

Programmazione Mobile

Nicola Noviello

nicola.noviello@unimol.it

Corso di Laurea in Informatica
Dipartimento di Bioscienze e Territorio
Università degli Studi del Molise
Anno 2023/2024

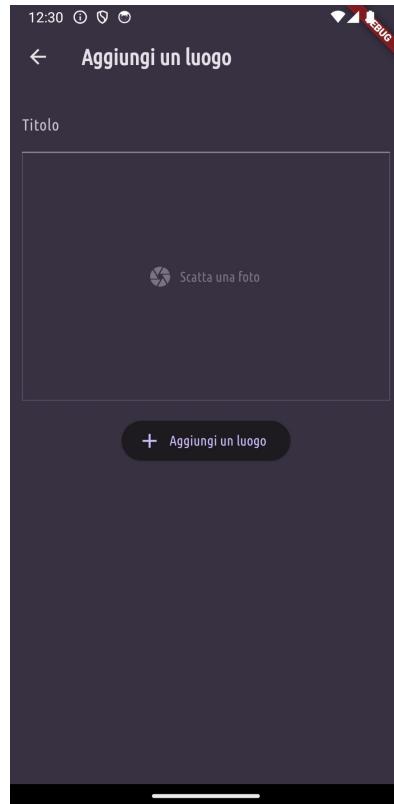
Lezione: Utilizzo delle feature native dei device mobile

- Cenni alla persistenza degli stati
- Utilizzo della camera
- Location e Google Maps



Dove eravamo

Dove eravamo



image_picker - https://pub.dev/packages/image_picker

The screenshot shows the pub.dev package page for the `image_picker` plugin. The page title is `image_picker 1.1.1`. It was published 15 days ago and is Dart 3 compatible. The package is available for `SDK`, `FLUTTER`, `PLATFORM`, `ANDROID`, `IOS`, `LINUX`, `MACOS`, `WEB`, and `WINDOWS`. It has 6.5K likes. The package has 6583 likes, 140 pub points, and 100% popularity. The publisher is `@flutter.dev`. The package is a Flutter plugin for selecting images from the Android and iOS image library and taking new pictures with the camera. It supports `Android`, `iOS`, `Linux`, `macOS`, `Web`, and `Windows`. Support is provided for `SDK 21+`, `iOS 12+`, `Any`, `10.14+`, `See image_picker_for_web`, and `Windows 10+`. The `Installation` section instructs users to add `image_picker` as a dependency in their `pubspec.yaml` file. For `iOS`, it notes that version `0.8.1` uses `PHPicker` to pick multiple images on iOS 14 or higher, which prevents picking HEIC images on the simulator. A known issue is mentioned where Apple solves it. The `Info.plist` file needs keys for photo library usage. The `Topics` listed are `#camera #image-picker #files #file-selection`. The `Documentation` and `API reference` sections are available, along with the `License` information under the Apache-2.0, BSD-3-Clause (`LICENSE`) clause.

image_picker - https://pub.dev/packages/image_picker

Attenzione! Questo è un package “delicato”, cambia di frequente e alcune funzionalità possono cambiare nel tempo, quindi se viene usato in un’App in produzione va controllato e nel caso aggiornato, adeguandone l’implementazione

The screenshot shows a browser window displaying the [image_picker](https://pub.dev/packages/image_picker) package page on pub.dev. The URL in the address bar is `pub.dev/packages/image_picker`. The page content includes:

- A note: "First, add `image_picker` as a dependency in your `pubspec.yaml` file."
- A section for iOS: "Starting with version 0.8.1 the iOS implementation uses PHPicker to pick (multiple) images on iOS 14 or higher. As a result of implementing PHPicker it becomes impossible to pick HEIC images on the iOS simulator in iOS 14+. This is a known issue. Please test this on a real device, or test with non-HEIC images until Apple solves this issue. [63426347 - Apple known issue](#)"
- A section for Android: "Starting with version 0.8.1 the Android implementation support to pick (multiple) images on Android 4.3 or higher."

