Lab session 5: Create Geolocation Flutter application with Google Maps and Open Street Maps.

Task-1 Adding Google Maps to a Flutter App

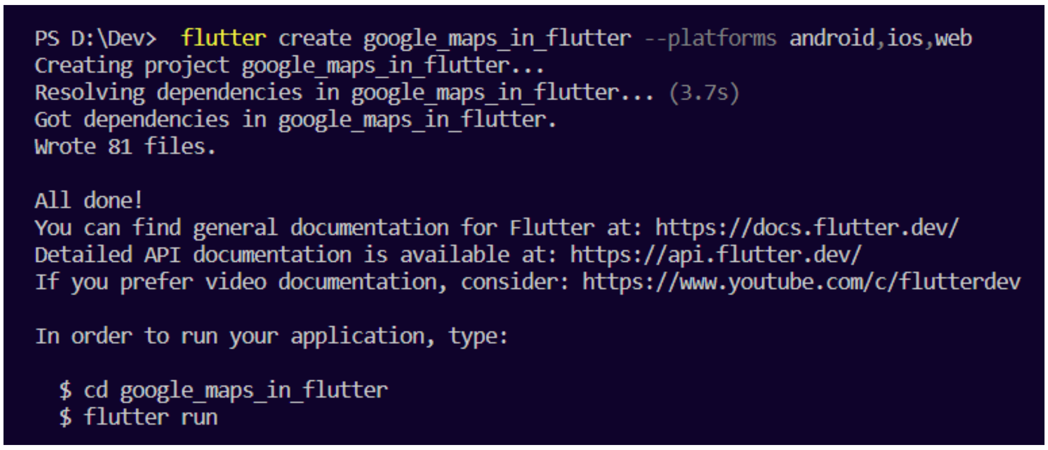
For this task, we will build a mobile app featuring a **Google Map** using the Flutter SDK.

The app will:

- Display a Google Map
- Retrieve map data from a web service
- Display this data as markers on the Map

Getting started with Flutter

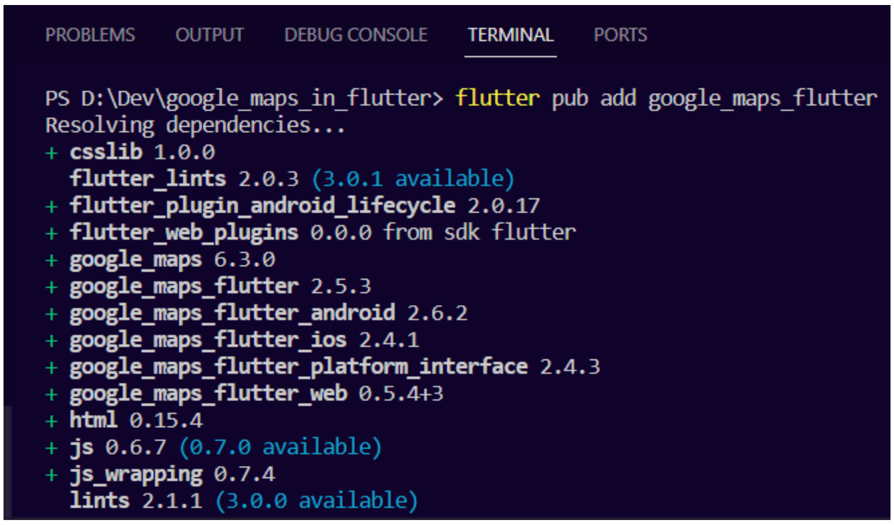
We will use the flutter command line tool to create a new project with the necessary platforms.



```
PS D:\Dev>  flutter create google_maps_in_flutter --platforms android,ios,web
Creating project google_maps_in_flutter...
Resolving dependencies in google_maps_in_flutter... (3.7s)
Got dependencies in google_maps_in_flutter.
Wrote 81 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

In order to run your application, type:

  $ cd google_maps_in_flutter
  $ flutter run
```

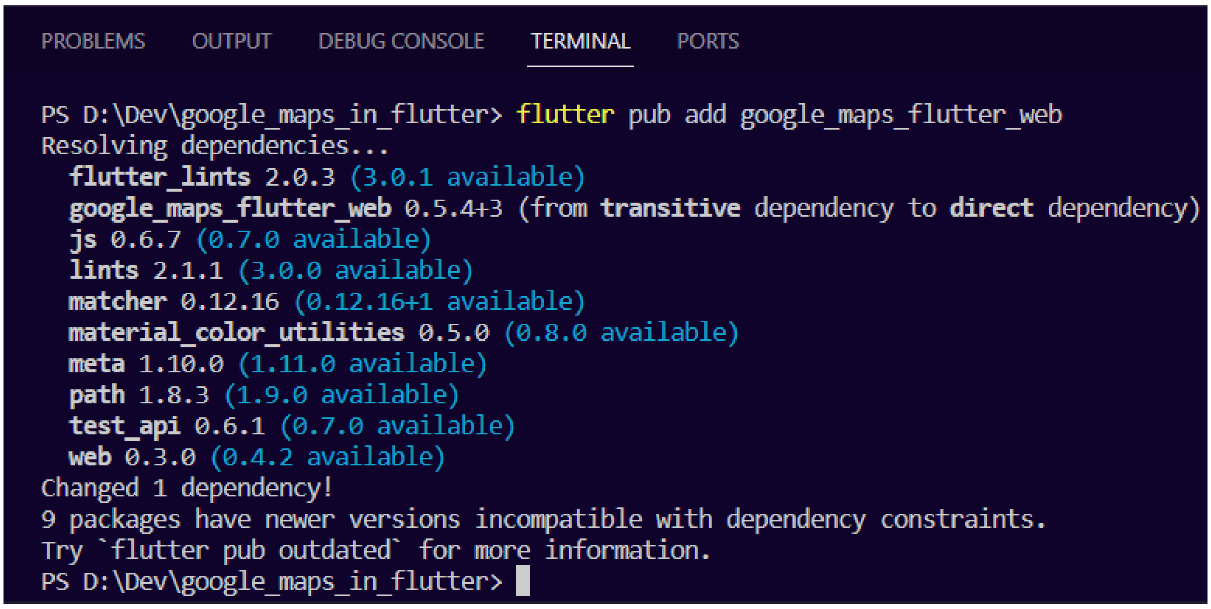
Adding Google Maps Flutter plugin as a dependency

To be able to add Google Map to our app we have to use the Google Maps Flutter plugin by running the following command from the project directory.



```
PS D:\Dev\google_maps_in_flutter> flutter pub add google_maps_flutter
Resolving dependencies...
+ csslib 1.0.0
  flutter_lints 2.0.3 (3.0.1 available)
+ flutter_plugin_android_lifecycle 2.0.17
+ flutter_web_plugins 0.0.0 from sdk flutter
+ google_maps 6.3.0
+ google_maps_flutter 2.5.3
+ google_maps_flutter_android 2.6.2
+ google_maps_flutter_ios 2.4.1
+ google_maps_flutter_platform_interface 2.4.3
+ google_maps_flutter_web 0.5.4+3
+ html 0.15.4
+ js 0.6.7 (0.7.0 available)
+ js_wrapping 0.7.4
  lints 2.1.1 (3.0.0 available)
```

As we will also see how to use Google Maps in **Flutter for Web,** we need to also add **Web version of the plugin** to the project as it is not yet federated.



