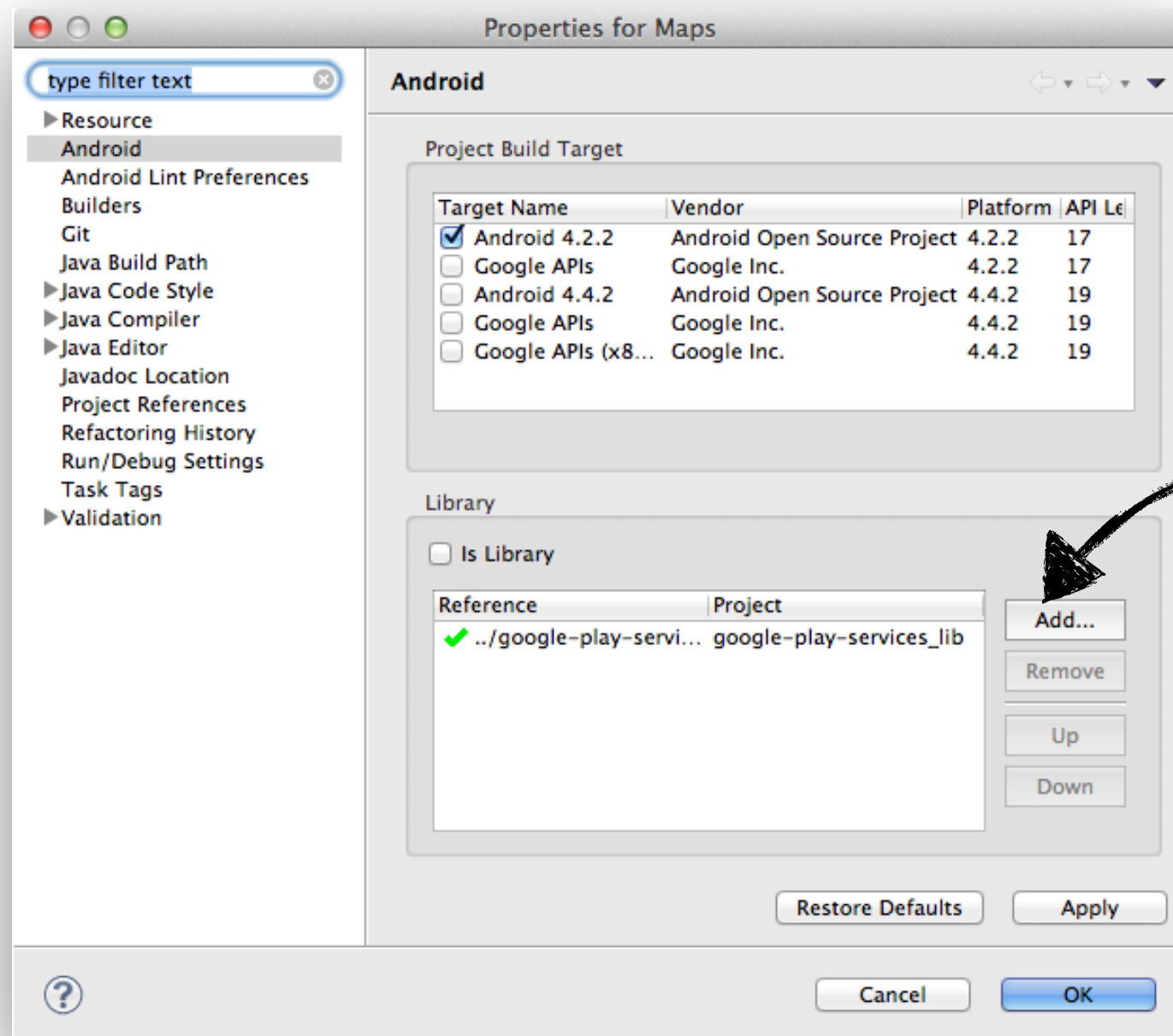
# Instalar Google Play Services

Comprobar que las “Android Private Libraries” se encuentran en el Build Path



# Instalar Google Play Services

En nuestros proyectos, añadir la referencia a la librería de  
"Google Play Services"



Añadir la  
librería

# Instalar Google Play Services

Finalmente, en el manifiesto del proyecto, añadir la siguiente cláusula `<meta-data>` dentro del elemento `<application>`

