

# Google Maps

---

## Map Activity



Google Map Activity can be inserted into an app via a Wizard from Studio. API Keys must be acquired from google directly.

# New Set Location Button

strings.xml

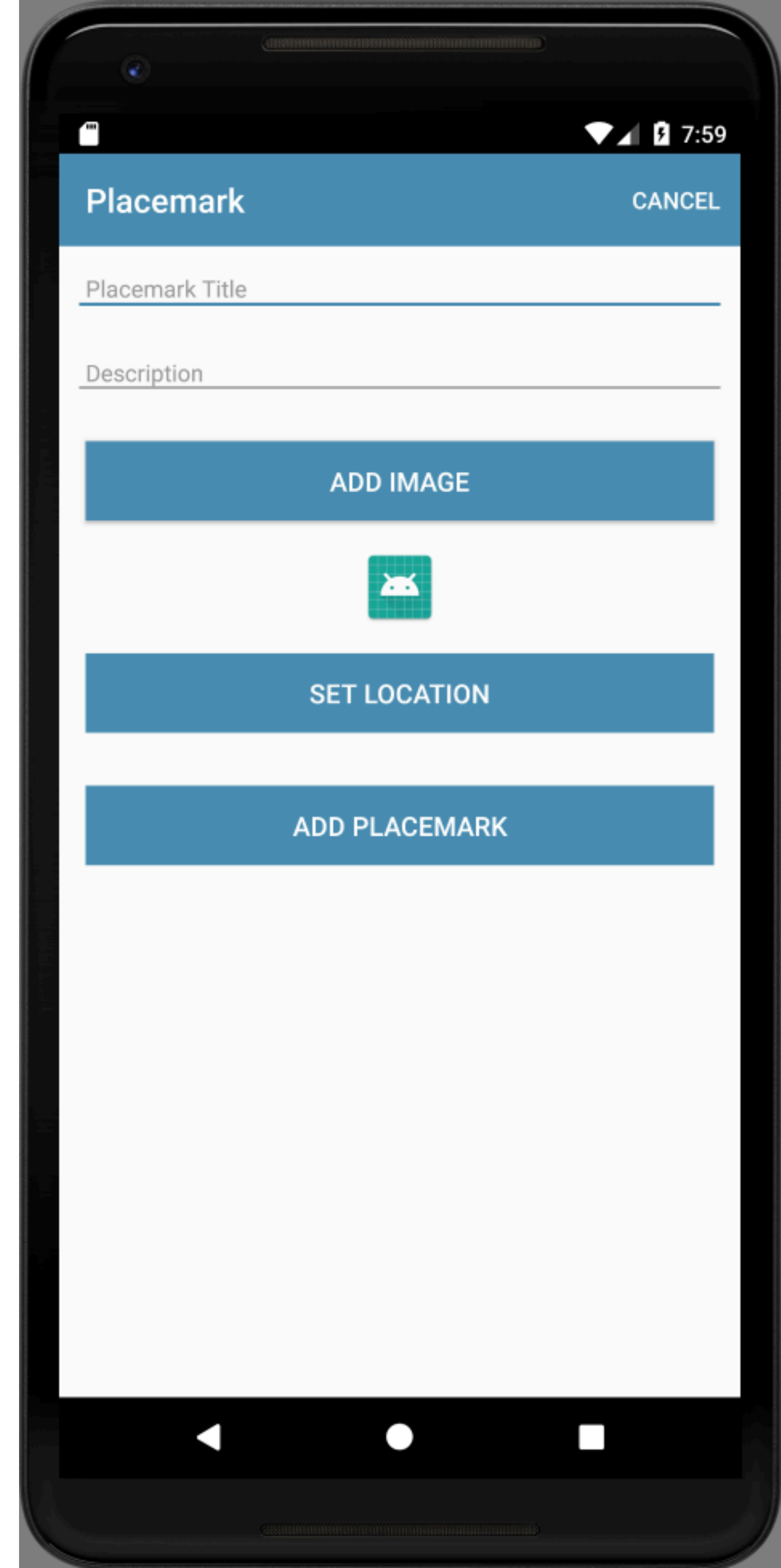
```
<string name="button_location">Set Location</string>
```

activity\_placemark.xml

```
<Button  
    android:id="@+id/placemarkLocation"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="16dp"  
    android:background="@color/colorAccent"  
    android:paddingBottom="8dp"  
    android:paddingTop="8dp"  
    android:stateListAnimator="@null"  
    android:text="@string/button_location"  
    android:textColor="@color/colorPrimary"  
    android:textSize="16sp"/>
```