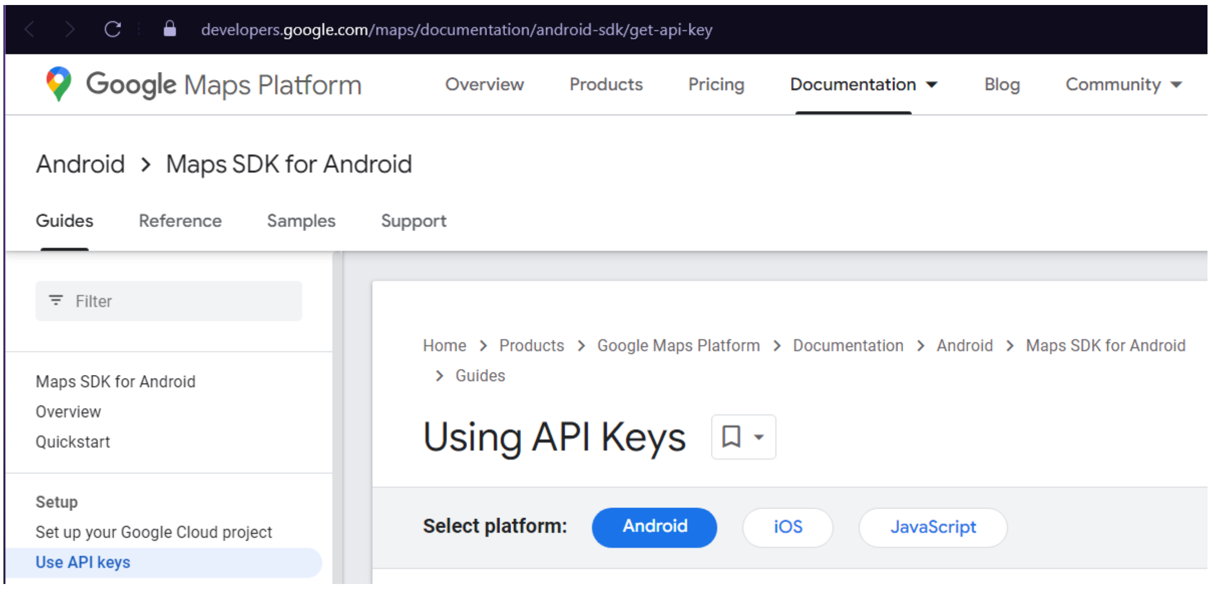
```
PS D:\Dev\google_maps_in_flutter> flutter pub add google_maps_flutter_web
Resolving dependencies...
  flutter_lints 2.0.3 (3.0.1 available)
  google_maps_flutter_web 0.5.4+3 (from transitive dependency to direct dependency)
  js 0.6.7 (0.7.0 available)
  lints 2.1.1 (3.0.0 available)
  matcher 0.12.16 (0.12.16+1 available)
  material_color_utilities 0.5.0 (0.8.0 available)
  meta 1.10.0 (1.11.0 available)
  path 1.8.3 (1.9.0 available)
  test_api 0.6.1 (0.7.0 available)
  web 0.3.0 (0.4.2 available)
Changed 1 dependency!
9 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
PS D:\Dev\google_maps_in_flutter>
```

Note:

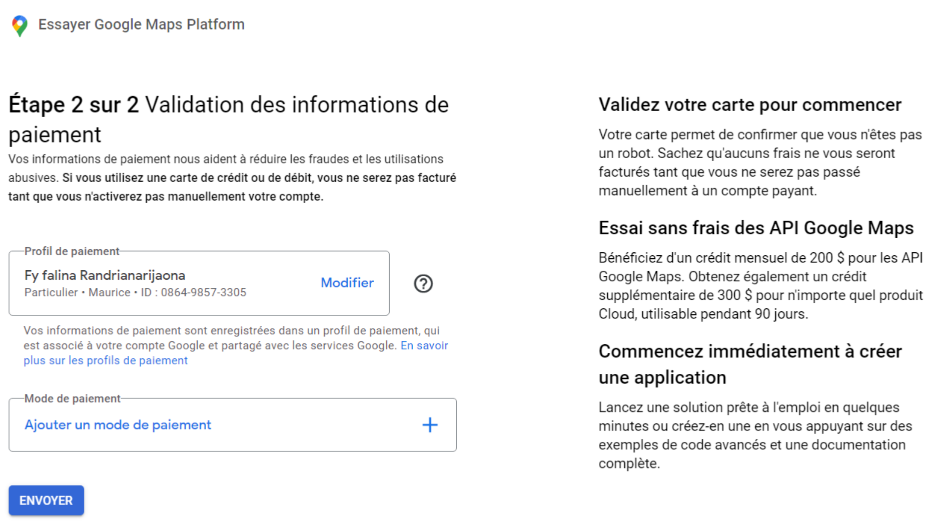
- To get the latest version of the Google Maps SDK on iOS requires a platform minimum version of iOS 13.
- To use Google Maps SDK on Android requires setting the minSDK to 20.

Adding Google Maps to the app

To use Google Maps in a Flutter app, we need to configure an API project with the Google Maps Platform, following the Maps SDK for Android's Using API key, Maps SDK for iOS' Using API key, and Maps JavaScript API's Using API key.


developers.google.com/maps/documentation/android-sdk/get-api-key
Google Maps Platform
Overview
Products
Pricing
Documentation
Blog
Community
Android > Maps SDK for Android
Guides
Reference
Samples
Support
Filter
Maps SDK for Android
Overview
Quickstart
Setup
Set up your Google Cloud project
Use API keys
Home > Products > Google Maps Platform > Documentation > Android > Maps SDK for Android > Guides
Using API Keys
Select platform:
Android
iOS
JavaScript

Unfortunately, Google Maps Platform prompts the user to setup billing accounts before letting users create API keys.



Essayer Google Maps Platform

Étape 2 sur 2 Validation des informations de paiement

Vos informations de paiement nous aident à réduire les fraudes et les utilisations abusives. **Si vous utilisez une carte de crédit ou de débit, vous ne serez pas facturé tant que vous n'activerez pas manuellement votre compte.**

Profil de paiement
Fy falina Randrianarijaona
Particulier • Maurice • ID : 0864-9857-3305
Modifier

Vos informations de paiement sont enregistrées dans un profil de paiement, qui est associé à votre compte Google et partagé avec les services Google. En savoir plus sur les profils de paiement

Mode de paiement
Ajouter un mode de paiement

ENVOYER

Validez votre carte pour commencer

Votre carte permet de confirmer que vous n'êtes pas un robot. Sachez qu'aucuns frais ne vous seront facturés tant que vous ne serez pas passé manuellement à un compte payant.

Essai sans frais des API Google Maps

Bénéficiez d'un crédit mensuel de 200 $ pour les API Google Maps. Obtenez également un crédit supplémentaire de 300 $ pour n'importe quel produit Cloud, utilisable pendant 90 jours.

