Mobile Applications

6. Maps and location-based services for Android

Boni García

boni.garcia@uc3m.es

Telematic Engineering Department
School of Engineering

2024/2025

uc3m | Universidad Carlos III de Madrid

Table of contents

1. Introduction

- In this unit, we study two kind of features for Android apps:

1. Location-based services
   - Emulated location
   - Location providers
   - Geocoding and reverse geocoding
2. Google Maps Platform
   - Google Maps
   - Google Places
   - Google Directions

Table of contents

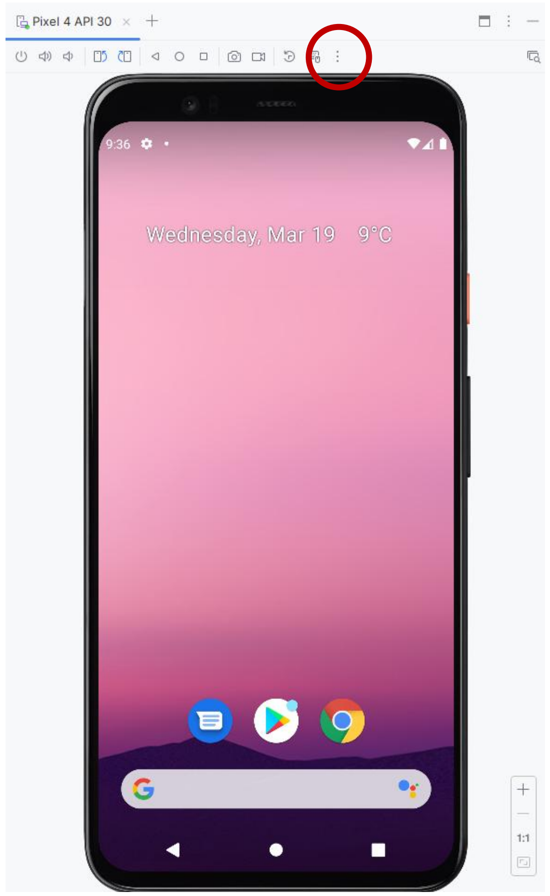
2. Location-based services

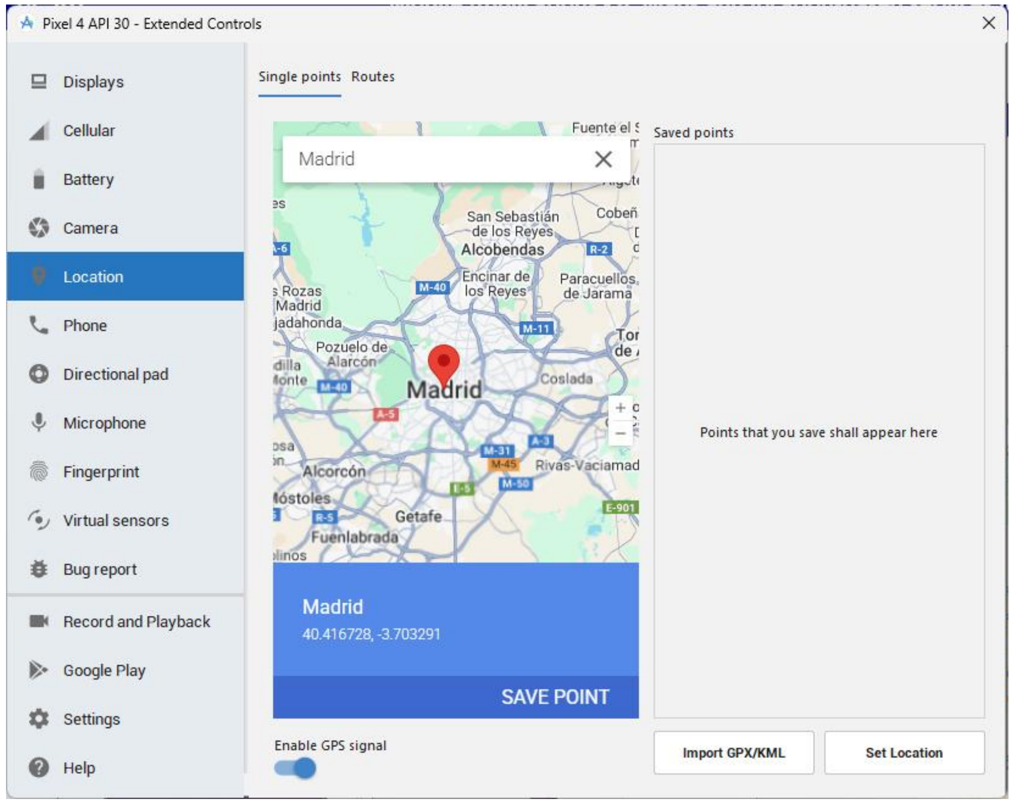
- A location-based service (LBS) is a software service for mobile apps that requires the knowledge about where the device is geographically located
- We study:
  - How to emulate the device location using and emulator in Android Studio
  - The permissions required to manage the device location
  - The most common types of location providers in Android
  - How to implement a location listener (i.e., an app that tracks the location changes)
  - How to implement geocoding in Android (i.e., translate an address to its coordinates and vice versa)

https://developer.android.com/training/location

2. Location-based services - Emulated location

- Android Studio allows to change the location of the device, and even simulate routes





2. Location-based services - Location permissions

- To protect user privacy, apps that use location services must request location **permissions**
- There are different types of permissions regarding the location accuracy:
  - Precise: access to the device's GPS coordinates
    - Pro: It provides accurate real-time location
    - Cons: It consumes more battery

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

  - Approximate: location data based on less accurate sources like the network address (wifi or cellular)
    - Pro: It is more battery friendly
    - Cons: It provides an approximate location

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

As usual, we declare these permission in the manifest file

2. Location-based services - Location providers

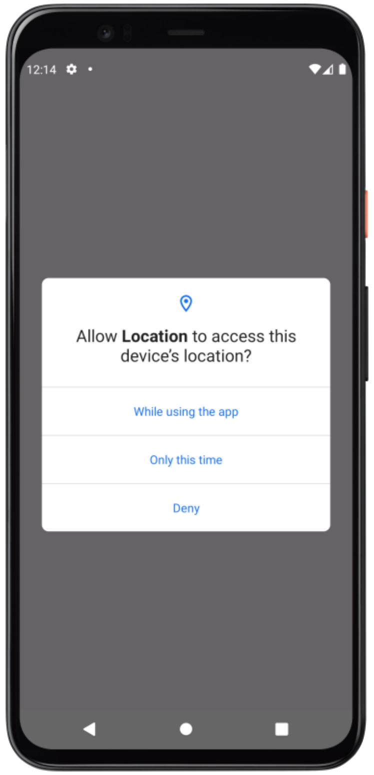
- Depending on the device, there are different alternatives to discover the location
  - This way, we use different **location providers** for implementing these alternatives
  - In Kotlin or Java, we select location providers using the class `LocationManager`
- Some common locations providers in Android are:
  - `LocationManager.`*`GPS_PROVIDER`* : Based on Global Positioning System (GPS), i.e., coordinates obtained by satellites
  - `LocationManager.`*`NETWORK_PROVIDER`* : This provider determines location based on nearby of cell tower and wifi access points. Operation of this provider may require a data connection
  - `LocationManager.`*`FUSED_PROVIDER`* : This provider may combine inputs from several other location providers to provide the best possible location