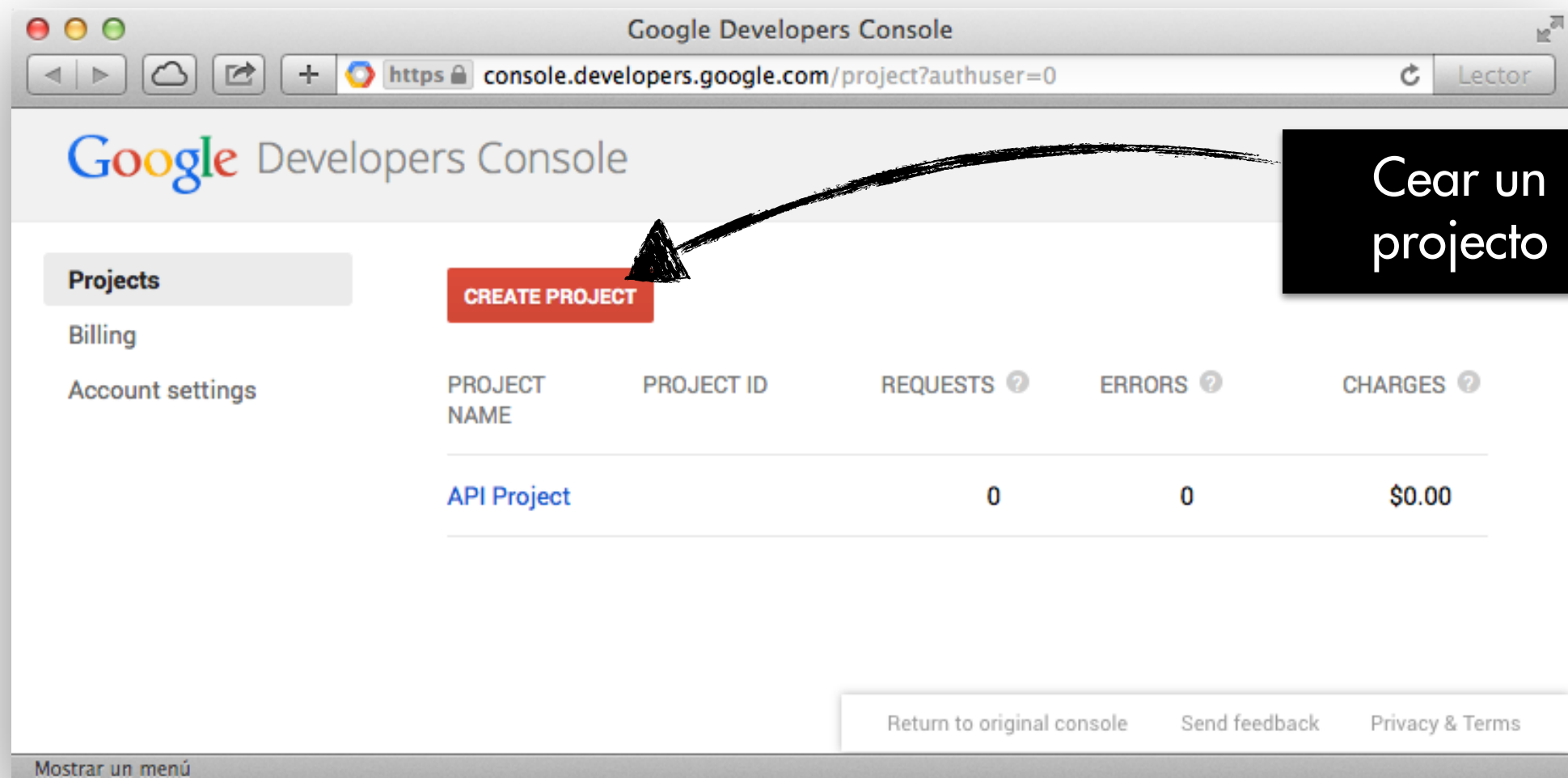
```
<meta-data android:name="com.google.android.gms.version"  
            android:value="@integer/google_play_services_version" />
```



# Obtener una Google Maps API Key

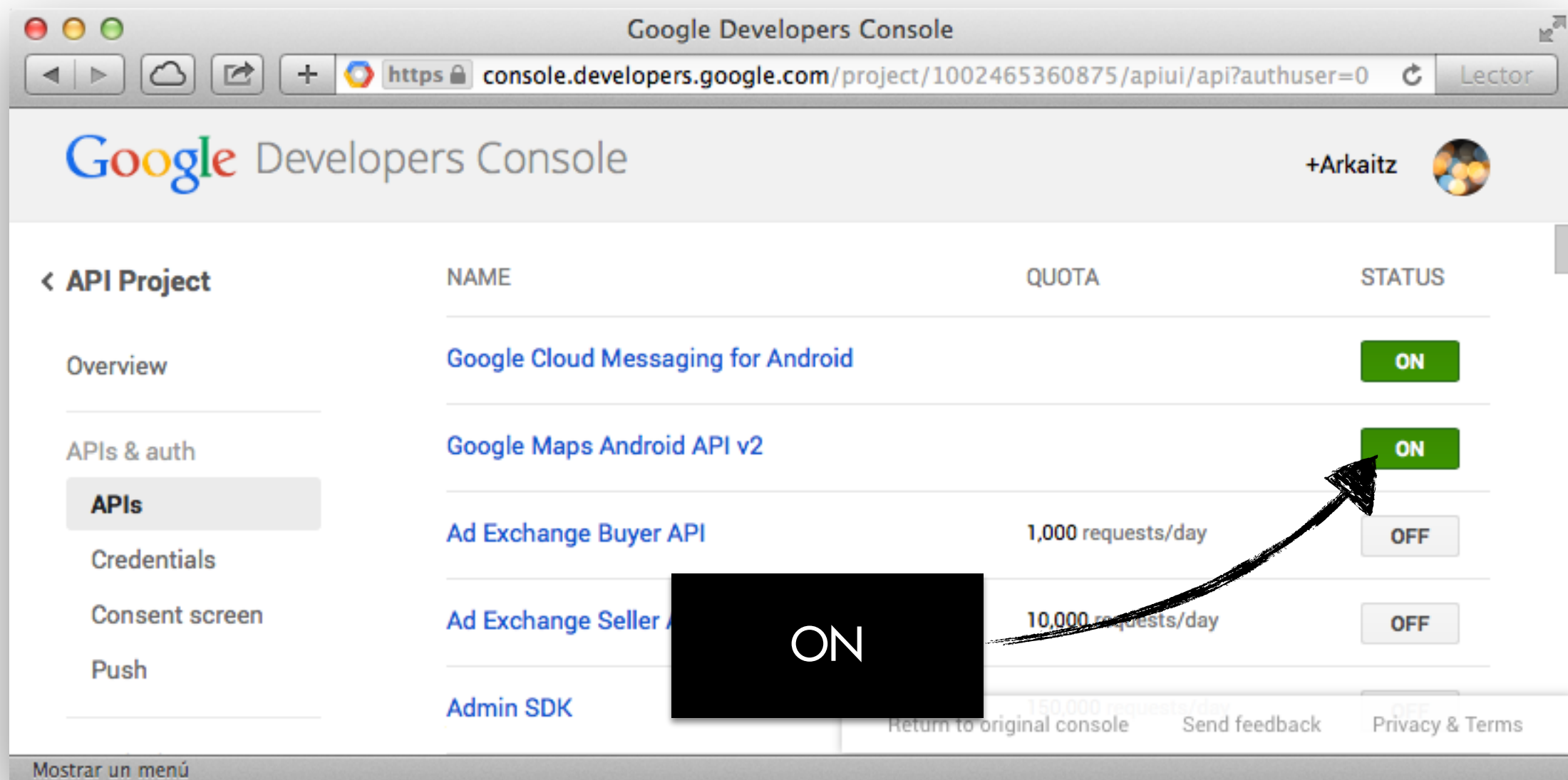
## Acceder a la consola de APIs de Google