Commencez immédiatement à créer une application

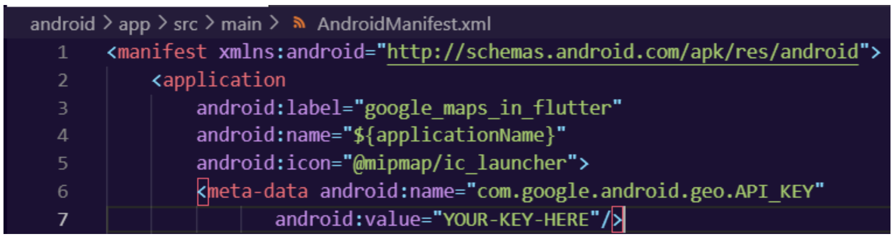
Lancez une solution prête à l'emploi en quelques minutes ou créez-en une en vous appuyant sur des exemples de code avancés et une documentation complète.

For security reason, we prefer not to input the billing information so we will not be able to test the implementation correctly.

Adding an API key for an Android app

To add an API key to the Android app, we need to edit the **AndroidManifest.xml** file in *android/app/src/main*.

Add a single meta-data entry containing the API key created in the previous step inside the application node.



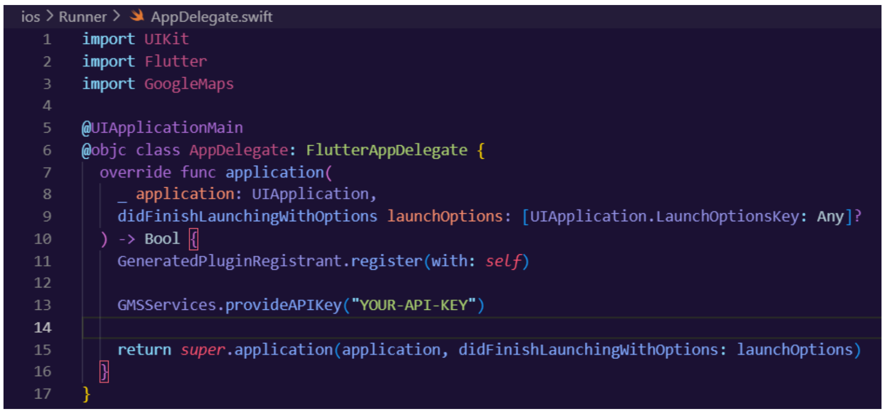
android > app > src > main > AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <application
        android:label="google_maps_in_flutter"
        android:name="${applicationName}"
        android:icon="@mipmap/ic_launcher">
        <meta-data android:name="com.google.android.geo.API_KEY"
                   android:value="YOUR-KEY-HERE"/>
```

Adding an API key for an iOS app

To add an API key to the iOS app, we need to edit the **AppDelegate.swift** file in *ios/Runner*.

We will make two changes to this file. First, add an **#import** statement to pull in the Google Maps headers, and then call the **provideAPIKey()** method of the GMSServices singleton. This API key enables Google Maps to correctly serve map tiles.

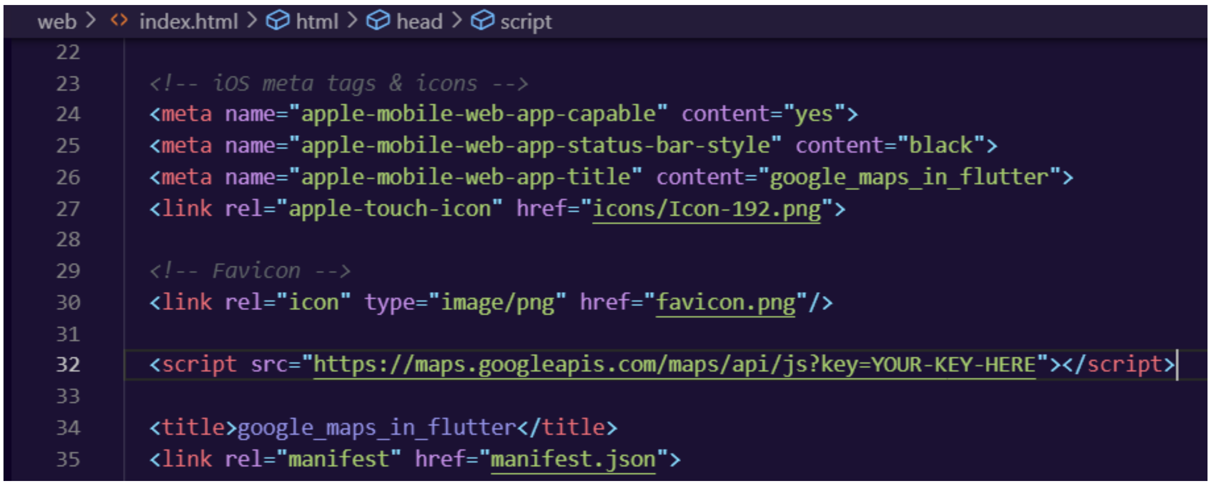


ios > Runner > AppDelegate.swift

```
import UIKit
import Flutter
import GoogleMaps

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
  override func application(
    _ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
  ) -> Bool {
    GeneratedPluginRegistrant.register(with: self)

    GMSServices.provideAPIKey("YOUR-API-KEY")

    return super.application(application, didFinishLaunchingWithOptions: launchOptions)
  }
}
```

Adding an API key for a Web app

To add an API key to the Web app, we need to edit the **index.html** file in *web*. Add a reference to the Maps JavaScript script in the head section, with your API key.

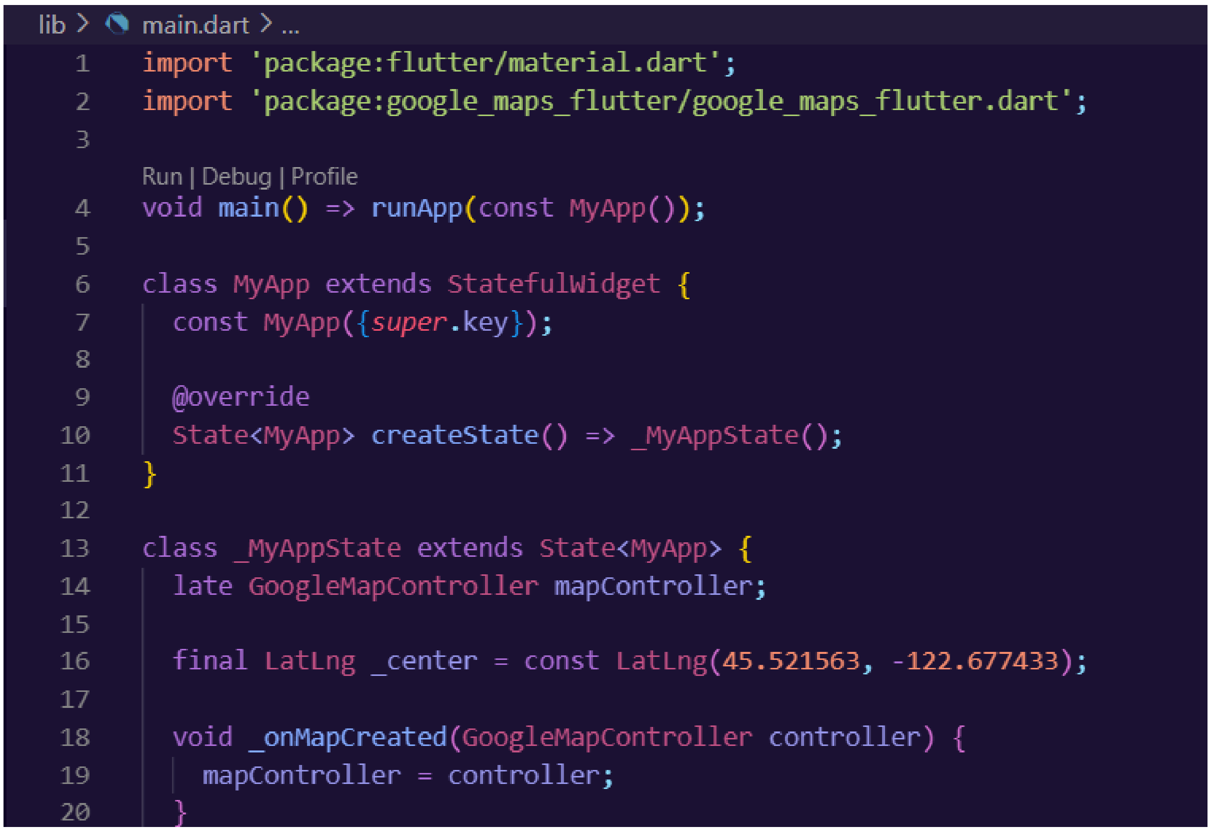


web > index.html > html > head > script