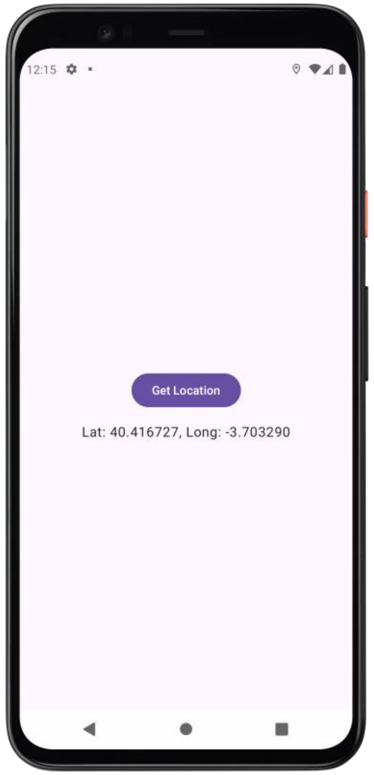
https://developer.android.com/reference/android/location/LocationManager

Fork me on GitHub

2. Location-based services - Location listener

```
@Composable
fun LocationApp(modifier: Modifier = Modifier) {
    val context = LocalContext.current
    var location by remember { mutableStateOf("") }
    var permissionsGranted by remember { mutableStateOf(false) }

    val permissionLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.RequestPermission()
    ) { isGranted ->
        permissionsGranted = isGranted
        if (!isGranted) {
            Toast.makeText(
                context, context.getString(R.string.permissions_denied), Toast.LENGTH_LONG
            ).show()
        }
    }
    if (permissionsGranted) {
        LaunchedEffect(Unit) {
            enableLocationManager(context) { loc ->
                location = context.getString(R.string.lat_long, loc.latitude, loc.longitude)
            }
        }
    }
    Column(
        modifier = modifier.fillMaxSize().padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Button(
            onClick = {
                permissionLauncher.launch(Manifest.permission.ACCESS_FINE_LOCATION)
            }) {
            Text(stringResource(R.string.get_location))
        }
        Spacer(modifier = Modifier.height(16.dp))
        Text(text = location, style = MaterialTheme.typography.bodyLarge)
    }
}
```





Lat: 40.416727, Long: -3.703290

```
@SuppressLint("MissingPermission")
fun enableLocationManager(context: Context, onLocationUpdate: (Location) -> Unit) {
    val locationManager = context.getSystemService(Context.LOCATION_SERVICE) as LocationManager
    val locationListener = LocationListener { location ->
        onLocationUpdate(location)
    }
    locationManager.requestLocationUpdates(
        LocationManager.GPS_PROVIDER, 0L, 1f, locationListener
    )
}
```

We register a location listener here

2. Location-based services - Geocoding

- **Geocoding** is the process of converting human-readable addresses (like "1600 Amphitheatre Parkway, Mountain View, CA") into geographic coordinates (like latitude 37.423021 and longitude -122.083739)
  - This process is sometimes referred as forward geocoding
  - Some uses of geocoding are: to place markers on a map, or centering a map
- **Reverse geocoding** is the process of converting geographic coordinates into a human-readable address
- We use the `Geocoder` class to implement both features in Android apps
  - We also need to declare the Internet connection permission

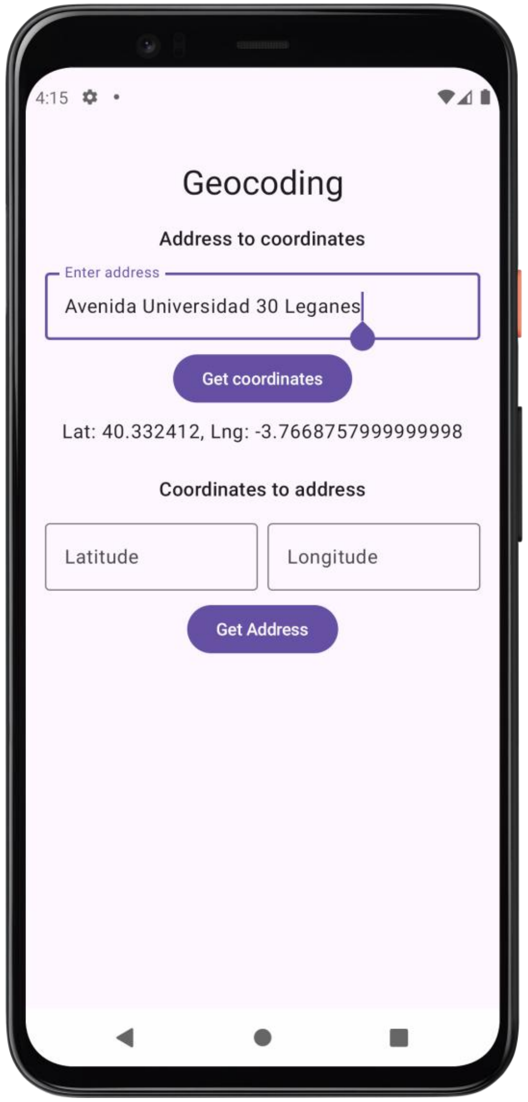
```
<uses-permission android:name="android.permission.INTERNET" />
```

https://developers.google.com/maps/documentation/geocoding/

Fork me on GitHub

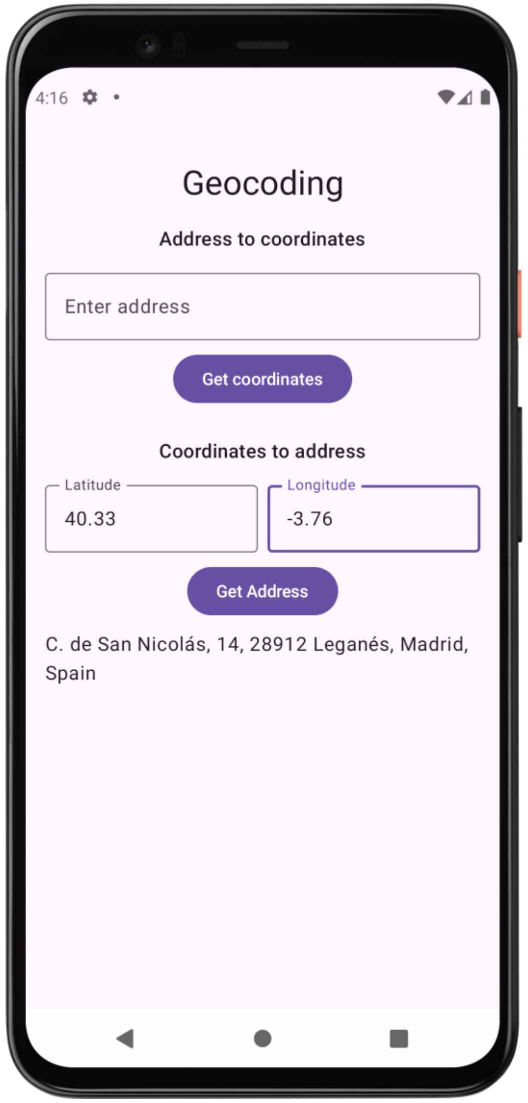
2. Location-based services - Geocoding

```kotlin
class GeocodingViewModel : ViewModel() {
    private val _address = MutableStateFlow("")
    val address: StateFlow<String> = _address

    private val _coordinates = MutableStateFlow("")
    val coordinates: StateFlow<String> = _coordinates

    private val _errorMessage = MutableStateFlow("")
    val errorMessage: StateFlow<String> = _errorMessage

    fun geocodeAddress(context: Context, address: String) {
        viewModelScope.launch {
            try {
                val geocoder = Geocoder(context, Locale.getDefault())
                val addresses = geocoder.getFromLocationName(address, 1)
                if (addresses?.isNotEmpty() == true) {
                    val location = addresses[0]
                    val lat = location.latitude
                    val lng = location.longitude
                    _coordinates.value = "Lat: $lat, Lng: $lng"
                    _errorMessage.value = ""
                }
            } catch (e: Exception) {
                setErrorMessage(e.message)
            }
        }
    }

    fun setErrorMessage(message: String?) {
        _errorMessage.value = message.orEmpty()
    }
}
```