Integrazione di ImagePicker()

```
class _ImageInputState extends State<ImageInput> {
  File? _selectedImage;

  void _takePicture() async {
    final imagePicker = ImagePicker();
    final pickedImage =
        await imagePicker.pickImage(source: ImageSource.camera, maxWidth: 600);

    if (pickedImage == null) {
      return;
    }

    setState(() {
      _selectedImage = File(pickedImage.path);
    });
  }
}
```

Integrazione di ImagePicker()

- Viene creato un nuovo oggetto **ImagePicker**. **ImagePicker** è un plugin che consente di accedere alla fotocamera o alla galleria del dispositivo;
- Viene chiamato il metodo **pickImage** sull'oggetto **ImagePicker**. Questo metodo apre la fotocamera del dispositivo e permette all'utente di scattare una foto. Il parametro source è impostato su **ImageSource.camera**, il che significa che la foto verrà scattata con la fotocamera e non acquisita dalla galleria. Il parametro maxWidth è impostato su 600, il che significa che la larghezza massima dell'immagine sarà di 600 pixel, è importante evitare immagini troppo grandi se non è necessario;
- Controlla se **pickedImage** è null. Se è null, la funzione termina con un return. Questo può accadere se l'utente annulla l'operazione di scatto della foto o in caso di problemi;
- Se **pickedImage** non è null, chiama il metodo **setState**. Questo metodo è usato per aggiornare lo stato dell'applicazione. All'interno di setState, `_selectedImage` viene impostato come un nuovo oggetto File creato dal percorso dell'immagine scattata (`pickedImage.path`). `pickedImage` è un XFile, quindi è necessario fare un cast.

Integrazione di ImagePicker()

```
@override
Widget build(BuildContext context) {
    Widget content = TextButton.icon(
        icon: const Icon(Icons.camera),
        label: const Text('Scatta una foto'),
        onPressed: _takePicture,
    ); // TextButton.icon

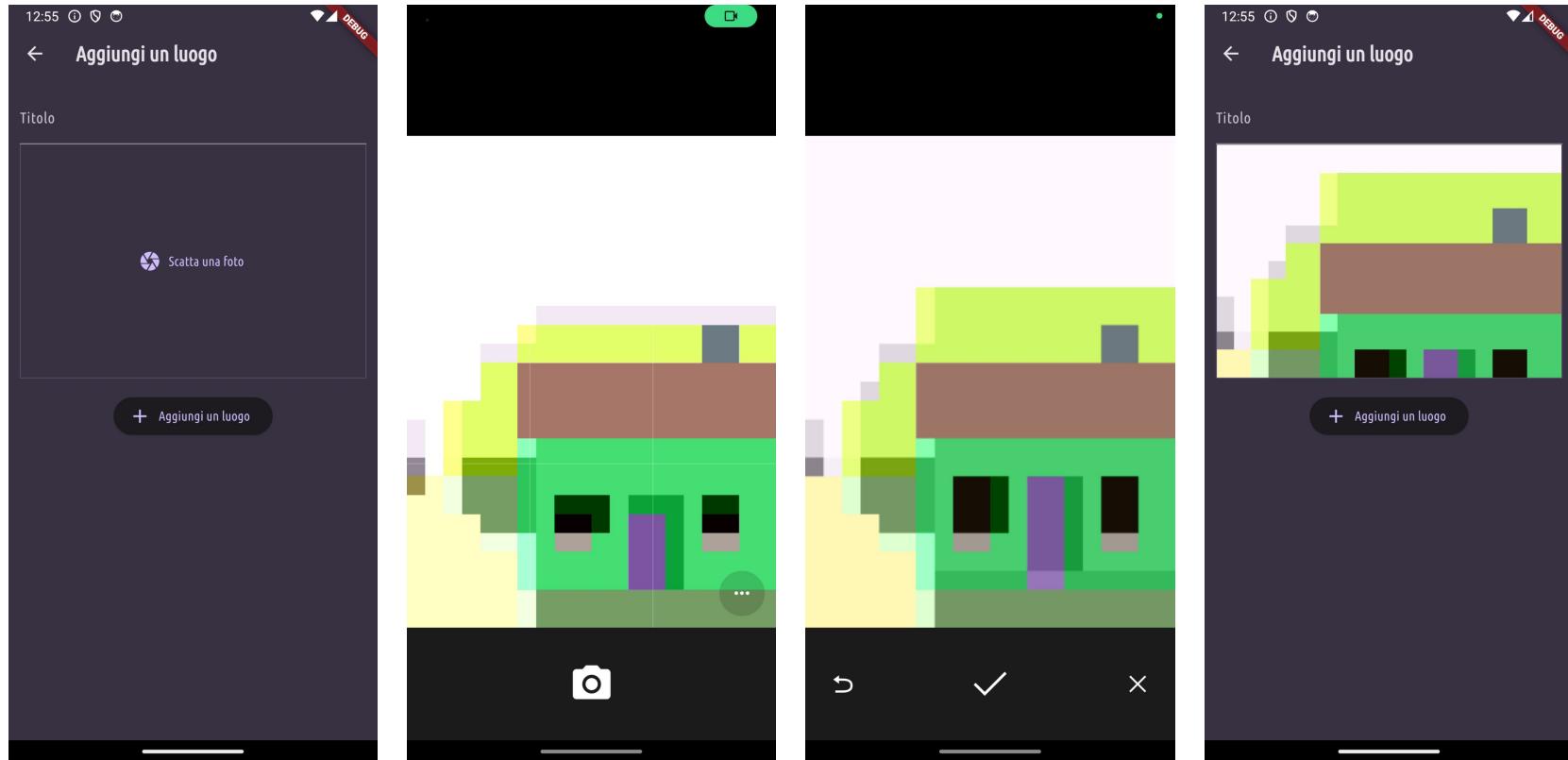
    if (_selectedImage != null) {
        content = Image.file(
            _selectedImage!,
            fit: BoxFit.cover,
            width: double.infinity,
            height: double.infinity,
        );
    }
}

return Container(
    decoration: BoxDecoration(
        border: Border.all(
```