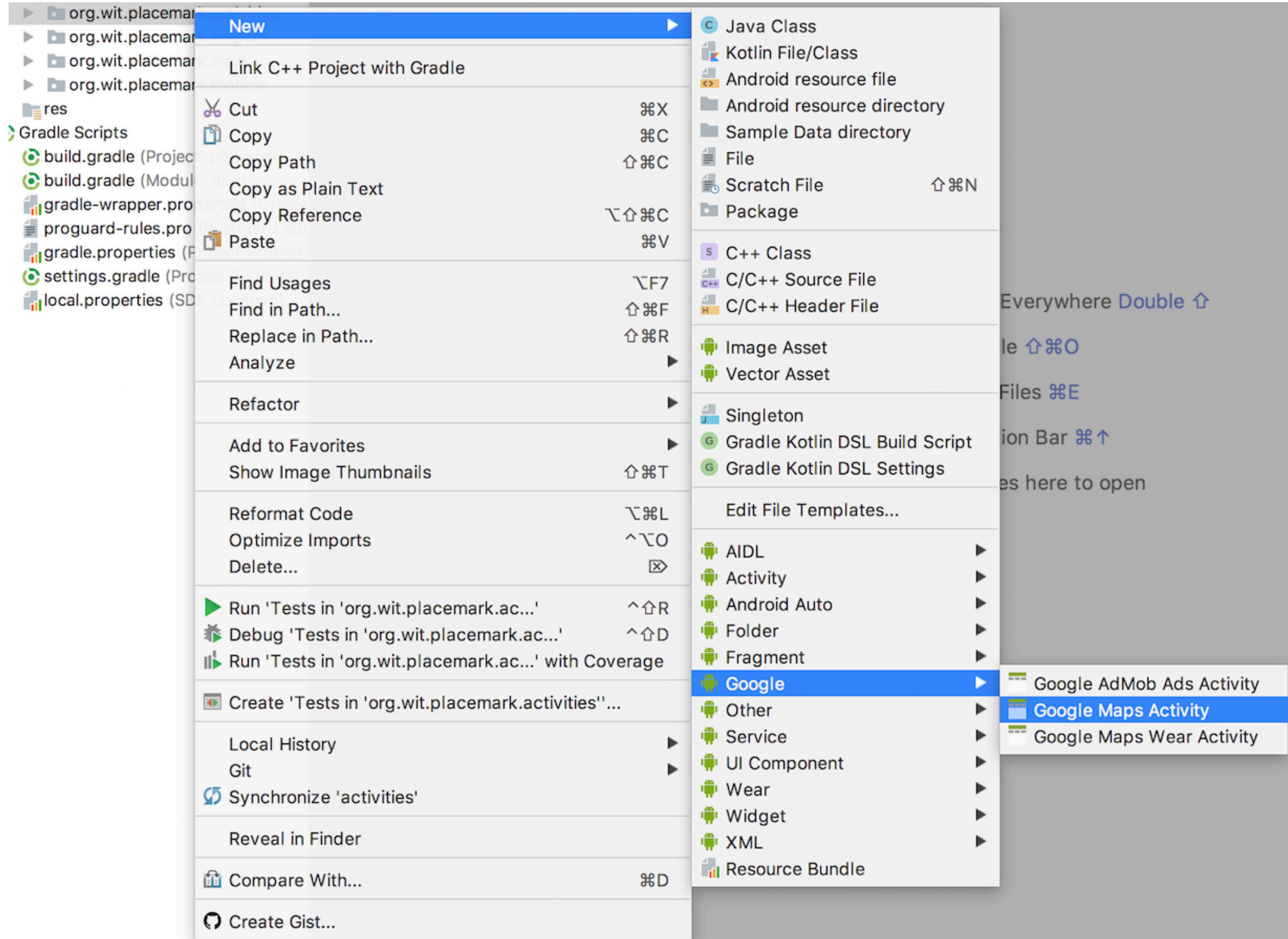
PlacemarkActivity

```
placemarkLocation.setOnClickListener {  
}
```







# Add Google Maps Activity

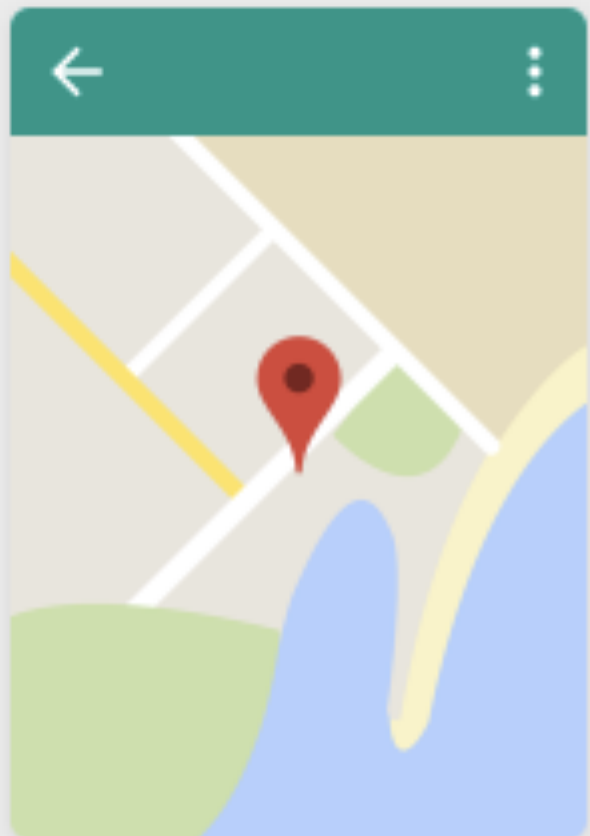


New Android Activity

 **Configure Activity**  
Android Studio



**Creates a new activity with a Google Map**



Activity Name:

MapsActivity

Layout Name:

activity\_maps

Title:

Map

☐ Launcher Activity

Hierarchical Parent:

Package name:

org.wit.placemark.activities

Source Language:

Kotlin

The hierarchical parent activity, used to provide a default implementation for the 'Up' button

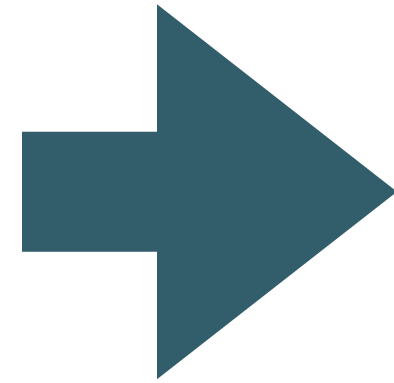
Cancel

Previous

Next

Finish

## Add Google Maps Activity



Generates these updates

build.gradle:

```
implementation 'com.google.android.gms:play-services-maps:15.0.1'
```

strings.xml

```
<string name="title_activity_maps">Map</string>
```





# AndroidManifest.xml

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

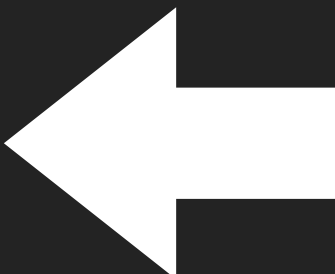
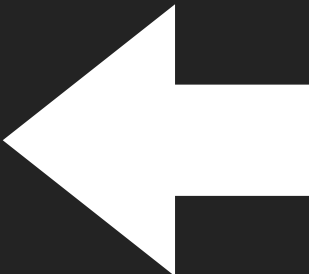
    <!--
        The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
        Google Maps Android API v2, but you must specify either coarse or fine
        location permissions for the 'MyLocation' functionality.
    -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

    <application
        android:name=".main.MainApp"
        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">
        <activity android:name=".activities.PlacemarkActivity">
        </activity>
        <activity android:name=".activities.PlacemarkListActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <!--
            The API key for Google Maps-based APIs is defined as a string resource.
            (See the file "res/values/google_maps_api.xml").
            Note that the API key is linked to the encryption key used to sign the APK.
            You need a different API key for each encryption key, including the release key that is u
            sign the APK for publishing.
            You can define the keys for the debug and release targets in src/debug/ and src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key"/>

        <activity
            android:name=".activities.MapsActivity"
            android:label="@string/title_activity_maps">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="org.wit.placemark.activities.PlacemarkActivity"/>
            </activity>
        </application>

</manifest>
```



# AndroidManifest.xml - Permissions

## Specify App Permissions

---

Apps that use location services must request location permissions. Android offers two location permissions: [ACCESS\\_COARSE\\_LOCATION](#) and [ACCESS\\_FINE\\_LOCATION](#). The permission you choose determines the accuracy of the location returned by the API. If you specify [ACCESS\\_COARSE\\_LOCATION](#), the API returns a location with an accuracy approximately equivalent to a city block.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.android.gms.location.sample.basiclocationsample" >

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

```
<!--
    The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
    Google Maps Android API v2, but you must specify either coarse or fine
    location permissions for the 'MyLocation' functionality.
-->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```



# AndroidManifest.xml - Keys

## Get API Key



To use the Google Maps Android API, you must register your app project on the Google API Console and get a Google API key which you can add to your app.

### Quick guide to getting a key

---

#### Step 1. Get an API key from the Google API Console

Click the button below, which guides you through the process of registering a project in the Google API Console, activates the Google Maps Android API automatically, and generates a generic, unrestricted API key.

GET A KEY

```
</activity>
```

```
<!--
```

*The API key for Google Maps-based APIs is defined as a string resource.*

*(See the file "res/values/google\_maps\_api.xml").*

*Note that the API key is linked to the encryption key used to sign the APK.*

*You need a different API key for each encryption key, including the release key that is used to sign the APK for publishing.*

*You can define the keys for the debug and release targets in src/debug/ and src/release/.*

```
-->
```

```
<meta-data
```

```
  android:name="com.google.android.geo.API_KEY"
```

```
  android:value="@string/google_maps_key"/>
```





Your free trial is waiting: activate now to get \$300 credit to explore Google Cloud products. [Learn more](#)

DISMISS

ACTIVATE



Google Cloud Platform

API Project ▼



API key



REGENERATE KEY



DELETE

This API key can be used in this project and with any API that supports it. To use this key in your application, pass it with the `key=API_KEY` parameter.

Creation date

Sep 26, 2018, 6:12:57 PM

Created by

edeleastar.tssg@gmail.com (you)

API key

[REDACTED]



Name

API key 1

### Key restrictions

Restrictions prevent unauthorized use and quota theft. [Learn more](#)



Application restrictions: None

API restrictions: Maps SDK for Android

[Application restrictions](#)

API restrictions

Application restrictions specify which web sites, IP addresses, or apps can use this key. You can set one restriction type per key.

#### Application restrictions

☒ None

☐ HTTP referrers (web sites)

☐ IP addresses (web servers, cron jobs, etc.)