```html

  <!-- iOS meta tags & icons -->
  <meta name="apple-mobile-web-app-capable" content="yes">
  <meta name="apple-mobile-web-app-status-bar-style" content="black">
  <meta name="apple-mobile-web-app-title" content="google_maps_in_flutter">
  <link rel="apple-touch-icon" href="icons/Icon-192.png">

  <!-- Favicon -->
  <link rel="icon" type="image/png" href="favicon.png"/>

  <script src="https://maps.googleapis.com/maps/api/js?key=YOUR-KEY-HERE"></script>

  <title>google_maps_in_flutter</title>
  <link rel="manifest" href="manifest.json">
```

Putting a map on the screen

Now it's time to get a map on the screen. Update lib/main.dart as follows.



lib > main.dart > ...

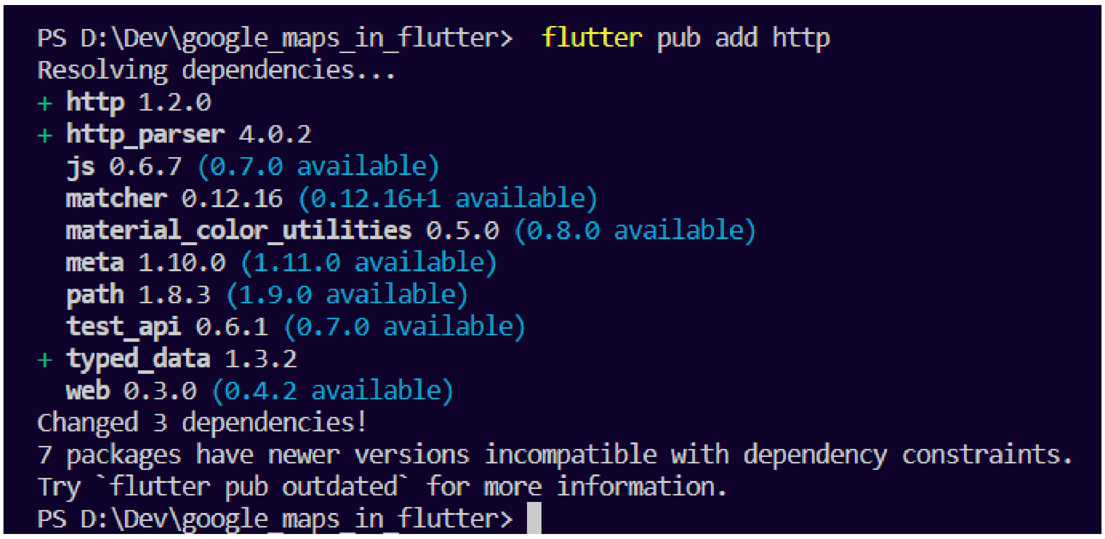
```dart
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';

Run | Debug | Profile
void main() => runApp(const MyApp());

class MyApp extends StatefulWidget {
  const MyApp({super.key});

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  late GoogleMapController mapController;

  final LatLng _center = const LatLng(45.521563, -122.677433);

  void _onMapCreated(GoogleMapController controller) {
    mapController = controller;
  }
```

Put Google on the Map

Google has many offices around the world, from North America, Latin America, Europe, Asia Pacific, to Africa & Middle East. The nice thing about these maps is that they have an easily usable API endpoint for supplying the office location information in JSON format.

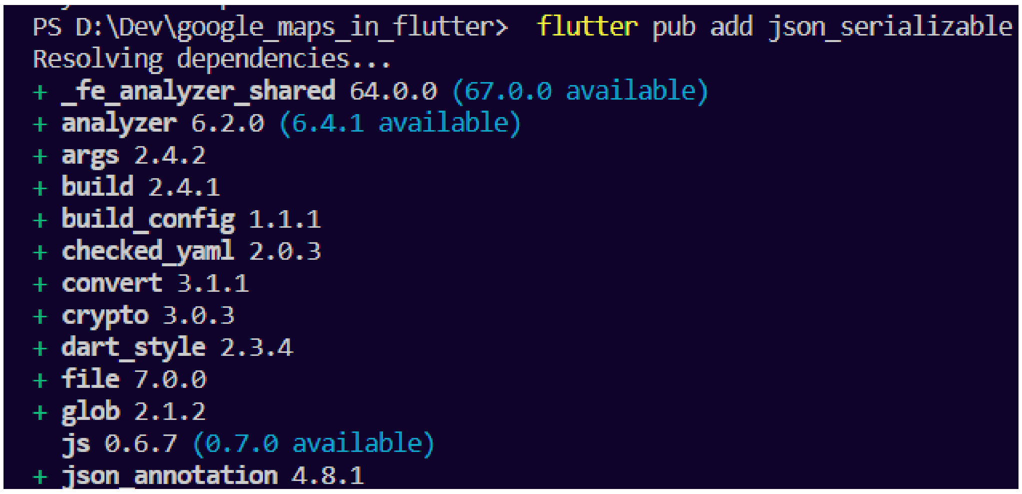
In this step, we will put these office locations on the map. To do so, we will use code generation to parse JSON.

Add three new Flutter dependencies to the project as follows. Firstly, add the http package for making HTTP requests easily.



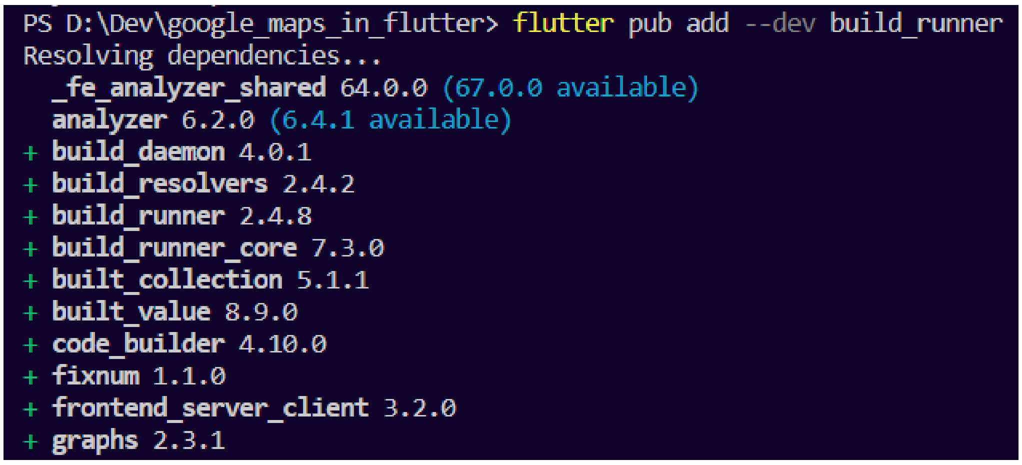
```
PS D:\Dev\google_maps_in_flutter>  flutter pub add http
Resolving dependencies...
+ http 1.2.0
+ http_parser 4.0.2
  js 0.6.7 (0.7.0 available)
  matcher 0.12.16 (0.12.16+1 available)
  material_color_utilities 0.5.0 (0.8.0 available)
  meta 1.10.0 (1.11.0 available)
  path 1.8.3 (1.9.0 available)
  test_api 0.6.1 (0.7.0 available)
+ typed_data 1.3.2
  web 0.3.0 (0.4.2 available)
Changed 3 dependencies!
7 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
PS D:\Dev\google_maps_in_flutter>
```

Next, add json_serializable and json_annotation for declaring object structure for representing JSON documents.