2. Location-based services - Geocoding

```kotlin
class GeocodingViewModel : ViewModel() {
    private val _address = MutableStateFlow("")
    val address: StateFlow<String> = _address

    private val _coordinates = MutableStateFlow("")
    val coordinates: StateFlow<String> = _coordinates

    private val _errorMessage = MutableStateFlow("")
    val errorMessage: StateFlow<String> = _errorMessage

    fun reverseGeocode(context: Context, lat: Double, lng: Double) {
        viewModelScope.launch {
            try {
                val geocoder = Geocoder(context, Locale.getDefault())
                val addresses = geocoder.getFromLocation(lat, lng, 1)
                if (addresses?.isNotEmpty() == true) {
                    val address = addresses[0]
                    val addressText = (0..address.maxAddressLineIndex).joinToString("\n") {
                        address.getAddressLine(it)
                    }
                    _address.value = addressText
                    _errorMessage.value = ""
                }
            } catch (e: Exception) {
                setErrorMessage(e.message)
            }
        }
    }

    fun setErrorMessage(message: String?) {
        _errorMessage.value = message.orEmpty()
    }
}
```



C. de San Nicolás, 14, 28912 Leganés, Madrid, Spain

Table of contents

3. Google Maps Platform

- The Google Maps Platform (previously called Google Maps API) is a set of APIs and SDKs that allows to develop map-based services
- Some of the services provided by the Google Maps Platforms are:
  - Satellite imagery, aerial photography, street maps, 360° interactive panoramic views of streets (Street View), real-time traffic conditions, or route planning
- We can use the Google Maps Platforms for different types of apps, such as web, mobile, or desktop applications

Google Maps Platform

https://developers.google.com/maps

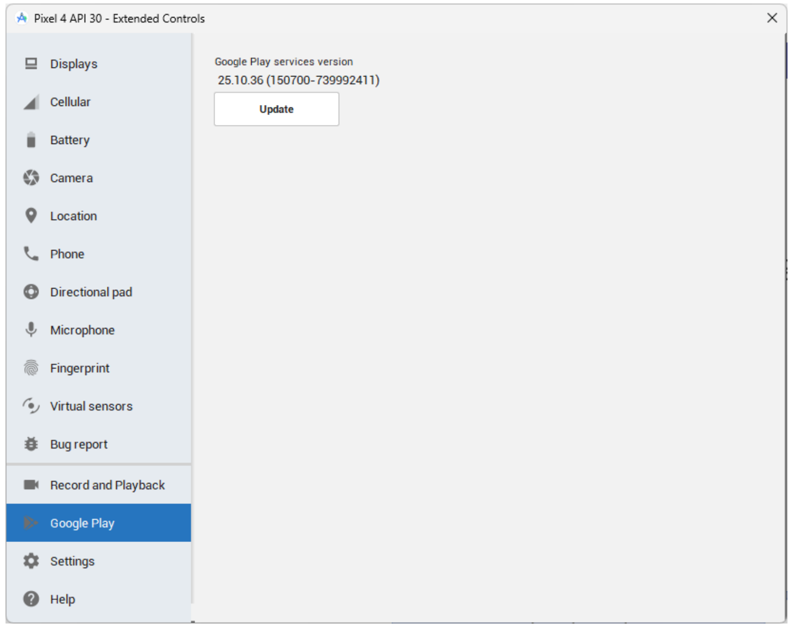
3. Google Maps Platform - Google Play Services

- The Google Maps SDK on Android depends on the Google Play services
  - Google Play Services is a background service produced by Google for Android devices
  - These services include maps and location, single sign-on account services, user health and fitness tracking, payment processing, integrated advertising or security scanning

Google Play services

https://developers.google.com/android/

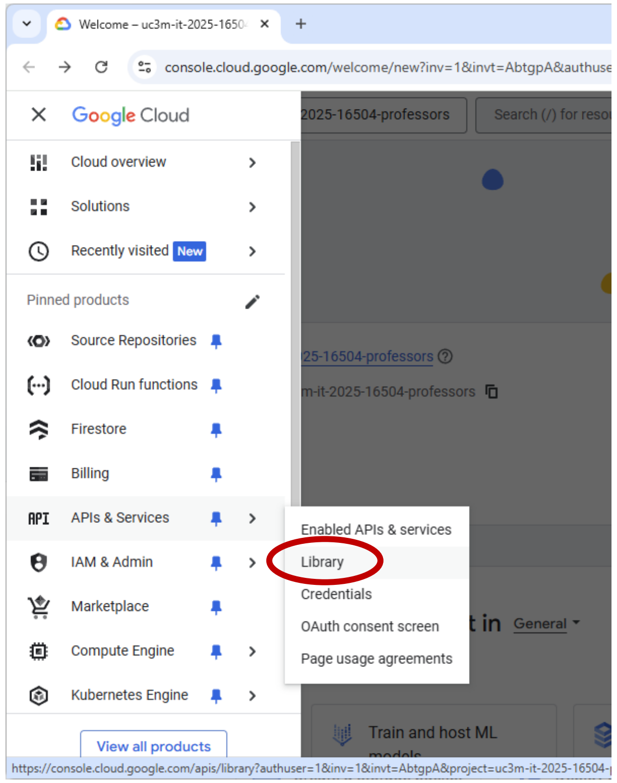
We need to ensure that Google Play is enabled in the emulator with use for development in Android Studio