☐ Android apps

☐ iOS apps

Note: It may take up to 5 minutes for settings to take effect

Save

Cancel

# google maps api.xml

```
<resources>
```

```
<!--
```

*TODO: Before you run your application, you need a Google Maps API key.*

*To get one, follow this link, follow the directions and press "Create" at the*

*[https://console.developers.google.com/flows/enableapi?apiid=maps\\_android\\_backend](https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend)*

*You can also add your credentials to an existing key, using these values:*

*Package name:*

*BC:AA:86:5A:D7:8C:52:EA:1C:F2:24:FB:80:2C:A6:73:1D:B4:DA:8B*

*SHA-1 certificate fingerprint:*

*BC:AA:86:5A:D7:8C:52:EA:1C:F2:24:FB:80:2C:A6:73:1D:B4:DA:8B*

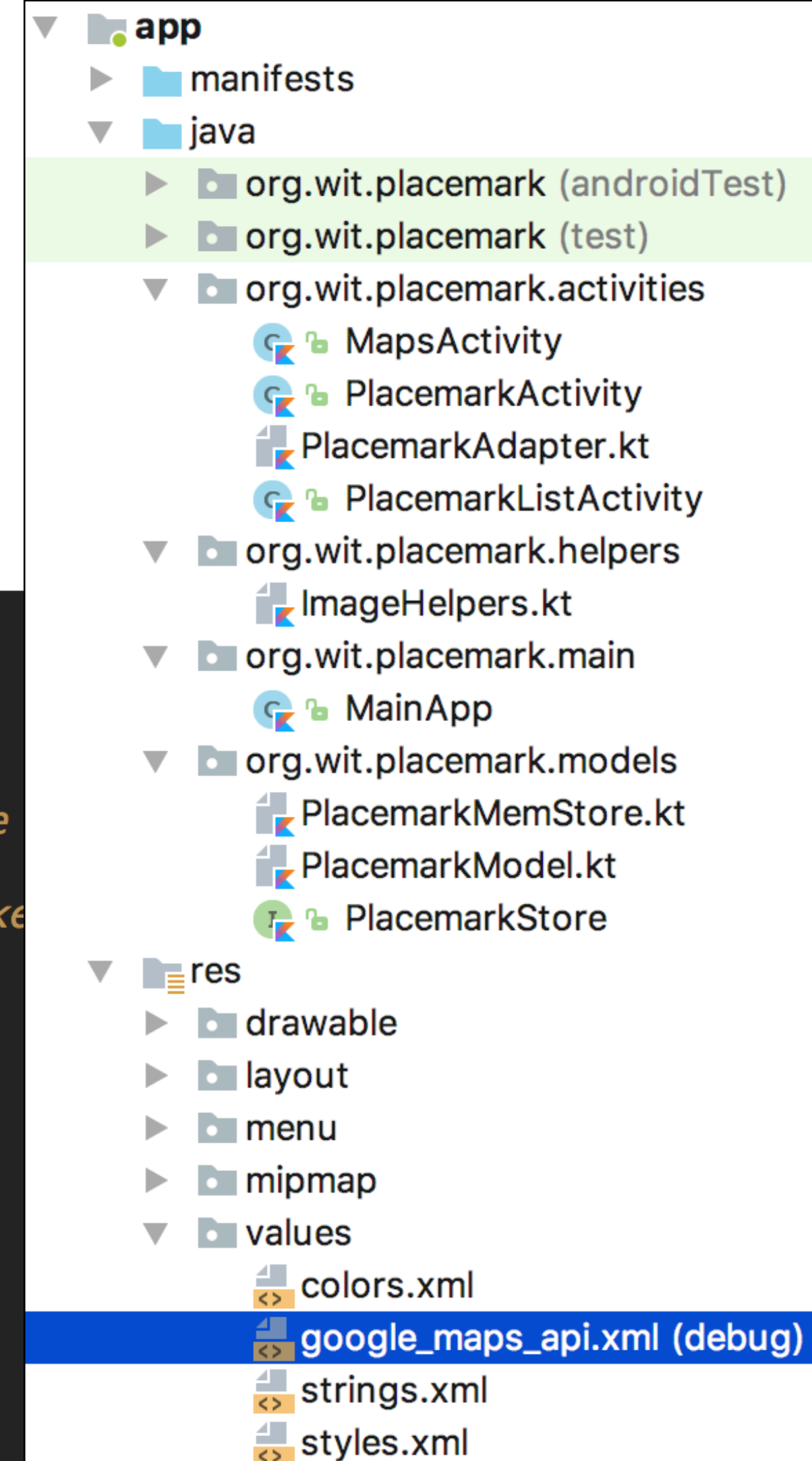
*Alternatively, follow the directions here:*