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



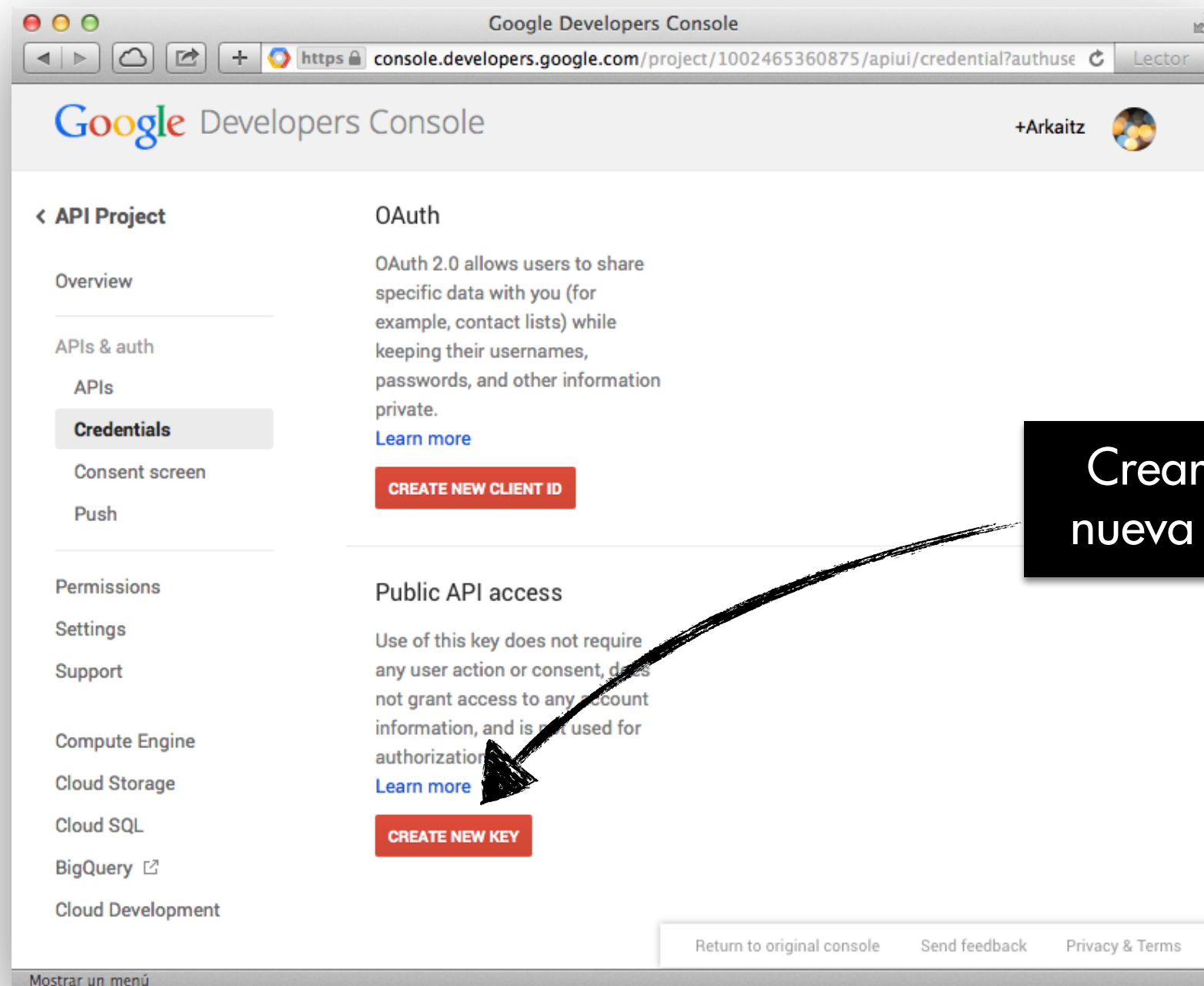
# Obtener una Google Maps API Key

## Activar Google Maps Android API v2



# Obtener una Google Maps API Key

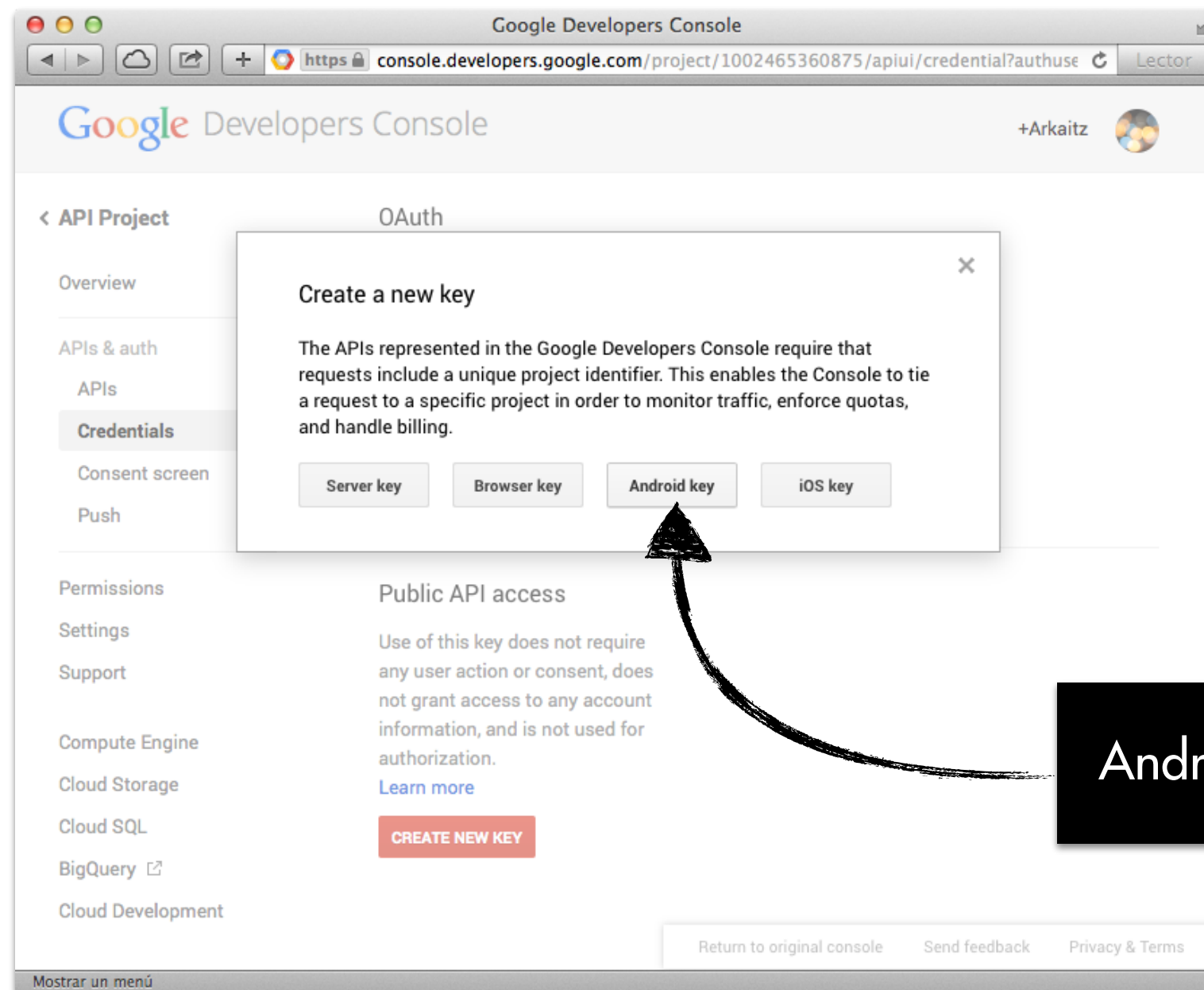
## Crear un nuevo API Key



Crear una  
nueva clave

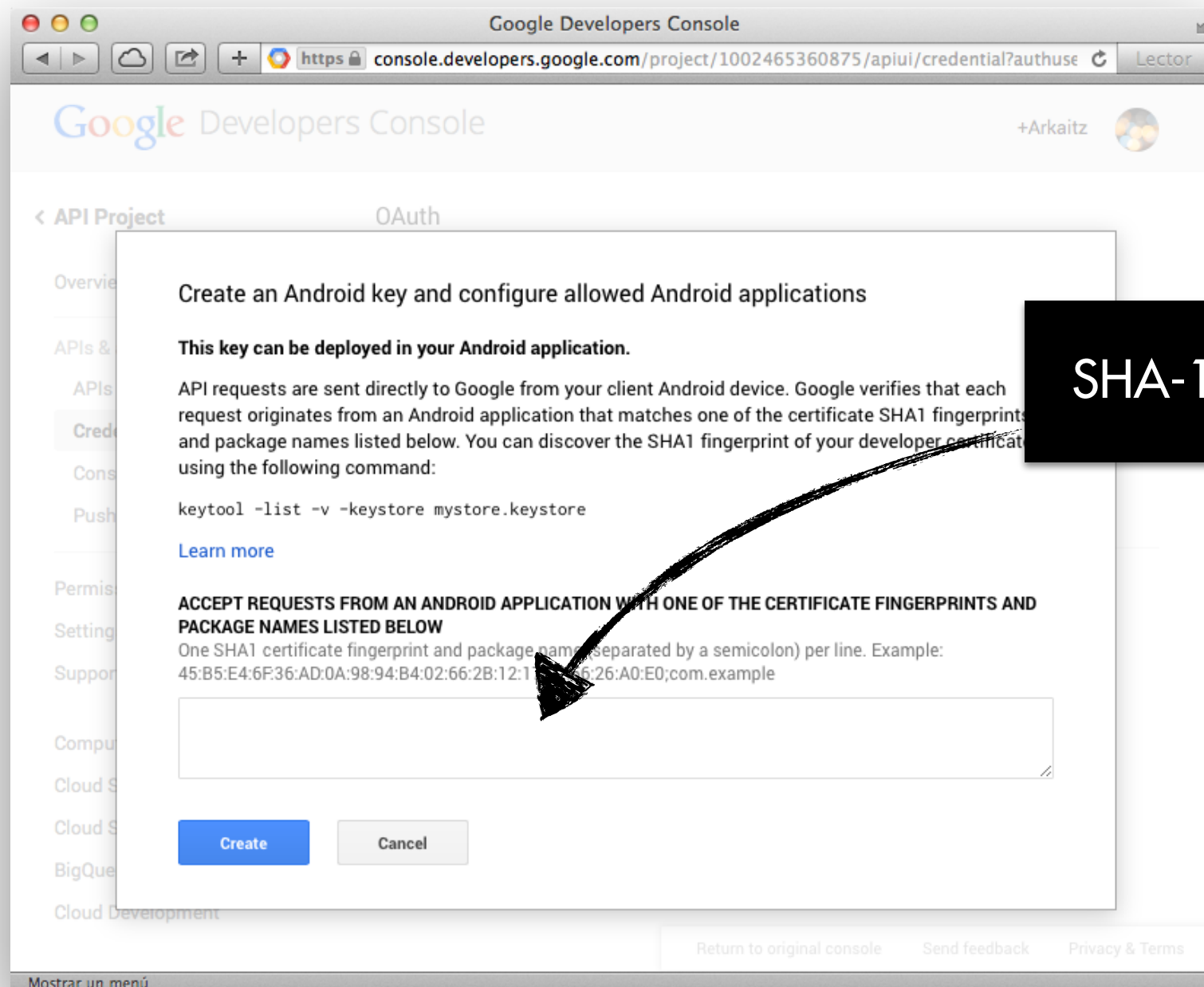