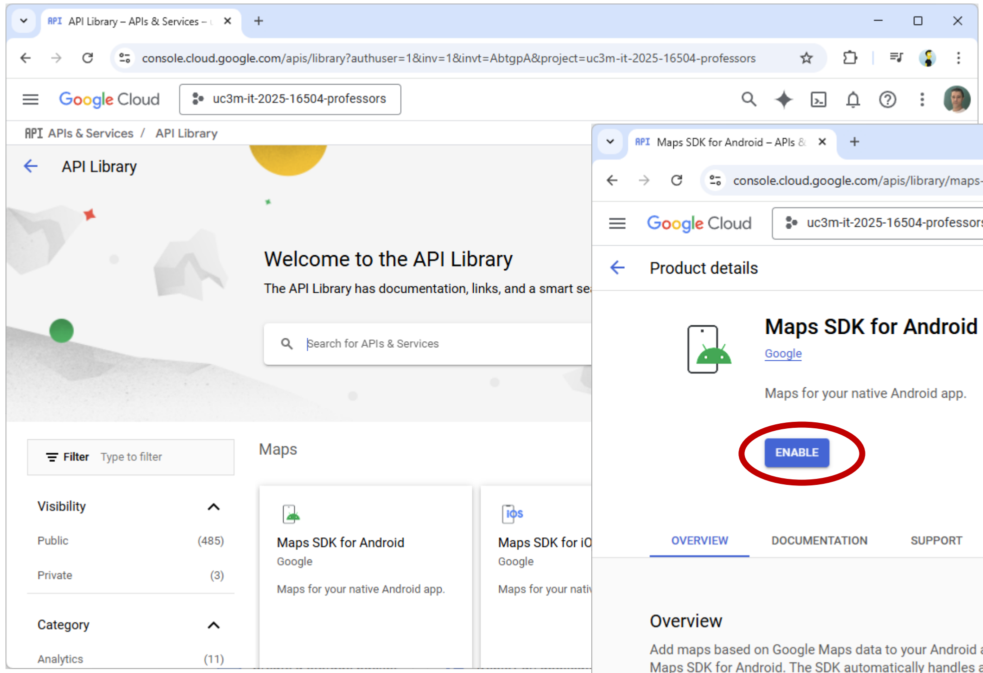


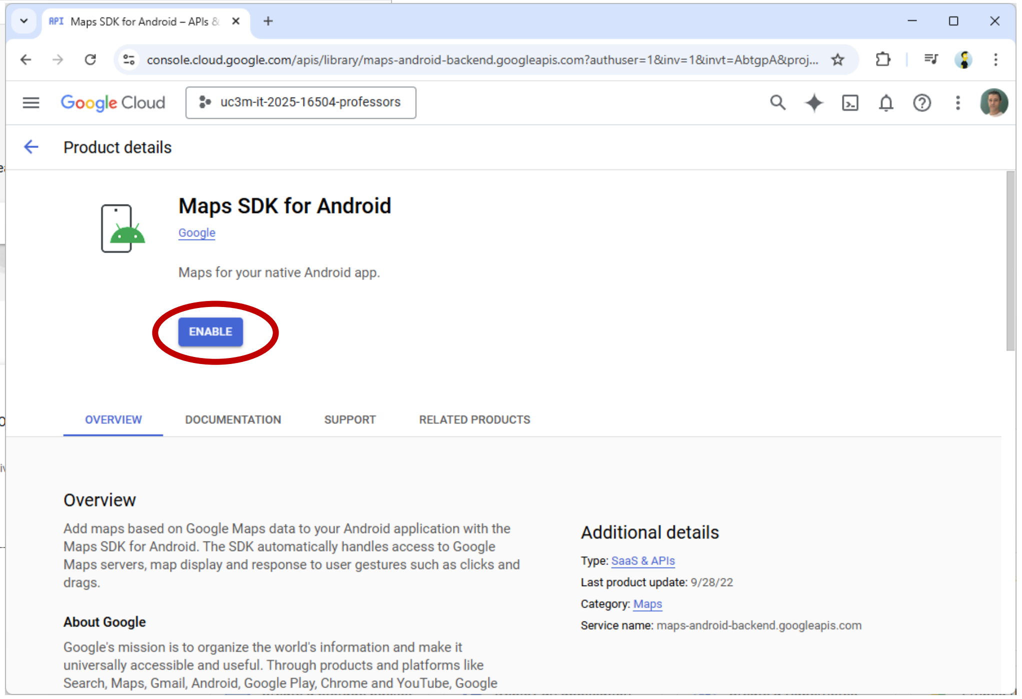
Google Play services version
25.10.36 (150700-739992411)

3. Google Maps Platform - Google Maps

- To use any API of Google Maps Platform, first we need we need enable this SDK in the Google Cloud console
  - APIs & Services → Library




API Library – APIs & Services –
console.cloud.google.com/apis/library?authuser=1&inv=1&invt=AbtgpA&project=uc3m-it-2025-16504-professors
Google Cloud
uc3m-it-2025-16504-professors
APIs & Services / API Library
API Library
Welcome to the API Library
The API Library has documentation, links, and a smart se
Search for APIs & Services
Filter Type to filter
Maps
Visibility
Public (485)
Private (3)
Category
Analytics (11)
Maps SDK for Android
Google
Maps for your native Android app.
Maps SDK for iO
Google
Maps for your nati


Maps SDK for Android – APIs &
console.cloud.google.com/apis/library/maps-android-backend.googleapis.com?authuser=1&inv=1&invt=AbtgpA&proj...
Google Cloud
uc3m-it-2025-16504-professors
Product details
Maps SDK for Android
Google
Maps for your native Android app.
ENABLE
OVERVIEW
DOCUMENTATION
SUPPORT
RELATED PRODUCTS
Overview
Additional details
Type: SaaS & APIs
Last product update: 9/28/22
Category: Maps
Service name: maps-android-backend.googleapis.com
About Google

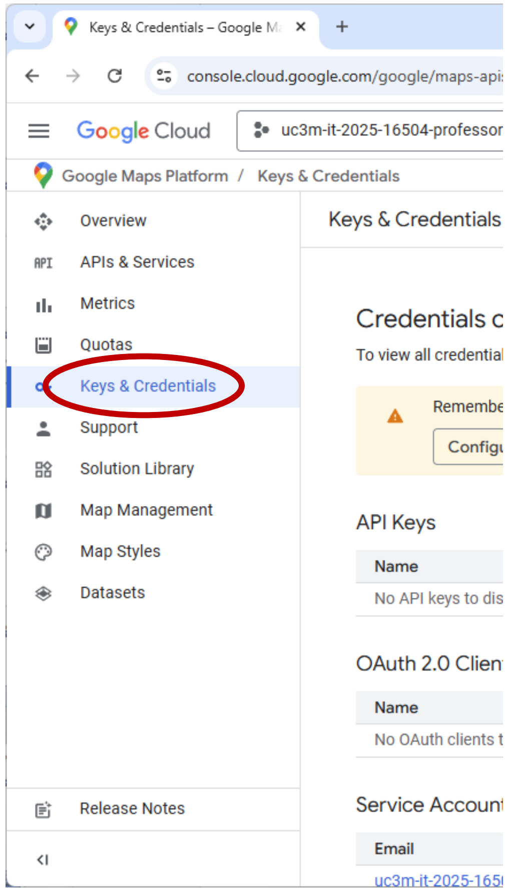
Add maps based on Google Maps data to your Android application with the Maps SDK for Android. The SDK automatically handles access to Google Maps servers, map display and response to user gestures such as clicks and drags.