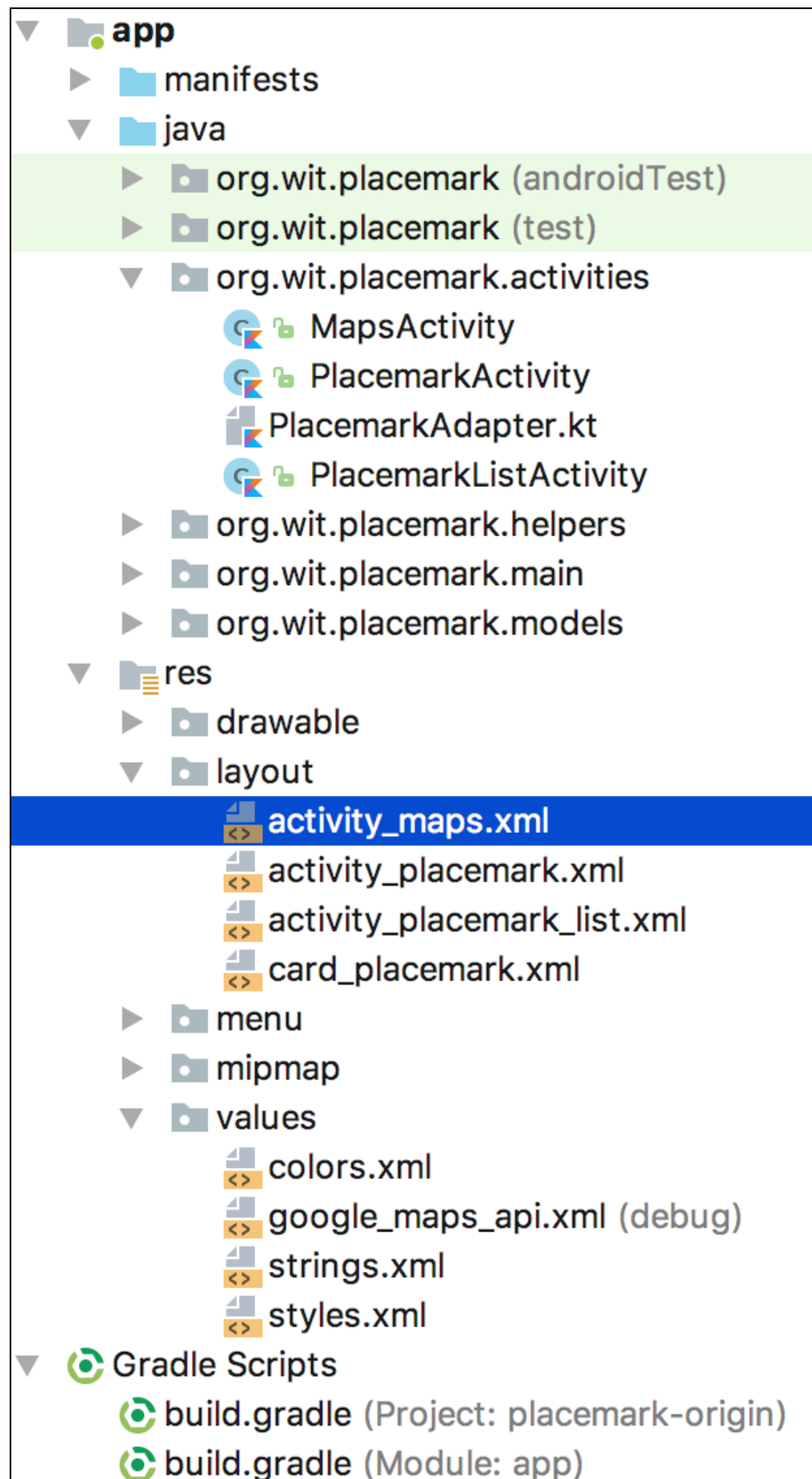
*<https://developers.google.com/maps/documentation/android/start#get-key>*

*Once you have your key (it starts with "AIza"), replace the "google\_maps\_key" string in this file.*

```
-->
```

```
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">YOUR API KEY HERE</string>
</resources>
```





## activity\_maps.xml

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="org.wit.placemark.activities.MapsActivity"/>
```



```
package org.wit.placemark.activities
```

```
import android.support.v7.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  
import org.wit.placemark.R
```

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {
```