```
PS D:\Dev\google_maps_in_flutter>  flutter pub add json_serializable
Resolving dependencies...
+ _fe_analyzer_shared 64.0.0 (67.0.0 available)
+ analyzer 6.2.0 (6.4.1 available)
+ args 2.4.2
+ build 2.4.1
+ build_config 1.1.1
+ checked_yaml 2.0.3
+ convert 3.1.1
+ crypto 3.0.3
+ dart_style 2.3.4
+ file 7.0.0
+ glob 2.1.2
  js 0.6.7 (0.7.0 available)
+ json_annotation 4.8.1
```

Finally, add build_runner as a development time dependency. This will be used for code generation later in this step.

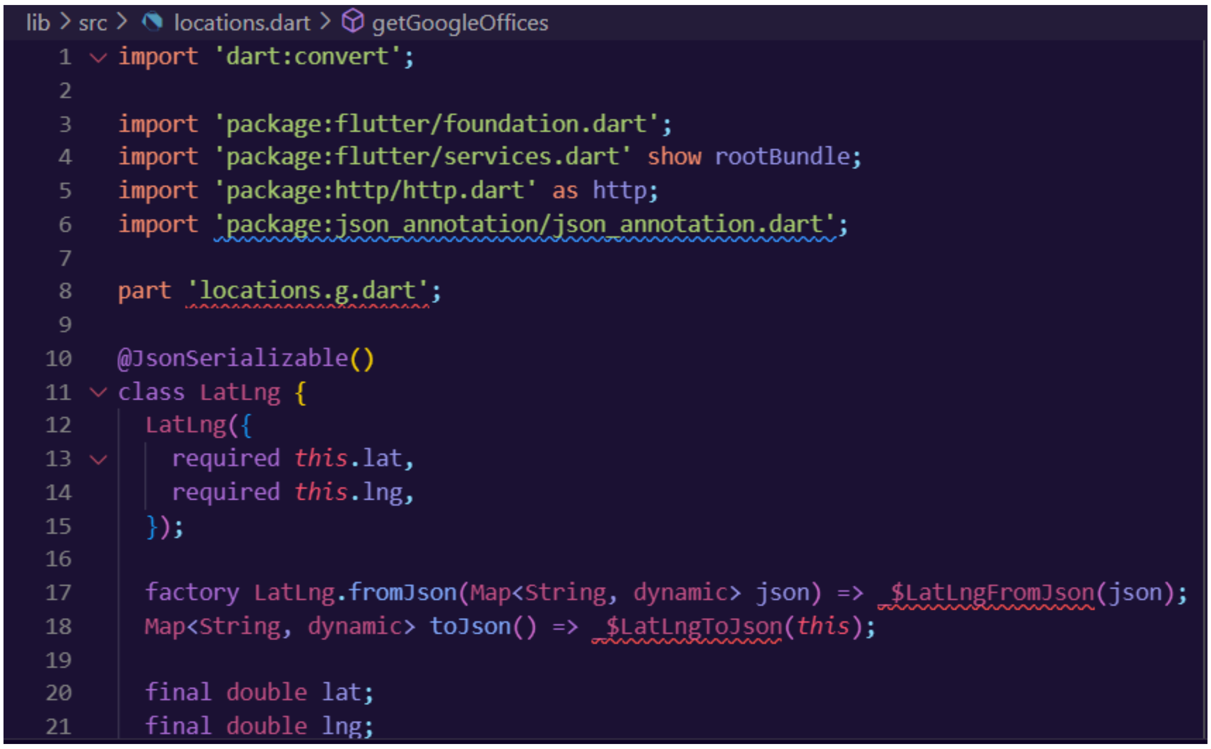


```
PS D:\Dev\google_maps_in_flutter> flutter pub add --dev build_runner
Resolving dependencies...
  _fe_analyzer_shared 64.0.0 (67.0.0 available)
  analyzer 6.2.0 (6.4.1 available)
+ build_daemon 4.0.1
+ build_resolvers 2.4.2
+ build_runner 2.4.8
+ build_runner_core 7.3.0
+ built_collection 5.1.1
+ built_value 8.9.0
+ code_builder 4.10.0
+ fixnum 1.1.0
+ frontend_server_client 3.2.0
+ graphs 2.3.1
```

Parsing JSON with code generation

The JSON data returned from the API endpoint has a regular structure. It would be handy to generate the code to marshal that data into objects that we can use in code.

In the *lib/src* directory, create a **locations.dart** file and describe the structure of the returned JSON data as follows:

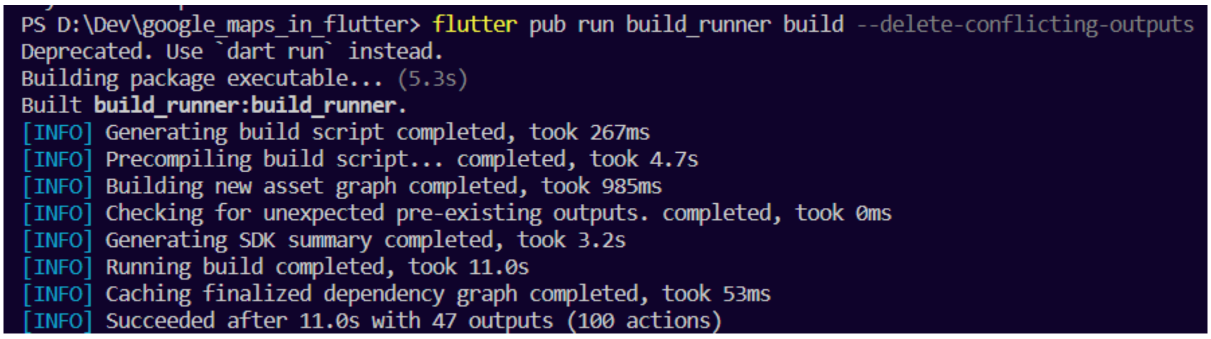


lib > src > locations.dart > getGoogleOffices

```dart
import 'dart:convert';

import 'package:flutter/foundation.dart';
import 'package:flutter/services.dart' show rootBundle;
import 'package:http/http.dart' as http;
import 'package:json_annotation/json_annotation.dart';

part 'locations.g.dart';