Integrazione di ImagePicker()

Controlliamo se la variabile `_selectedImage` non è nulla. Se non è nulla, assegna a `content` un nuovo oggetto `Image.file`. Questo oggetto prende `_selectedImage` come file di input, imposta il suo fit secondo i parametri `BoxFit.cover` (che significa che l'immagine coprirà tutto lo spazio disponibile *mantenendo il suo aspect ratio*), e imposta la sua larghezza e altezza a `double.infinity`, che significa che l'immagine occuperà tutto lo spazio disponibile.

Il risultato



Il risultato



GestureDetector

```
if (_selectedImage != null) {
    content = GestureDetector(
        onTap: _takePicture,
        child: Image.file(
            _selectedImage!,
            fit: BoxFit.cover,
            width: double.infinity,
            height: double.infinity,
        ), // Image.file
    ); // GestureDetector
}
```

GestureDetector

Abbiamo fatto un wrap del widget con un widget **GestureDetector**. Questo widget viene utilizzato per rilevare diversi tipi di gesti. In questo caso, viene utilizzato per rilevare un gesto di tocco. Quando l'utente tocca il widget, viene chiamata la funzione **_takePicture**.

Aggiornamento del data model

Aggiornamento della classe Place

```
lib > models > place.dart > ...
1 import 'dart:io';
2
3 import 'package:uuid/uuid.dart';
4
5 const uuid = Uuid();
6
7 class Place {
8     Place({required this.title, required this.image}) : id = uuid.v4();
9
10    final String id;
11    final String title;
12    final File image;
13 }
14
```

Aggiornamento del provider UserPlacesNotifier

```
lib > providers > user_places.dart > ...
1  ↘ import 'package:flutter_riverpod/flutter_riverpod.dart';
2  import 'dart:io';
3
4
5  import 'package:devicenative/models/place.dart';
6
7  ↘ class UserPlacesNotifier extends StateNotifier<List<Place>> {
8      UserPlacesNotifier() : super(const []);
9
10 ↘ void addPlace(String title, File image) {           ←
11    // questo potete scriverlo come volete, è a vostra discrezione
12    // potete passare anche un oggetto Place direttamente
13    final newPlace = Place(title: title, image: image); ←
14    // aggiorna lo stato con il nuovo luogo e il resto della lista "spalmata"
15    state = [newPlace, ...state];
16  }
17}
18
19 final userPlacesProvider =
20 ↘   StateNotifierProvider<UserPlacesNotifier, List<Place>>(
21     (ref) => UserPlacesNotifier(),
22   ); // StateNotifierProvider
```