# Obtener una Google Maps API Key

## Crear un nuevo API Key



# Obtener una Google Maps API Key

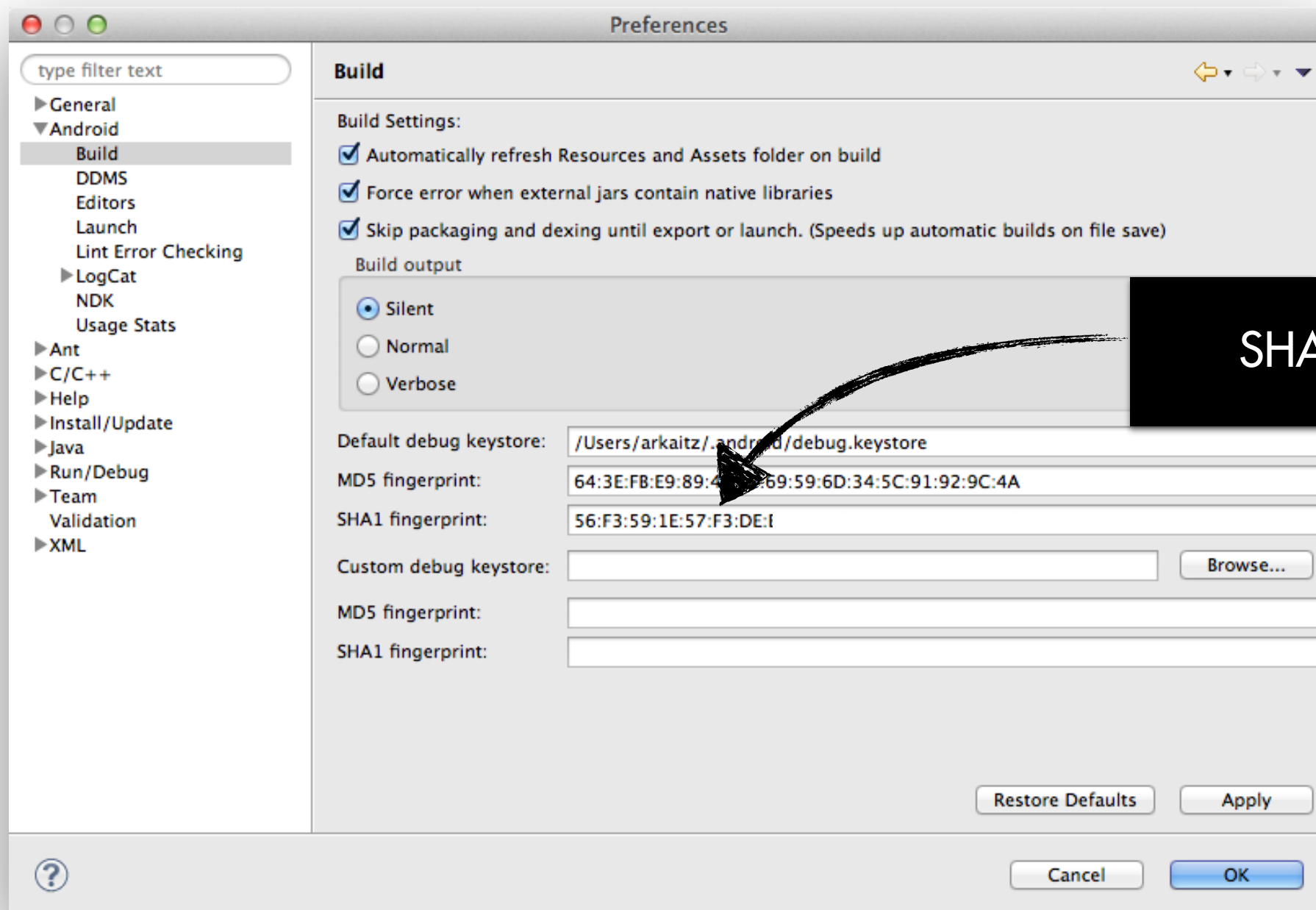
## Crear un nuevo API Key



SHA-1 ;package.name

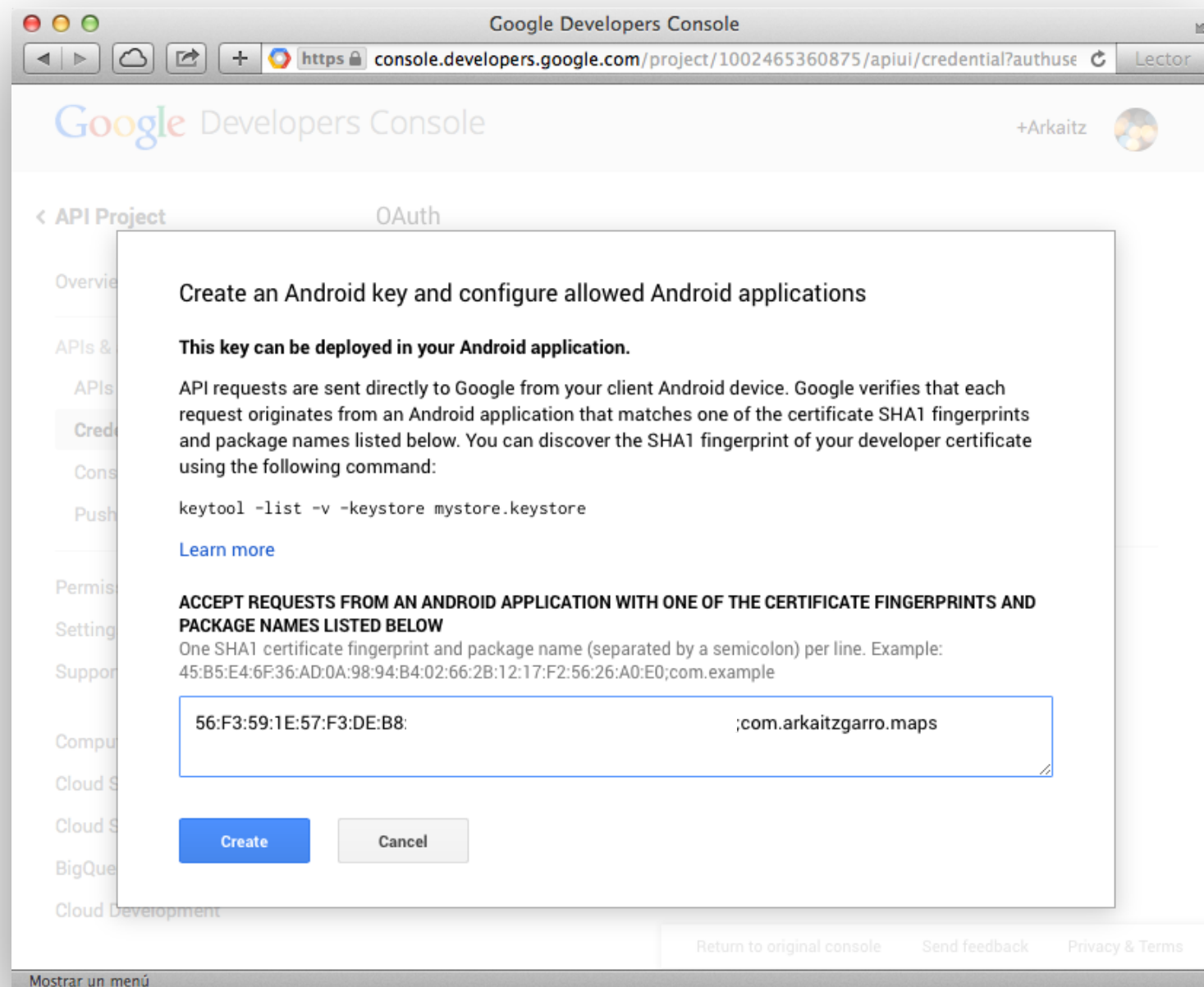
# Obtener una Google Maps API Key

## Obtener la marca SHA-1 de Eclipse



# Obtener una Google Maps API Key

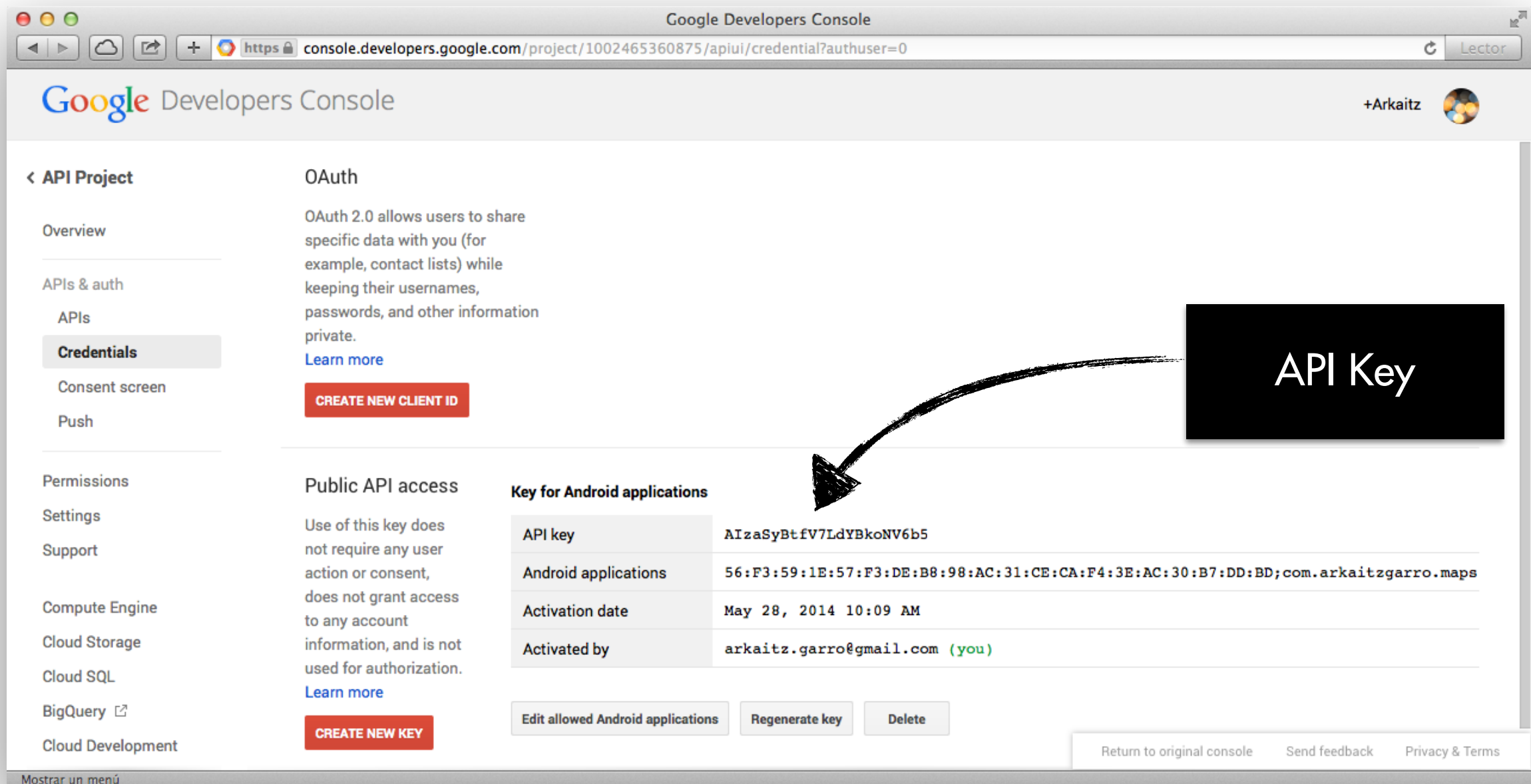
## Obtener la marca SHA-1 de Eclipse





# Obtener una Google Maps API Key

## Obtener el API Key generado



The screenshot shows the Google Developers Console interface. On the left, the 'API Project' sidebar is visible with 'Credentials' selected. The main content area shows the 'OAuth' section with a 'CREATE NEW CLIENT ID' button. Below this, the 'Public API access' section is visible. To the right, the 'Key for Android applications' table displays the generated API key. A hand-drawn arrow points from a black box labeled 'API Key' to the 'API key' field in the table.

Key for Android applications	
API key	AIzaSyBtfV7LdYBkoNV6b5
Android applications	56:F3:59:1E:57:F3:DE:B8:98:AC:31:CE:CA:F4:3E:AC:30:B7:DD:BD;com.arkaitzgarro.maps
Activation date	May 28, 2014 10:09 AM
Activated by	arkaitz.garro@gmail.com (you)

Buttons at the bottom of the table: Edit allowed Android applications, Regenerate key, Delete.

Footer links: Return to original console, Send feedback, Privacy & Terms.