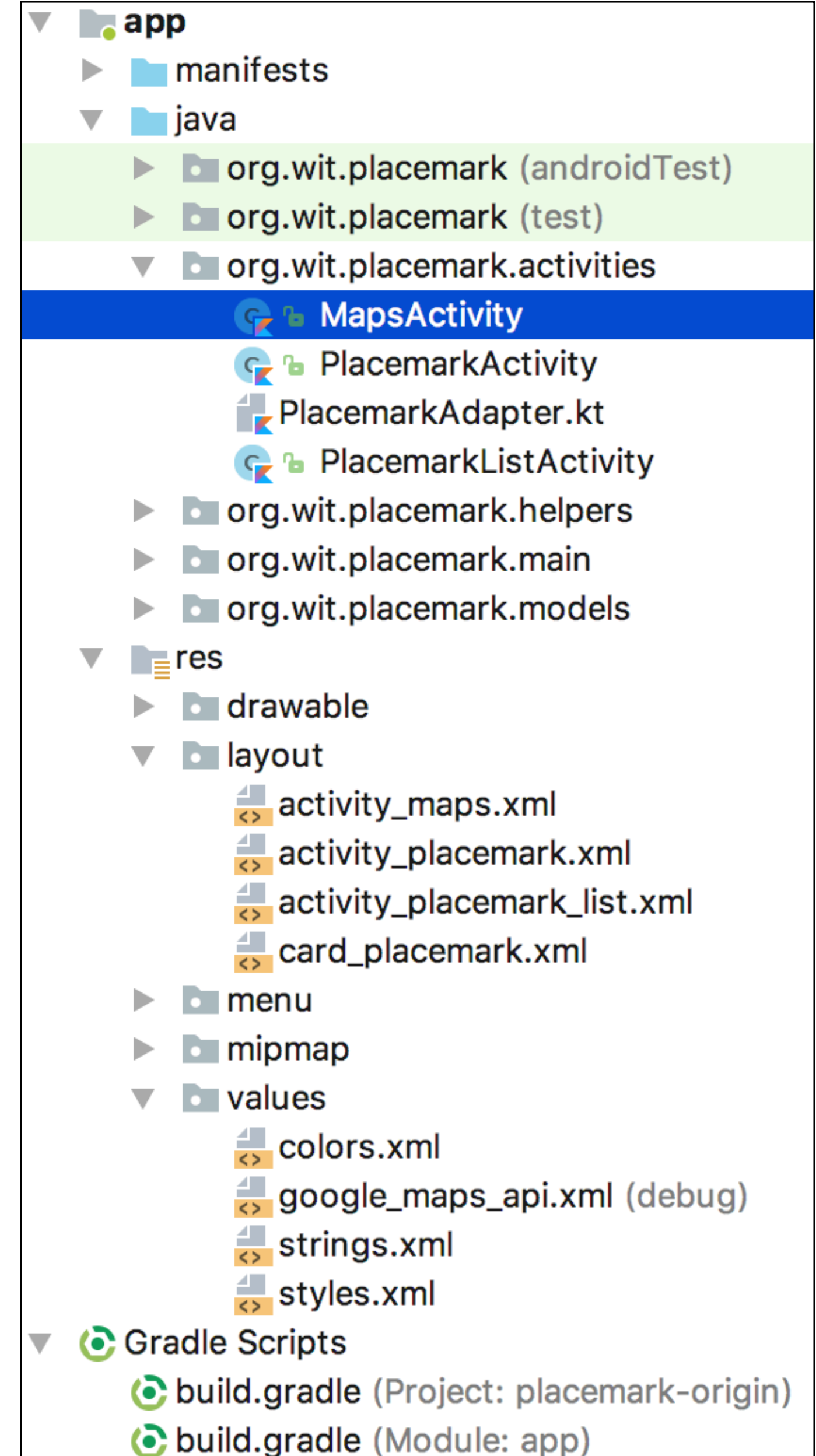
```
    private lateinit var mMap: GoogleMap
```

```
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_maps)  
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.  
        val mapFragment = supportFragmentManager  
            .findFragmentById(R.id.map) as SupportMapFragment  
        mapFragment.getMapAsync(this)  
    }
```

```
    /**  
     * Manipulates the map once available.  
     * This callback is triggered when the map is ready to be used.  
     * This is where we can add markers or lines, add listeners or move the camera. In this case,  
     * we just add a marker near Sydney, Australia.  
     * If Google Play services is not installed on the device, the user will be prompted to install  
     * it inside the SupportMapFragment. This method will only be triggered once the user has  
     * installed Google Play services and returned to the app.  
     */
```

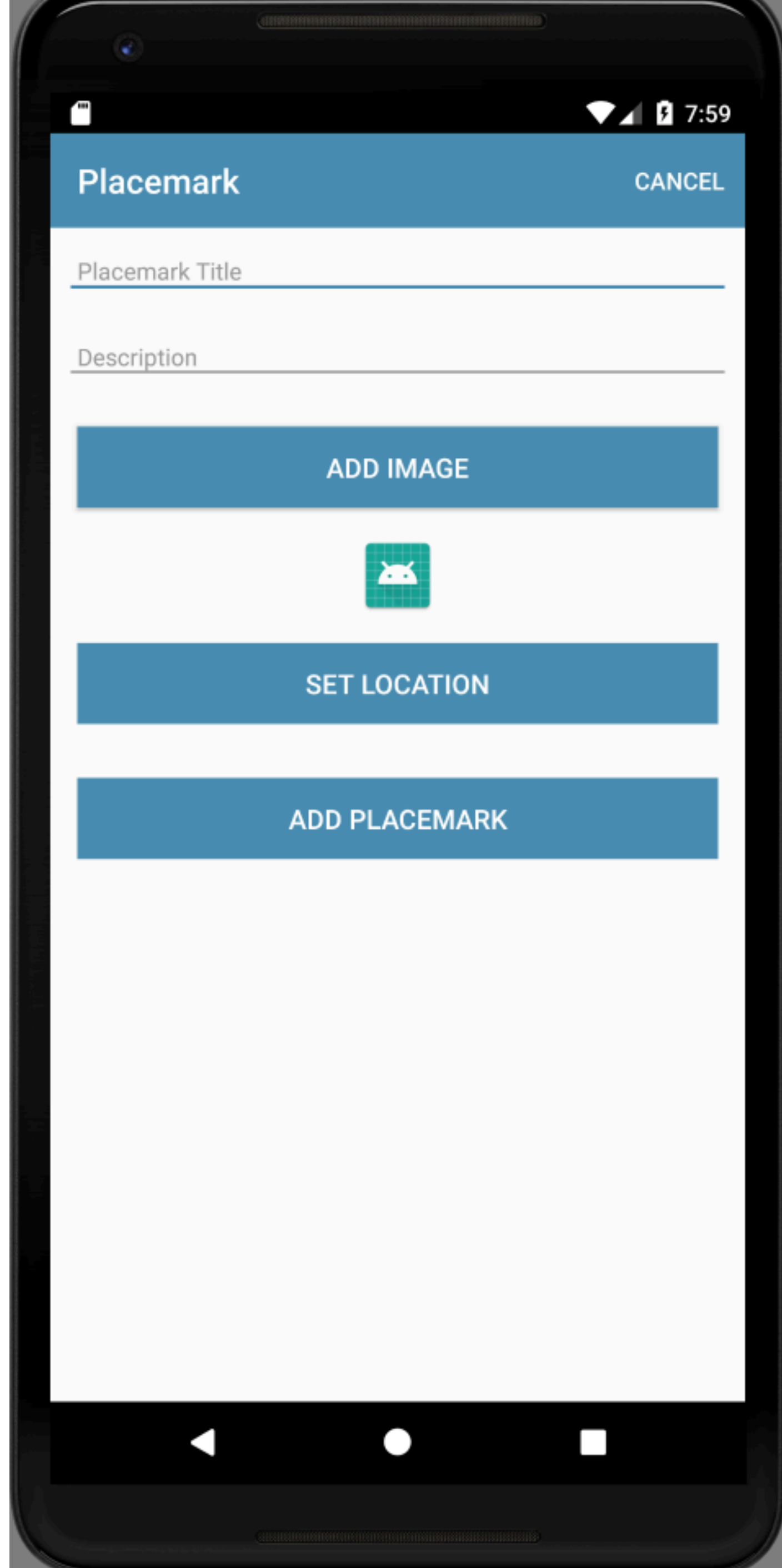
```
    override fun onMapReady(googleMap: GoogleMap) {  
        mMap = googleMap  
  
        // Add a marker in Sydney and move the camera  
        val sydney = LatLng(-34.0, 151.0)  
        mMap.addMarker(MarkerOptions().position(sydney).title("Marker in Sydney"))  
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))  
    }  
}
```