@JsonSerializable()
class LatLng {
  LatLng({
    required this.lat,
    required this.lng,
  });

  factory LatLng.fromJson(Map<String, dynamic> json) => _$LatLngFromJson(json);
  Map<String, dynamic> toJson() => _$LatLngToJson(this);

  final double lat;
  final double lng;
```

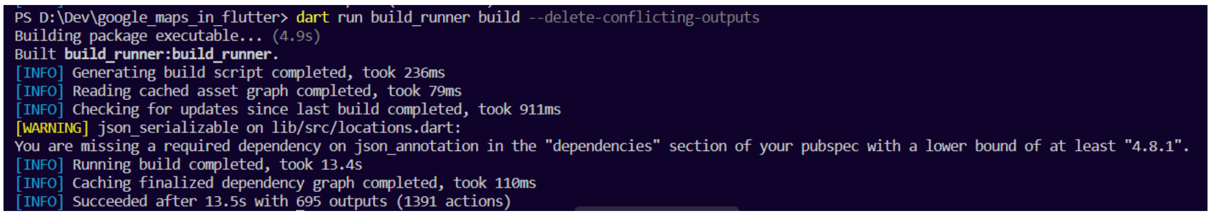
Once this code added, VSCode displays some red squiggles, as it references a nonexistent sibling file, **locations.g.dart**. This generated file converts between untyped JSON structures and named objects.

Create it by running the build_runner:



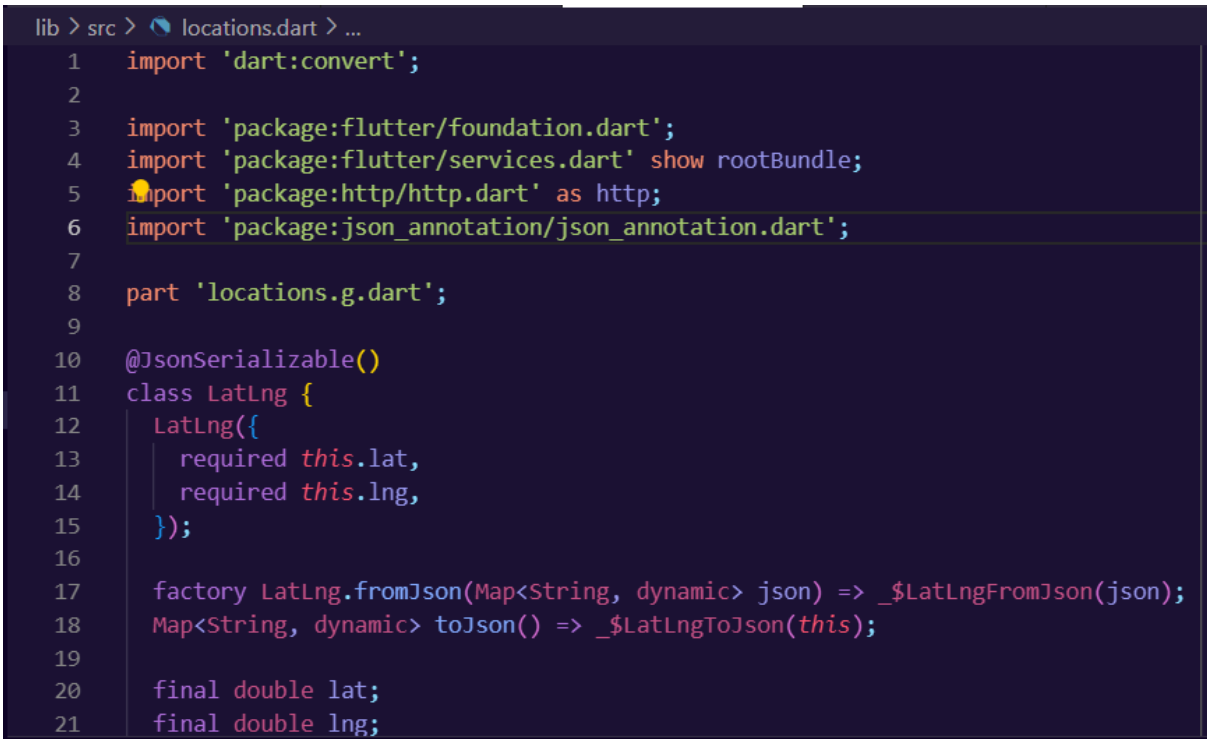
```
PS D:\Dev\google_maps_in_flutter> flutter pub run build_runner build --delete-conflicting-outputs
Deprecated. Use `dart run` instead.
Building package executable... (5.3s)
Built build_runner:build_runner.
[INFO] Generating build script completed, took 267ms
[INFO] Precompiling build script... completed, took 4.7s
[INFO] Building new asset graph completed, took 985ms
[INFO] Checking for unexpected pre-existing outputs. completed, took 0ms
[INFO] Generating SDK summary completed, took 3.2s
[INFO] Running build completed, took 11.0s
[INFO] Caching finalized dependency graph completed, took 53ms
[INFO] Succeeded after 11.0s with 47 outputs (100 actions)
```

Since the creation of the codelab until now, updates where made so we needed to use the new command as follow:



```
PS D:\Dev\google_maps_in_flutter> dart run build_runner build --delete-conflicting-outputs
Building package executable... (4.9s)
Built build_runner:build_runner.
[INFO] Generating build script completed, took 236ms
[INFO] Reading cached asset graph completed, took 79ms
[INFO] Checking for updates since last build completed, took 911ms
[WARNING] json_serializable on lib/src/locations.dart:
You are missing a required dependency on json_annotation in the "dependencies" section of your pubspec with a lower bound of at least "4.8.1".
[INFO] Running build completed, took 13.4s
[INFO] Caching finalized dependency graph completed, took 110ms
[INFO] Succeeded after 13.5s with 695 outputs (1391 actions)
```

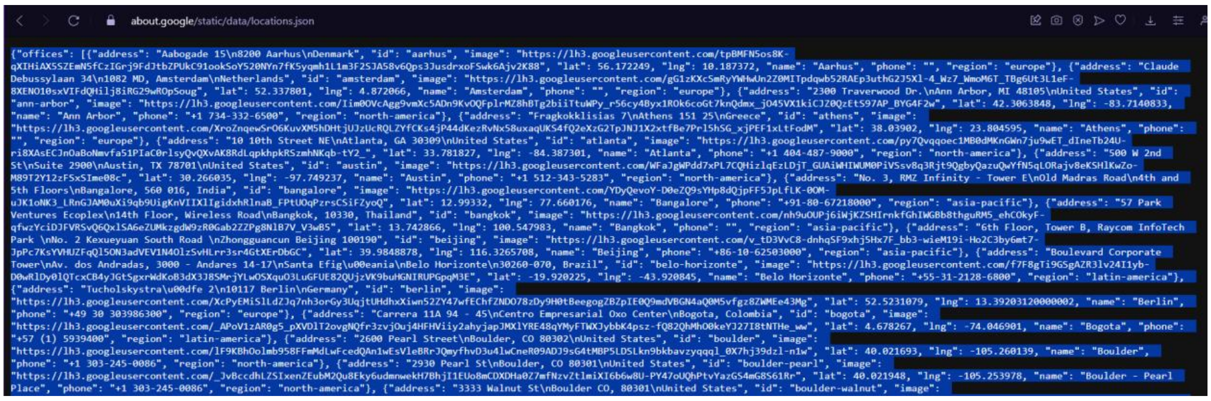
Now, the code should now analyze cleanly again:



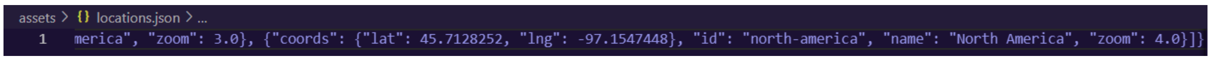
```dart
import 'dart:convert';

import 'package:flutter/foundation.dart';
import 'package:flutter/services.dart' show rootBundle;
import 'package:http/http.dart' as http;
import 'package:json_annotation/json_annotation.dart';

part 'locations.g.dart';