# Obtener una Google Maps API Key

Finalmente, en el manifiesto del proyecto, añadir la siguiente cláusula `<meta-data>` dentro del elemento `<application>`

```
<meta-data android:name="com.google.android.maps.v2.API_KEY"
          android:value="api_key" />
```

Añadir los siguientes permisos.

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

# Mostrar un mapa

Para mostrar una vista de mapa, creamos una nueva actividad cuya vista es un fragmento de tipo *MapFragment*.

```
public class MyMapActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.map_layout);  
    }  
}
```

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

# Configurar el mapa

La vista de mapa puede ser personalizada y consultada:

```
// Cambiar el tipo de visualización
map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
map.setMapType(MAP_TYPE_HYBRID);
map.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
map.setMapType(GoogleMap.MAP_TYPE_TERRAIN);

int type = map.getMapType();
```

# Configurar el mapa

## Añadir marcadores

```
MarkerOptions marker = new MarkerOptions()  
    .position(new LatLng(0, 0)).title("Marker");  
  
map.addMarker(marker);
```

# Configurar el mapa

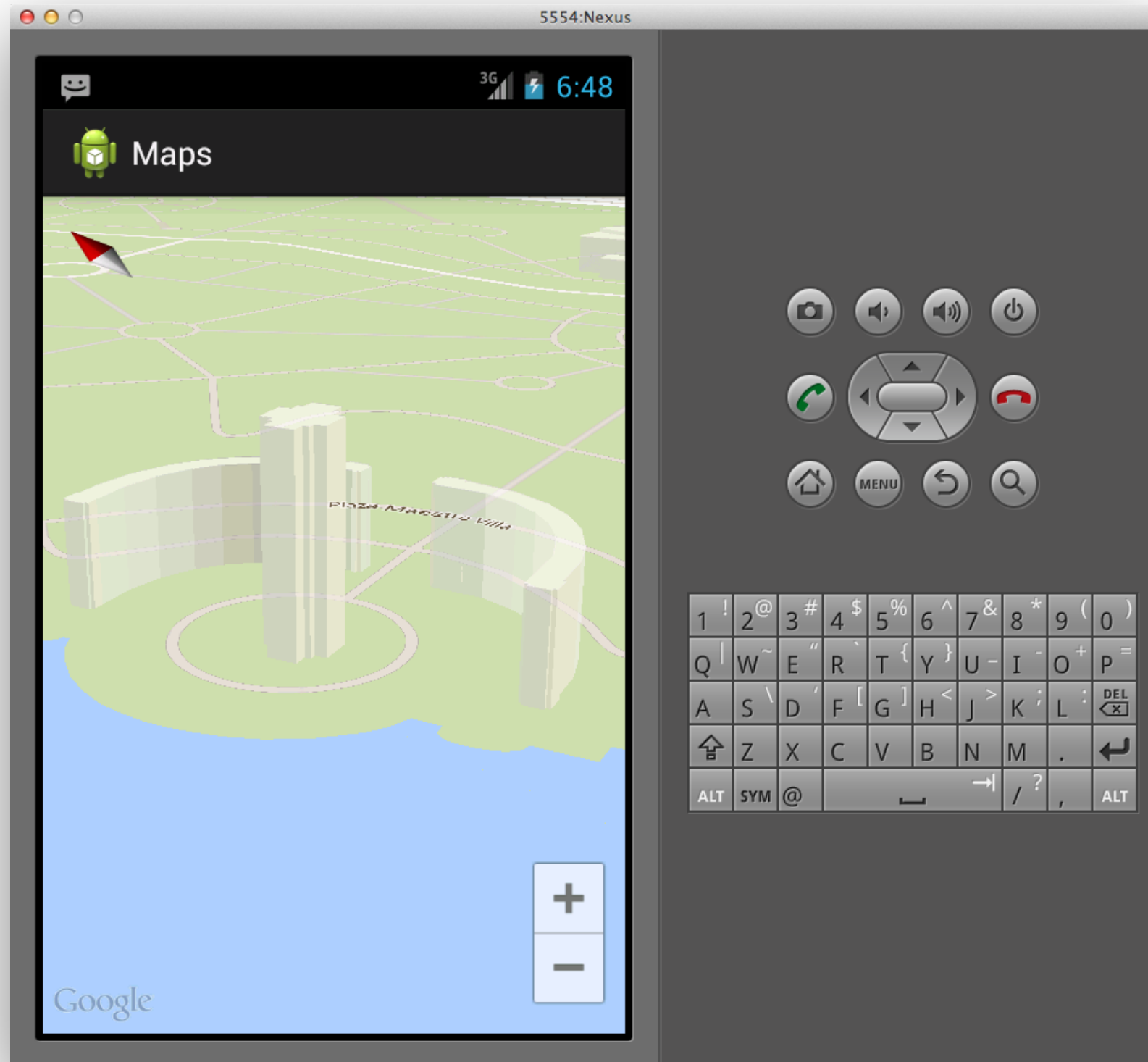
## Configurar la vista

```
LatLng madrid = new LatLng(40.417325, -3.683081);
CameraPosition camPos = new CameraPosition.Builder().target(madrid)
    .zoom(19)
    .bearing(45)
    .tilt(70)
    .build();

CameraUpdate camUpd = CameraUpdateFactory.newCameraPosition(camPos);

mMap.animateCamera(camUpd);
```

# Configurar el mapa



Ejercicio