# MapsActivity



The image shows the project structure of an Android application in Android Studio. The project is named 'app'. The structure is as follows:

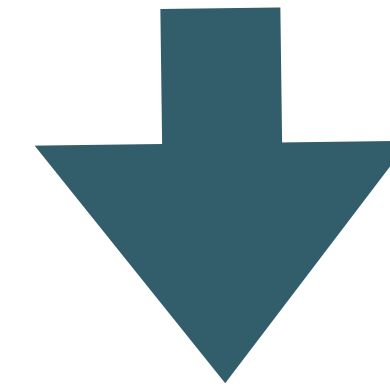
- app
  - manifests
  - java
    - org.wit.placemark (androidTest)
    - org.wit.placemark (test)
    - org.wit.placemark.activities
      - MapsActivity** (selected)
      - PlacemarkActivity
      - PlacemarkAdapter.kt
      - PlacemarkListActivity
      - org.wit.placemark.helpers
      - org.wit.placemark.main
      - org.wit.placemark.models
  - res
    - drawable
    - layout
      - activity\_maps.xml
      - activity\_placemark.xml
      - activity\_placemark\_list.xml
      - card\_placemark.xml
    - menu
    - mipmap
    - values
      - colors.xml
      - google\_maps\_api.xml (debug)
      - strings.xml
      - styles.xml
- Gradle Scripts
  - build.gradle (Project: placemark-origin)
  - build.gradle (Module: app)



## PlacemarkActivity

---

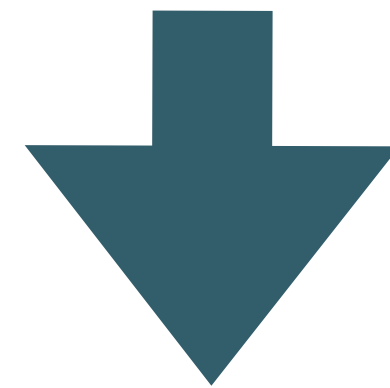
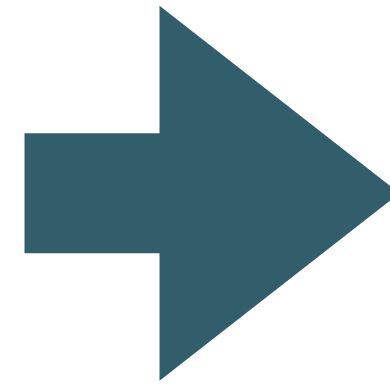
```
placemarkLocation.setOnClickListener {  
}
```