@JsonSerializable()
class LatLng {
  LatLng({
    required this.lat,
    required this.lng,
  });

  factory LatLng.fromJson(Map<String, dynamic> json) => _$LatLngFromJson(json);
  Map<String, dynamic> toJson() => _$LatLngToJson(this);

  final double lat;
  final double lng;
```

Next, we should add in the fallback locations.json file used in the getGoogleOffices function. One of the reasons for including this fallback is that the static data being loaded in this function is served without CORS headers, and thus will fail to load in a web browser. The Android and iOS Flutter apps don't need CORS headers, but mobile data access can be finicky at the best of times.

Navigate to *https://about.google/static/data/locations.json* in a browser, and save the contents into the asset directory.



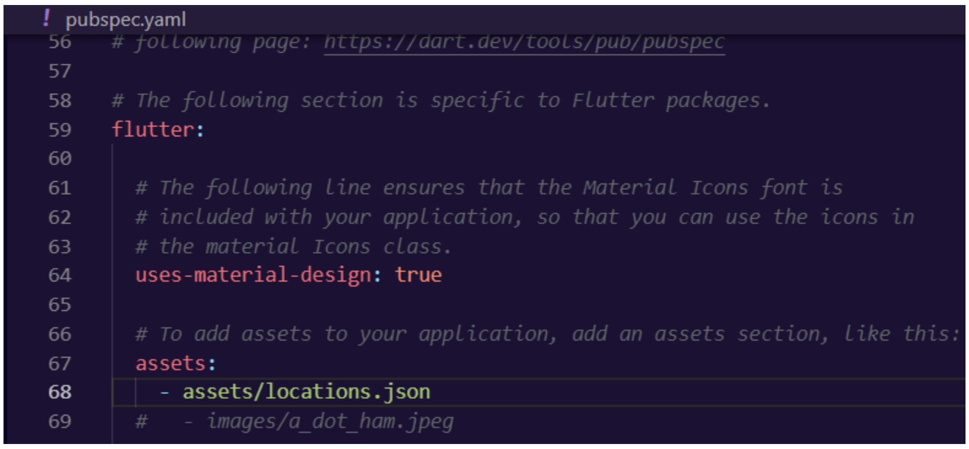
```
{"offices": [{"address": "Aabogade 15\n8200 Aarhus\nDenmark", "id": "aarhus", "image": "https://lh3.googleusercontent.com/tpBMFN5os8K-
qXIHiAX5SZEmN5fCzIGrj9FdJtbZPUkC91ookSoY520NYn7fK5yqmh1L1m3F2SJA58v6Qps3JusdrxoFSwk6Ajv2K88", "lat": 56.172249, "lng": 10.187372, "name": "Aarhus", "phone": "", "region": "europe"}, {"address": "Claude
Debussylaan 34\n1082 MD, Amsterdam\nNetherlands", "id": "amsterdam", "image": "https://lh3.googleusercontent.com/gG1zKXcSmRyYWHwUn2Z0MITpdqwb52RAEp3uthG2J5Xl-4_Wz7_WmoM6T_TBg6Ut3L1eF-
8XENO10sxVIFdQHilj8iRG29wROpSoug", "lat": 52.337801, "lng": 4.872066, "name": "Amsterdam", "phone": "", "region": "europe"}, {"address": "2300 Traverwood Dr.\nAnn Arbor, MI 48105\nUnited States", "id":
"ann-arbor", "image": "https://lh3.googleusercontent.com/Iim0OVcAgg9vmXc5ADn9KvOQFplrMZ8hBTg2biiTtuWPy_r56cy4Byx1ROk6coGt7knQdmx_jO45VX1kiCJZ0QzEtS97AP_BYG4F2w", "lat": 42.3063848, "lng": -83.7140833,
"name": "Ann Arbor", "phone": "+1 734-332-6500", "region": "north-america"}, {"address": "Fragkokklisias 7\nAthens 151 25\nGreece", "id": "athens", "image":
"https://lh3.googleusercontent.com/XroZnqewSrO6KuvXM5hDHtjUJzUcRQLZYfCKs4jP44dKezRvNx58uxaqUKS4fQ2eXzG2TpJNJ1X2xtfBe7Prl5hSG_xjPEF1xLtFodM", "lat": 38.03902, "lng": 23.804595, "name": "Athens", "phone":
"", "region": "europe"}, {"address": "10 10th Street NE\nAtlanta, GA 30309\nUnited States", "id": "atlanta", "image": "https://lh3.googleusercontent.com/py7Qvqqoec1MB0dMKnGWn7ju9wET_dIneTb24U-
ri8XAsECJnOaBoNmvfa51PIaC0rlsyQvQXvAK8RdLqpkhpkRSzmhNKqb-tY2_", "lat": 33.781827, "lng": -84.387301, "name": "Atlanta", "phone": "+1 404-487-9000", "region": "north-america"}, {"address": "500 W 2nd
St\nSuite 2900\nAustin, TX 78701\nUnited States", "id": "austin", "image": "https://lh3.googleusercontent.com/WFaJgWPdd7xPL7CQHizlqEzLDjT_GUAiWHIWUM0PiVSsv8q3Rjt9QgbyQazuQwYfN5qLORajv8eKSHlKwZo-
M89T2Y12zFSxSIme08c", "lat": 30.266035, "lng": -97.749237, "name": "Austin", "phone": "+1 512-343-5283", "region": "north-america"}, {"address": "No. 3, RMZ Infinity - Tower E\nOld Madras Road\n4th and
5th Floors\nBangalore, 560 016, India", "id": "bangalore", "image": "https://lh3.googleusercontent.com/YDyQevoY-D0eZQ9sYHp8dQjpFF5JpLfLK-0OM-
uJK1oNK3_LRnGJAM0uXi9qb9UigKnVIIXlIgidxhRlnaB_FPtUOqPzrsCSiFZyoQ", "lat": 12.99332, "lng": 77.660176, "name": "Bangalore", "phone": "+91-80-67218000", "region": "asia-pacific"}, {"address": "57 Park
Ventures Ecoplex\n14th Floor, Wireless Road\nBangkok, 10330, Thailand", "id": "bangkok", "image": "https://lh3.googleusercontent.com/nh9uOUPj6iWjKZSHIrnkfGhIWGBb8thguRM5_ehCOkyF-
qfwzYciDJFVRSvQ6Qxl5A6eZUMkzgdW9zR0Gab2ZZPg8NlB7V_V3wB5", "lat": 13.742866, "lng": 100.547983, "name": "Bangkok", "phone": "", "region": "asia-pacific"}, {"address": "6th Floor, Tower B, Raycom InfoTech
Park \nNo. 2 Kexueyuan South Road \nZhongguancun Beijing 100190", "id": "beijing", "image": "https://lh3.googleusercontent.com/v_tD3VvC8-dnhqSF9xhj5Hx7F_bb3-wieM19i-Ho2C3by6mt7-
JpPc7KsYVHUZFqQl5ON3adVEV1N4OlzSvHLrr3sr4GtXErDbGC", "lat": 39.9848878, "lng": 116.3265708, "name": "Beijing", "phone": "+86-10-62503000", "region": "asia-pacific"}, {"address": "Boulevard Corporate
Tower\nAv. dos Andradas, 3000 - Andares 14-17\nSanta Efig\u00eania\nBelo Horizonte\n30260-070, Brazil", "id": "belo-horizonte", "image": "https://lh3.googleusercontent.com/f7F8gTi9GSgAZR3lv24I1yb-
D0wRlDy0lQTcxCB4yJGtSgxrWdKoB3dX3J8SMrjYLwOSXquO3LuGFUE82QUjzVK9buHGNIRUPGpqM3E", "lat": -19.920225, "lng": -43.920845, "name": "Belo Horizonte", "phone": "+55-31-2128-6800", "region": "latin-america"},
{"address": "Tucholskystra\u00dfe 2\n10117 Berlin\nGermany", "id": "berlin", "image":
"https://lh3.googleusercontent.com/XcPyEMiSlLdZJq7nh3orGy3UqjtUHdhxXiwn52ZY47wfEChfZNDO78zDy9H0tBeegogZBZpIE0Q9mdVBGN4aQ0M5vfgz8ZWMEe43Mg", "lat": 52.5231079, "lng": 13.39203120000002, "name": "Berlin",
"phone": "+49 30 303986300", "region": "europe"}, {"address": "Carrera 11A 94 - 45\nCentro Empresarial Oxo Center\nBogota, Colombia", "id": "bogota", "image":
"https://lh3.googleusercontent.com/_APoV1zAR0g5_pXVDlT2ovgNQfr3zvjOuj4HFHViiy2ahyjapJMXlYRE48qYMyFTWXJybbK4psz-fQ82QhMhO0keYJ27I8tNTHe_ww", "lat": 4.678267, "lng": -74.046901, "name": "Bogota", "phone":
"+57 (1) 5939400", "region": "latin-america"}, {"address": "2600 Pearl Street\nBoulder, CO 80302\nUnited States", "id": "boulder", "image":
"https://lh3.googleusercontent.com/lF9KBhOolmb958FFmMdLwFcedQAn1wEsVleBRrJQmyfhvD3u4lwCneR09ADJ9sG4tMBP5LDSLkn9bkbavzyqqql_0X7hj39dzl-n1w", "lat": 40.021693, "lng": -105.260139, "name": "Boulder",
"phone": "+1 303-245-0086", "region": "north-america"}, {"address": "2930 Pearl St\nBoulder, CO 80301\nUnited States", "id": "boulder-pearl", "image":
"https://lh3.googleusercontent.com/_JvBccdhLZSIxenZEubM2Qu8Eky6udmnwekH7BhjI1EUo8mCDXDHa0Z7mfNzvZtlmiXI6b6w8U-PY47oUQhPtvYazGS4mG8S61Rr", "lat": 40.021948, "lng": -105.253978, "name": "Boulder - Pearl
Place", "phone": "+1 303-245-0086", "region": "north-america"}, {"address": "3333 Walnut St\nBoulder CO, 80301\nUnited States", "id": "boulder-walnut", "image":
```