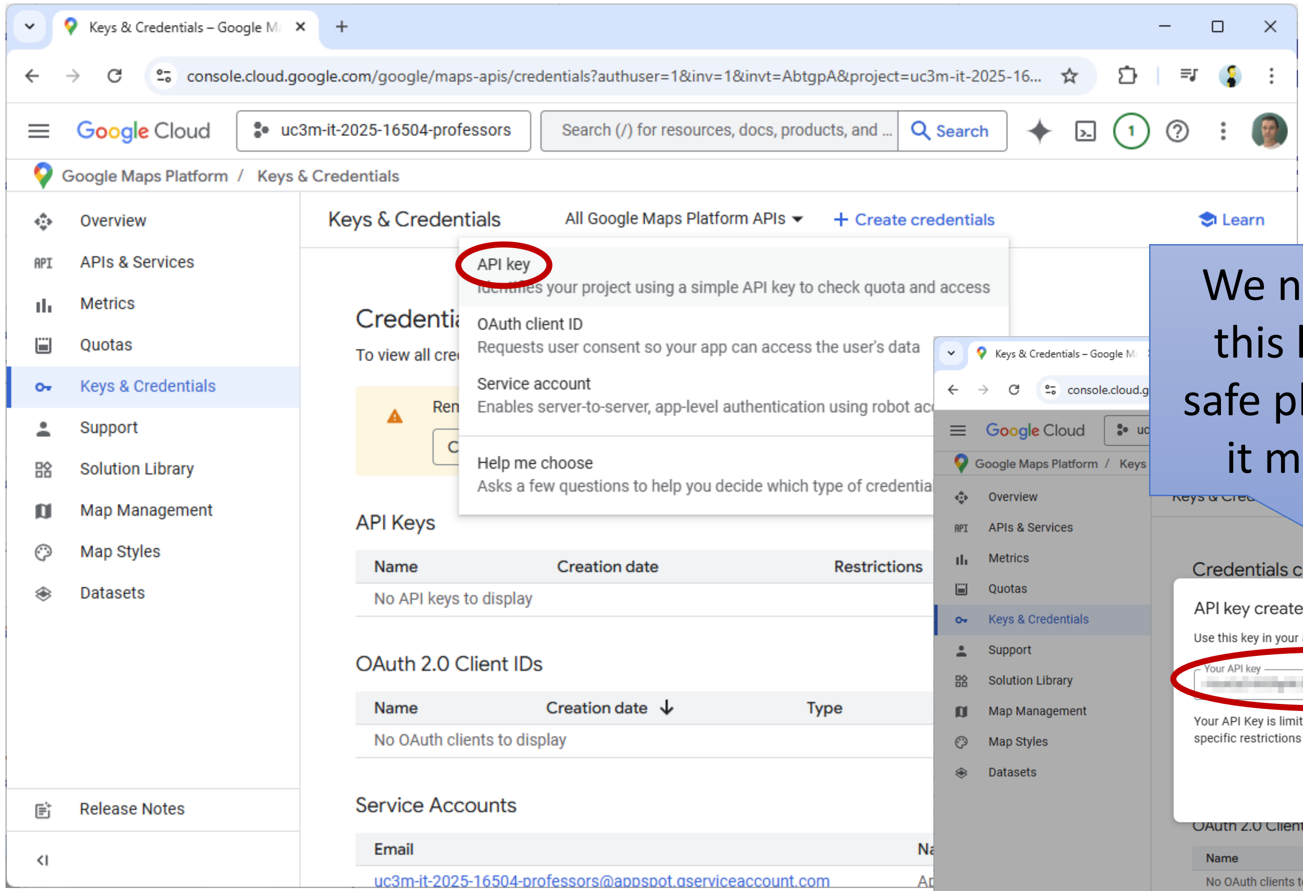
Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google

https://console.cloud.google.com/

3. Google Maps Platform - Google Maps

- Then, we need to create an API key:



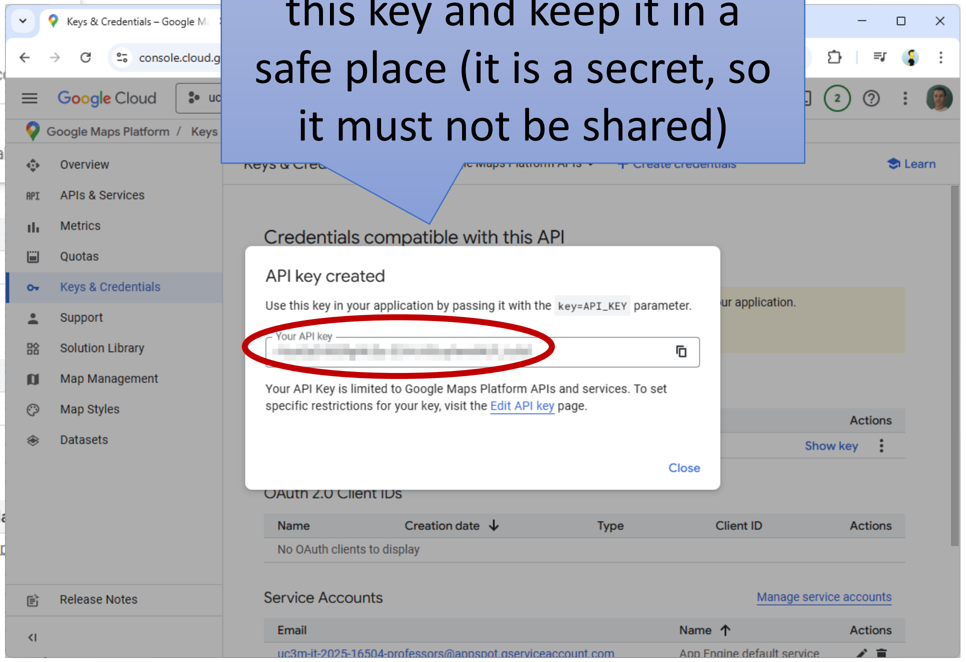

Keys & Credentials – Google M
console.cloud.google.com/google/maps-apis/credentials?authuser=1&inv=1&invt=AbtgpA&project=uc3m-it-2025-16...
Google Cloud
uc3m-it-2025-16504-professors
Search (/) for resources, docs, products, and ...
Search
Google Maps Platform / Keys & Credentials
Overview
APIs & Services
Metrics
Quotas
Keys & Credentials
Support
Solution Library
Map Management
Map Styles
Datasets
Release Notes
Keys & Credentials
All Google Maps Platform APIs
+ Create credentials
Learn
Credenti
To view all cre
Ren
API Keys
Name
Creation date
Restrictions
No API keys to display
OAuth 2.0 Client IDs
Name
Creation date
Type
No OAuth clients to display
Service Accounts
Email
uc3m-it-2025-16504-professors@appspot.gserviceaccount.com

API key
Identifies your project using a simple API key to check quota and access

OAuth client ID
Requests user consent so your app can access the user's data

Service account
Enables server-to-server, app-level authentication using robot ac

Help me choose
Asks a few questions to help you decide which type of credentia



API key created

Use this key in your application by passing it with the `key=API_KEY` parameter.

Your API key

Your API Key is limited to Google Maps Platform APIs and services. To set specific restrictions for your key, visit the Edit API key page.

Close

We need to copy a paste this key and keep it in a safe place (it is a secret, so it must not be shared)

https://console.cloud.google.com/

3. Google Maps Platform - Google Maps

- To use Google Maps in an Android app, we need to declare the following dependency in our `build.gradle.kts` (app):

build.gradle.kts (app)

```
dependencies {
    implementation(libs.maps.compose)
    implementation(libs.play.services.maps)
}
```

libs.version.toml

```
[versions]
mapsCompose = "4.3.3"
playServicesMaps = "19.1.0"

[libraries]
maps-compose = { module = "com.google.maps.android:maps-compose", version.ref = "mapsCompose" }
play-services-maps = { module = "com.google.android.gms:play-services-maps", version.ref = "playServicesMaps" }
```

- In addition, we need to declare this permissions in the manifest:

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

- Finally, we need to declare an API Key in our manifest:

```
    <application
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="${MAPS_API_KEY}" />
    </application>
```