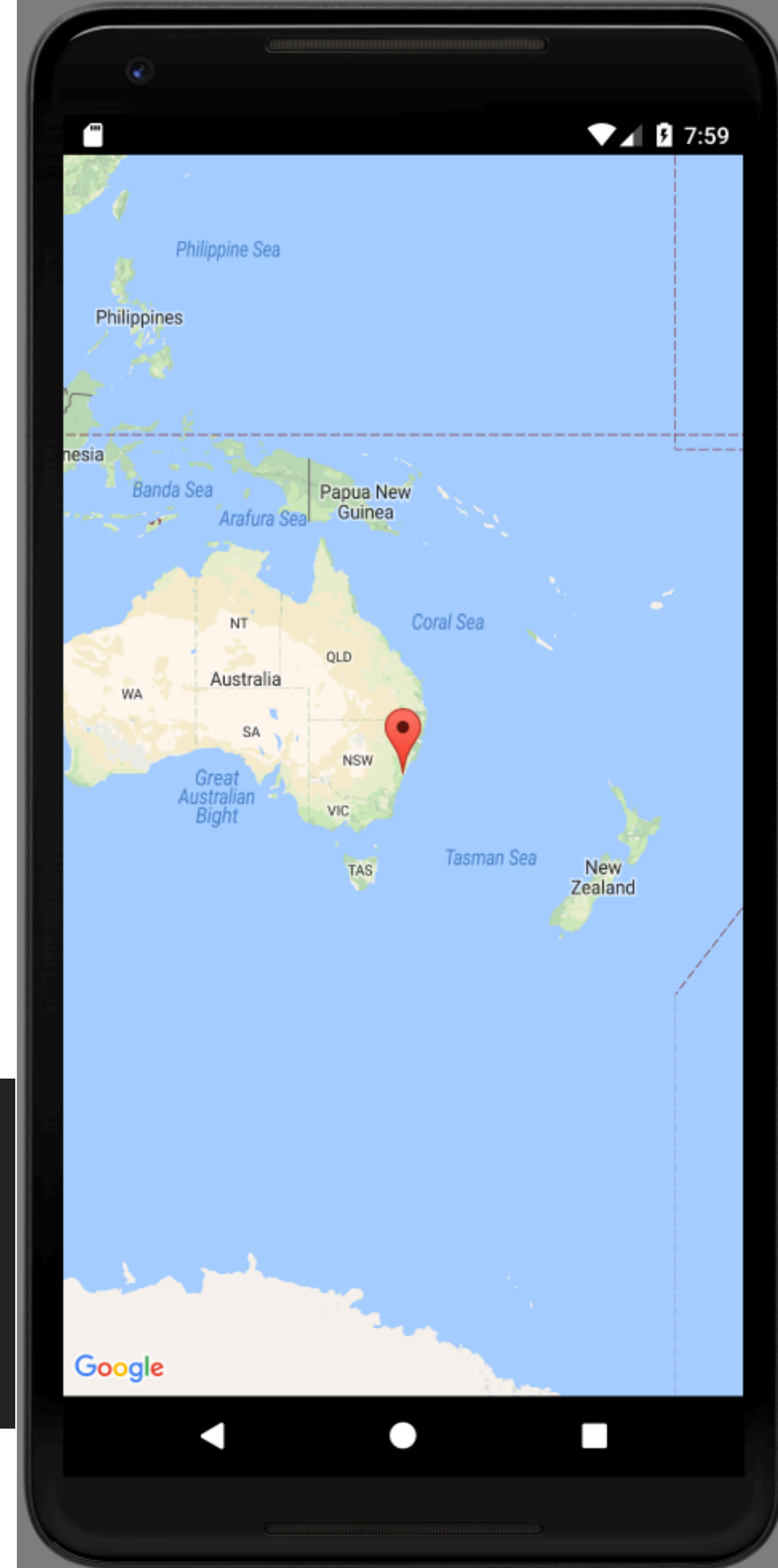
```
placemarkLocation.setOnClickListener {  
    startActivity (intentFor<MapsActivity>())  
}
```



```
placemarkLocation.setOnClickListener {  
    startActivity (intentFor<MapsActivity>())  
}
```



```
override fun onMapReady(googleMap: GoogleMap) {  
    mMap = googleMap  
  
    // Add a marker in Sydney and move the camera  
    val sydney = LatLng(-34.0, 151.0)  
    mMap.addMarker(MarkerOptions().position(sydney).title("Marker in Sydney"))  
    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))  
}
```





## Review AndroidManifest.xml

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

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

    <application
        android:name=".main.MainApp"
        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">
        <activity android:name=".activities.PlacemarkActivity">
        </activity>
        <activity android:name=".activities.PlacemarkListActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

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

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key"/>
        <activity
            android:name=".activities.MapsActivity"
            android:label="@string/title_activity_maps">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="org.wit.placemark.activities.PlacemarkActivity"/>
            </activity>
        </application>

</manifest>
```

## Review MapsActivity

```
package org.wit.placemark.activities

import android.support.v7.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
import org.wit.placemark.R

class MapsActivity : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var mMap: GoogleMap

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        val mapFragment = supportFragmentManager
            .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
    }

    override fun onMapReady(googleMap: GoogleMap) {
        mMap = googleMap
        val sydney = LatLng(-34.0, 151.0)
        mMap.addMarker(MarkerOptions().position(sydney).title("Marker in Sydney"))
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))
    }
}
```



# Change Location + Zoom Level

```
override fun onMapReady(googleMap: GoogleMap) {  
    mMap = googleMap  
    val sydney = LatLng(-34.0, 151.0)  
    mMap.addMarker(MarkerOptions().position(sydney).title("Marker in Sydney"))  
    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))  
}
```



```
override fun onMapReady(googleMap: GoogleMap) {  
    mMap = googleMap  
    val wit = LatLng(52.245696, -7.139102)  
    mMap.addMarker(MarkerOptions().position(wit).title("Marker in Waterford"))  
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(wit, 16f))  
}
```