```
merica", "zoom": 3.0}, {"coords": {"lat": 45.7128252, "lng": -97.1547448}, "id": "north-america", "name": "North America", "zoom": 4.0}]}
```

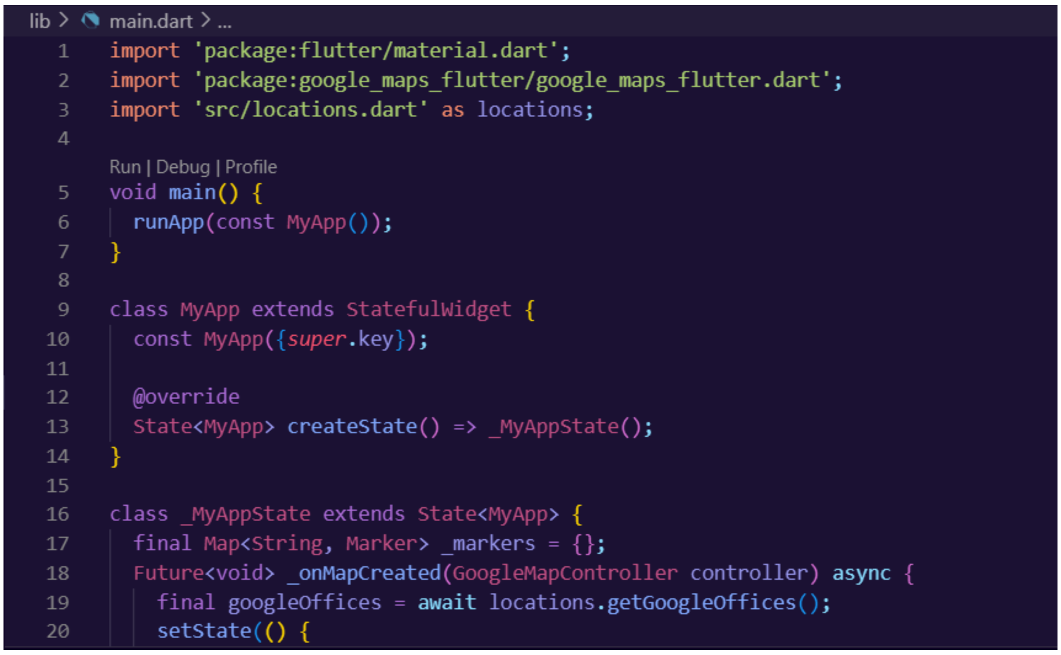
We needed to create the assets directory before pasting the data.

Now that we have the asset file downloaded, we can add it to the flutter section of the **pubspec.yaml** file.



```yaml
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  assets:
    - assets/locations.json
  #   - images/a_dot_ham.jpeg
```

Modify the main.dart file to request the map data, and then use the returned info to add offices to the map:



```dart
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'src/locations.dart' as locations;

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatefulWidget {
  const MyApp({super.key});

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  final Map<String, Marker> _markers = {};
  Future<void> _onMapCreated(GoogleMapController controller) async {
    final googleOffices = await locations.getGoogleOffices();
    setState(() {
```

This code performs several operations:

- In _onMapCreated, it uses the JSON parsing code from the previous step, awaiting until it's loaded. It then uses the returned data to create Markers inside a setState() callback. Once the app receives new markers, setState flags Flutter to repaint the screen, causing the office locations to display.
- The markers are stored in a Map that is associated with the GoogleMap widget. This links the markers to the correct map. You could, of course, have multiple maps and display different markers in each.

Conclusion

This task introduced us to the implementation of Google Map in a Flutter app. The steps are quite simple and easy to apply and this code is totally reusable for our future product.
The created code works perfectly with no errors. However, we did not test in an emulator because we were not able to get the APIs from Google Maps Platform as explained earlier. If ever we create a real project and get an API from an account with valid billing information, this code will perform his tasks without no errors: The app will display a map with all the locations of Google offices.