For security, it is not a good practice to hard-code the API key in the manifest

3. Google Maps Platform - Google Maps

- We can use a Gradle plugin to store the API key safely in the `local.properties` file (which in local, and should not be public):

build.gradle.kts (project)

```
plugins {
    // ...
    alias(libs.plugins.secrets.gradle) apply false
}
```

build.gradle.kts (app)

```
plugins {
    // ...
    alias(libs.plugins.secrets.gradle)
}

android {
    // ...
    buildFeatures {
        // ...
        buildConfig = true
    }
}
```

libs.version.toml

```
[versions]
secrets-gradle = "2.0.1"

[plugins]
secrets-gradle = { id = "com.google.android.libraries.mapsplatform.secrets-gradle-plugin", version.ref = "secrets-gradle" }
```

local.properties

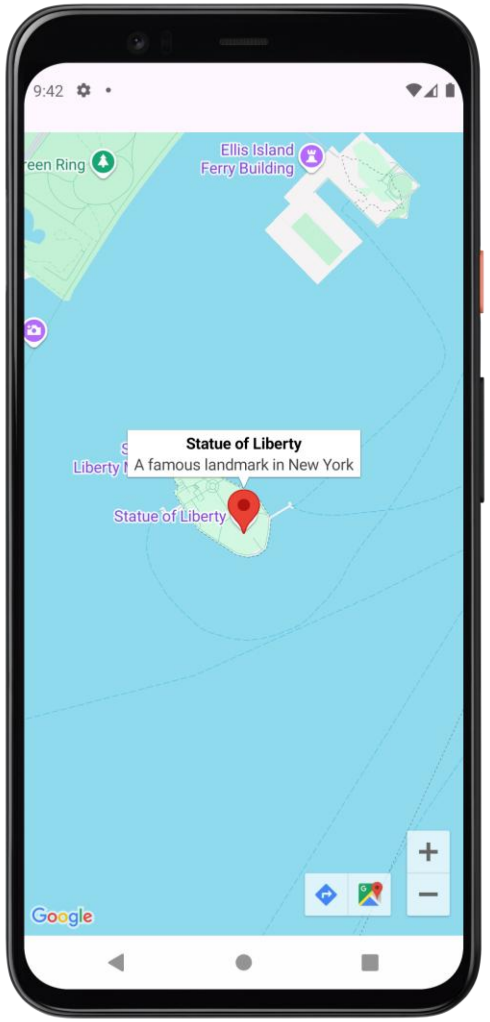
```
MAPS_API_KEY=<my-apy-key>
```

Fork me on GitHub

3. Google Maps Platform - Google Maps

```
@Composable
fun MapScreen(modifier: Modifier = Modifier) {
    val statueOfLiberty = LatLng(40.6892, -74.0445)
    val cameraPositionState = rememberCameraPositionState {
        position = CameraPosition.fromLatLngZoom(statueOfLiberty, 15f)
    }

    GoogleMap(
        modifier = modifier.fillMaxSize(),
        cameraPositionState = cameraPositionState
    ) {
        Marker(
            state = MarkerState(position = statueOfLiberty),
            title = stringResource(R.string.title),
            snippet = stringResource(R.string.description)
        )
    }
}
```

This basic demo renders a map with Google Maps in a fixed coordinates a with a Marker



https://developers.google.com/maps/documentation/android-sdk/maps-compose

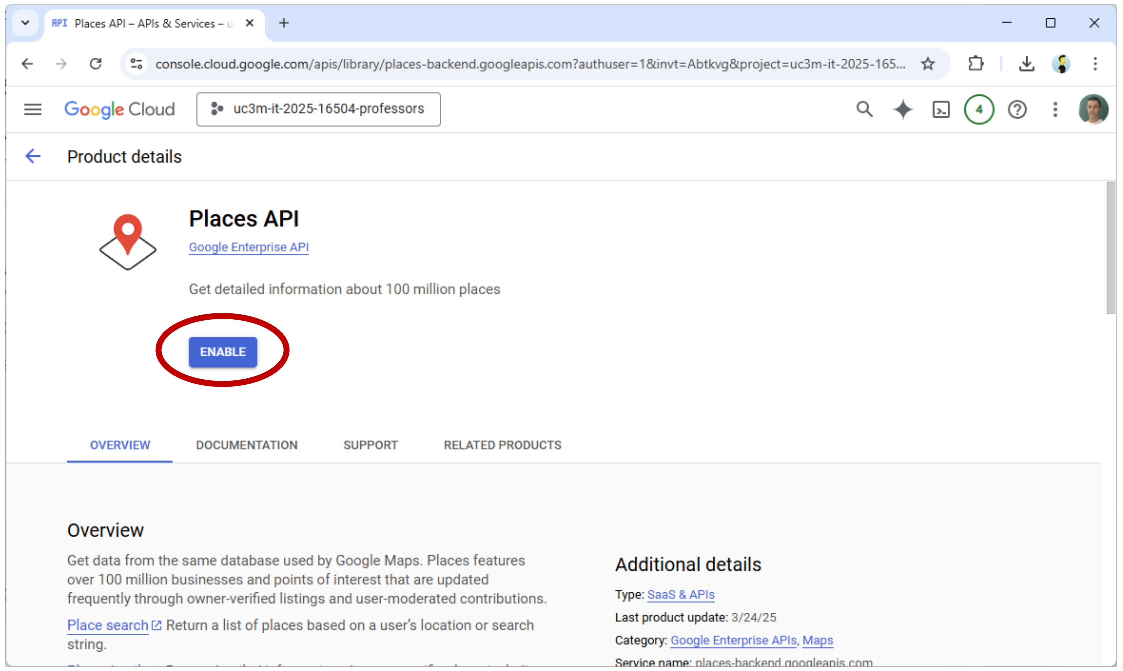
3. Google Maps Platform - Google Places

- The **Google Places API** is a web service provided by Google that allows developers to access detailed information about places (businesses, landmarks, geographic locations, etc.)
  - We are going to consume this API using a Java wrapper library (i.e., not requesting directly the web service API)
- Google Places is part of the Google Maps Platform and enables applications to search for places, retrieve place details, or auto-complete place names based on user input

https://developers.google.com/maps/documentation/places/android-sdk/overview

3. Google Maps Platform - Google Places

- To use Google Places, first we need to enable it in the Google Cloud console:
  - APIs & Services → Library


Places API – APIs & Services – u
console.cloud.google.com/apis/library/places-backend.googleapis.com?authuser=1&invt=Abtkvg&project=uc3m-it-2025-165...
Google Cloud
uc3m-it-2025-16504-professors
ENABLE
OVERVIEW
DOCUMENTATION
SUPPORT
RELATED PRODUCTS

Product details

Places API

Google Enterprise API

Get detailed information about 100 million places

Overview

Get data from the same database used by Google Maps. Places features over 100 million businesses and points of interest that are updated frequently through owner-verified listings and user-moderated contributions.

Place search Return a list of places based on a user's location or search string.

Additional details

Type: SaaS & APIs

Last product update: 3/24/25

Category: Google Enterprise APIs, Maps

Service name: places-backend.googleapis.com

https://console.cloud.google.com/

3. Google Maps Platform - Google Places

- To use Google Places, we need to declare the following permissions in the manifest:

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

- Then, we need to include the following dependency in our project:

build.gradle.kts (app)

```
dependencies {
    implementation(libs.places)
}
```

libs.version.toml

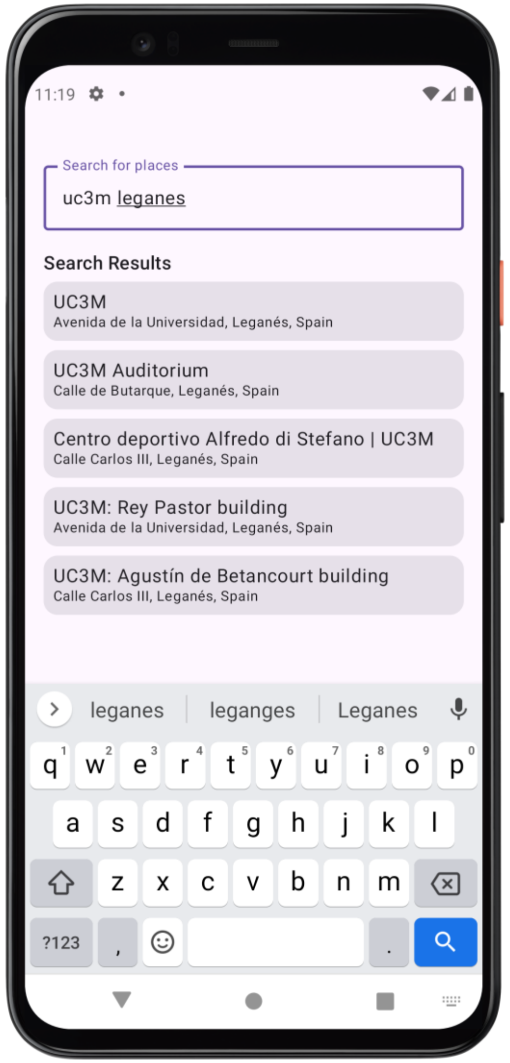
```
[versions]
places = "4.1.0"

[libraries]
places = { module = "com.google.android.libraries.places:places", version.ref = "places" }
```

Fork me on GitHub

3. Google Maps Platform - Google Places

```kotlin
class PlacesViewModel : ViewModel() {
    private lateinit var placesClient: PlacesClient

    private val _predictions = MutableStateFlow<List<PlaceAutocomplete>>(emptyList())
    val predictions: StateFlow<List<PlaceAutocomplete>> = _predictions

    fun initializePlaces(context: Context) {
        if (!Places.isInitialized()) {
            Places.initialize(context, BuildConfig.MAPS_API_KEY)
        }
        placesClient = Places.createClient(context)
    }

    fun searchPlaces(query: String) {
        viewModelScope.launch {
            val request = FindAutocompletePredictionsRequest.builder().setQuery(query).build()

            placesClient.findAutocompletePredictions(request).addOnSuccessListener { response ->
                _predictions.value = response.autocompletePredictions.map { prediction ->
                    PlaceAutocomplete(
                        placeId = prediction.placeId,
                        primaryText = prediction.getPrimaryText(null).toString(),
                        secondaryText = prediction.getSecondaryText(null).toString()
                    )
                }
            }.addOnFailureListener { exception ->
                exception.printStackTrace()
            }
        }
    }

    // ...

}

data class PlaceAutocomplete(
    val placeId: String, val primaryText: String, val secondaryText: String
)
```

We also need an API key for Google Places (we can use the same previously created)



Search Results

UC3M
Avenida de la Universidad, Leganés, Spain

UC3M Auditorium
Calle de Butarque, Leganés, Spain

Centro deportivo Alfredo di Stefano | UC3M
Calle Carlos III, Leganés, Spain

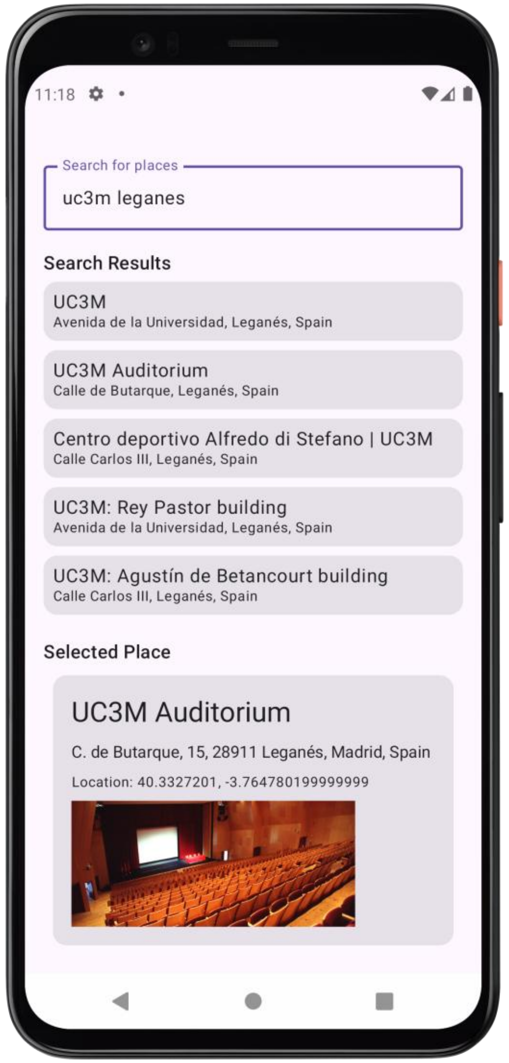
UC3M: Rey Pastor building
Avenida de la Universidad, Leganés, Spain

UC3M: Agustín de Betancourt building
Calle Carlos III, Leganés, Spain

Fork me on GitHub

3. Google Maps Platform - Google Places

```kotlin
class PlacesViewModel : ViewModel() {
    private val _selectedPlace = MutableStateFlow<PlaceDetails?>(null)
    val selectedPlace: StateFlow<PlaceDetails?> = _selectedPlace

    // ...

    fun getPlaceDetails(placeId: String) {
        viewModelScope.launch {
            val placeFields = listOf(
                Place.Field.ID, Place.Field.NAME, Place.Field.ADDRESS,
                Place.Field.LAT_LNG, Place.Field.PHOTO_METADATAS
            )
            val request = FetchPlaceRequest.builder(placeId, placeFields).build()

            placesClient.fetchPlace(request).addOnSuccessListener { response ->
                val place = response.place
                place.photoMetadatas?.first()?.let {
                    val photoRequest = FetchPhotoRequest.builder(it).build()
                    placesClient.fetchPhoto(photoRequest).addOnSuccessListener { response ->
                        _selectedPlace.value = PlaceDetails(
                            name = place.displayName ?: "Unknown",
                            address = place.formattedAddress ?: "No address",
                            latLng = place.location ?: LatLng(0.0, 0.0),
                            bitmap = response.bitmap
                        )
                    }.addOnFailureListener { exception ->
                        exception.printStackTrace()
                    }
                }
            }.addOnFailureListener { exception ->
                exception.printStackTrace()
            }
        }
    }

}

data class PlaceDetails(
    val name: String, val address: String, val latLng: LatLng, var bitmap: Bitmap
)
```



uc3m leganes

Search Results

UC3M
Avenida de la Universidad, Leganés, Spain

UC3M Auditorium
Calle de Butarque, Leganés, Spain

Centro deportivo Alfredo di Stefano | UC3M
Calle Carlos III, Leganés, Spain

UC3M: Rey Pastor building
Avenida de la Universidad, Leganés, Spain

UC3M: Agustín de Betancourt building
Calle Carlos III, Leganés, Spain

Selected Place

UC3M Auditorium
C. de Butarque, 15, 28911 Leganés, Madrid, Spain
Location: 40.3327201, -3.764780199999999

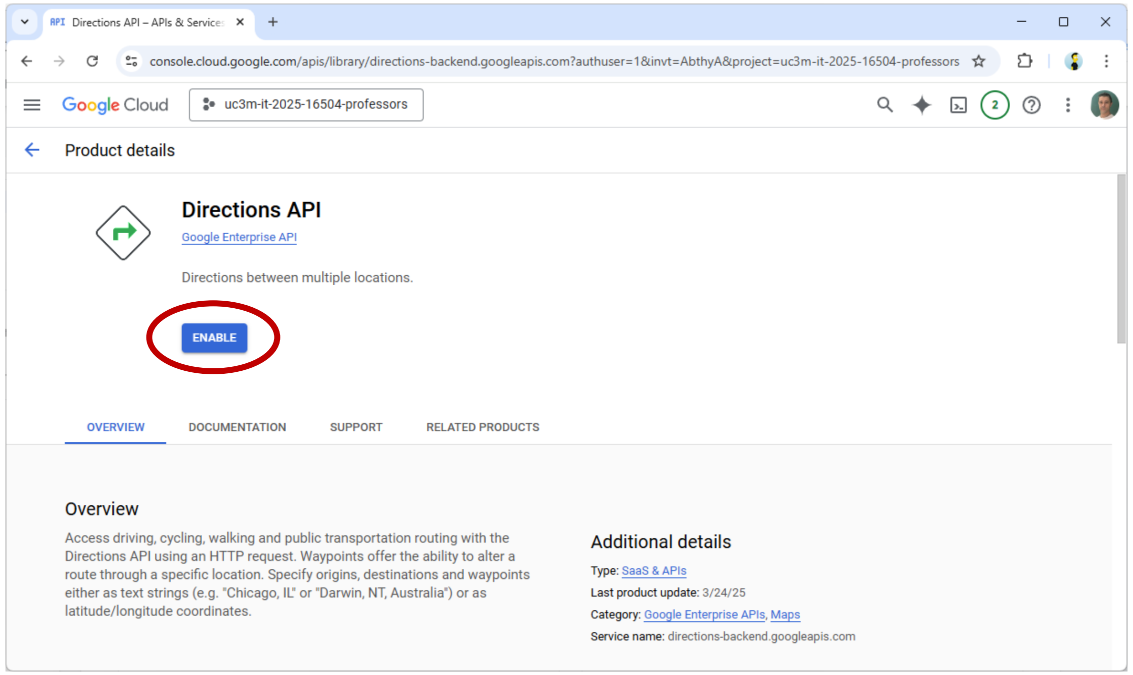
3. Google Maps Platform - Google Directions

- The **Google Directions API** is a web service provided by Google as part of the Google Maps Platform that calculates directions between multiple locations
  - It returns detailed route information, including travel time, distance, turn-by-turn navigation steps, and even alternative routes for different travel modes (driving, walking, cycling, or public transit)
  - Like we do with Google Places, we consume this API using a Java wrapper library (i.e., not requesting directly the web service API)

https://developers.google.com/maps/documentation/directions

3. Google Maps Platform - Google Directions

- To use Google Directions, first we need to enable it in the Google Cloud console:
  - APIs & Services → Library


Directions API – APIs & Services
console.cloud.google.com/apis/library/directions-backend.googleapis.com?authuser=1&invt=AbthyA&project=uc3m-it-2025-16504-professors
Google Cloud
uc3m-it-2025-16504-professors
Product details
ENABLE
OVERVIEW
DOCUMENTATION
SUPPORT
RELATED PRODUCTS

Directions API

Google Enterprise API

Directions between multiple locations.

Overview

Access driving, cycling, walking and public transportation routing with the Directions API using an HTTP request. Waypoints offer the ability to alter a route through a specific location. Specify origins, destinations and waypoints either as text strings (e.g. "Chicago, IL" or "Darwin, NT, Australia") or as latitude/longitude coordinates.

Additional details

Type: SaaS & APIs
Last product update: 3/24/25
Category: Google Enterprise APIs, Maps
Service name: directions-backend.googleapis.com

https://console.cloud.google.com/

3. Google Maps Platform - Google Directions

- To use Google Directions, we need to declare the following permissions in the manifest:

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

- Then, we need the following dependency:

build.gradle.kts (app)

```
dependencies {
    implementation(libs.google.google.maps.services)
}
```

libs.version.toml

```
[versions]
googleGoogleMapsServices = "2.2.0"

[libraries]
google-google-maps-services = { module = "com.google.maps:google-maps-services", version.ref = "googleGoogleMapsServices" }
```

Fork me on GitHub

3. Google Maps Platform - Google Directions

```kotlin
fun getDirections(
    origin: String, destination: String, callback: (DirectionsResult?, String?) ->
Unit
) {
    val context = GeoApiContext.Builder().apiKey(BuildConfig.MAPS_API_KEY).build()

    DirectionsApi.newRequest(context).mode(TravelMode.DRIVING).origin(origin)
        .destination(destination).alternatives(true)
        .setCallback(object : PendingResult.Callback<DirectionsResult> {
            override fun onResult(result: DirectionsResult) {
                callback(result, null)
            }

            override fun onFailure(e: Throwable) {
                callback(null, e.message)
            }
        })
}
```

```kotlin
@Composable
fun DrawRoute(route: DirectionsRoute) {
    // Convert the polyline points to LatLng objects
    val pathPoints = remember(route) {
        route.overviewPolyline.decodePath().map { LatLng(it.lat, it.lng) }
    }

    // Draw the polyline
    Polyline(
        points = pathPoints,
        color = Color.Blue,
        width = 8f
    )
}
```

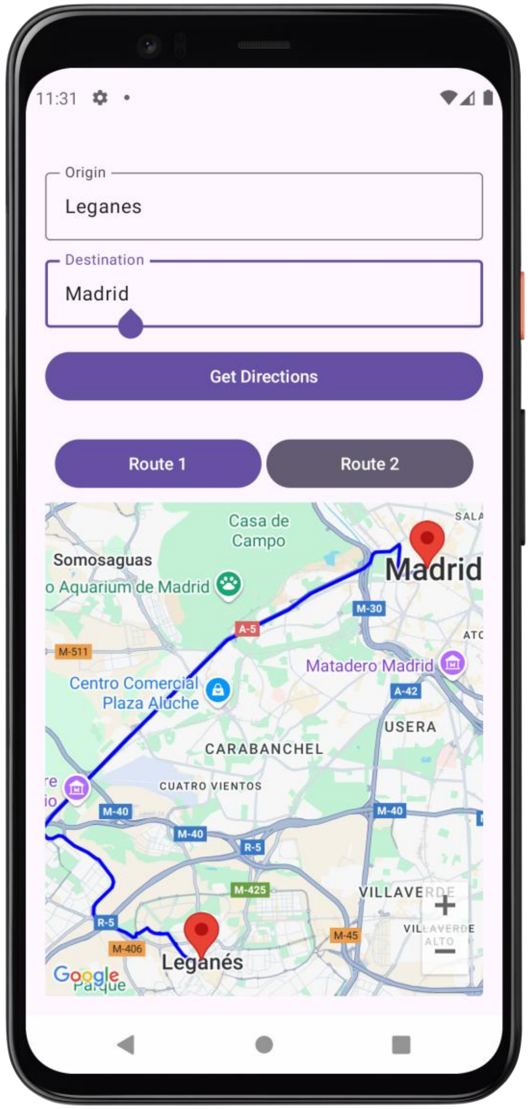


Table of contents

4. Takeaways

- Location-based services (LBS) allows mobile device applications to known where the device is geographically located
- There are different location providers in Android, such as GPS or network based
- We can use LBS to implement different features in Android apps, such as location listener or geocoding (forward or reverse)
- The Google Maps Platform is a set of APIs and SDKs that allows to develop map-based services
- We can use these APIs in Android app to implement map-based features, such as render maps and put markers on it, search places, or get directions