Aggiornamento di ImageInput

```
lib > widgets > image_input.dart > _ImageInputState > build
  1 import 'package:flutter/material.dart';
  2 import 'package:image_picker/image_picker.dart';
  3 import 'dart:io';
  4
  5 class ImageInput extends StatefulWidget {
  6   const ImageInput({super.key, required this.onPickImage});
  7
  8   final void Function(File image) onPickImage;
  9
 10  @override
 11  State<ImageInput> createState() => _ImageInputState();
 12 }
 13
 14 class _ImageInputState extends State<ImageInput> {
 15   File? _selectedImage;
 16
 17   void _takePicture() async {
 18     final imagePicker = ImagePicker();
 19     final pickedImage =
 20       await imagePicker.pickImage(source: ImageSource.camera, maxWidth: 600);
 21
 22     if (pickedImage == null) {
 23       return;
 24     }
 25
 26     setState(() {
 27       _selectedImage = File(pickedImage.path);
 28     });
 29     widget.onPickImage(_selectedImage!);
 30   }
 31 }
```

Aggiornamento di ImageInput: Perché usare il “lifting state up”?

```
lib > widgets > image_input.dart > _ImageInputState > build
1 import 'package:flutter/material.dart';
2 import 'package:image_picker/image_picker.dart';
3 import 'dart:io';
4
5 class ImageInput extends StatefulWidget {
6   const ImageInput({super.key, required this.onPickImage});
7
8   final void Function(File image) onPickImage;
9
10 @override
11 State<ImageInput> createState() => _ImageInputState();
12 }
13
14 class _ImageInputState extends State<ImageInput> {
15   File? _selectedImage;
16
17   void _takePicture() async {
18     final imagePicker = ImagePicker();
19     final pickedImage =
20       await imagePicker.pickImage(source: ImageSource.camera, maxWidth: 600);
21
22     if (pickedImage == null) {
23       return;
24     }
25
26     setState(() {
27       _selectedImage = File(pickedImage.path);
28     });
29     widget.onPickImage(_selectedImage!);
30   }
31 }
```

Aggiornamento di ImageInput: Perché usare il “lifting state up”?

```
lib > widgets > image_input.dart > _ImageInputState > build
1 import 'package:flutter/material.dart';
2 import 'package:image_picker/image_picker.dart';
3 import 'dart:io';
4
5 class ImageInput extends StatefulWidget {
6   const ImageInput({super.key, required this.onPickImage});
7
8   final void Function(File image) onPickImage;
9
10  @override
11  State<ImageInput> createState() => _ImageInputState();
12 }
13
14 class _ImageInputState extends State<ImageInput> {
15   File? _selectedImage;
16
17   void _takePicture() async {
18     final imagePicker = ImagePicker();
19     final pickedImage =
20       await imagePicker.pickImage(source: ImageSource.camera, maxWidth: 600);
21
22     if (pickedImage == null) {
23       return;
24     }
25
26     setState(() {
27       _selectedImage = File(pickedImage.path);
28     });
29     widget.onPickImage(_selectedImage!);
30 }
```

Perché l'aggiornamento del Provider è gestito dal widget AddPlaceScreen e l'operazione di inserimento deve essere unica. Quindi è ammesso l'uso di formule ibride per raggiungere un risultato.

Aggiornamento dello screen AddPlaceScreen

```
class _AddPlaceScreenState extends ConsumerState<AddPlaceScreen> {
    final _titleController = TextEditingController();
    File? _selectedImage;
    void _savePlace() {
        final enteredTitle = _titleController.text;
        if (enteredTitle.isEmpty || _selectedImage == null) {
            return;
        }
        ref
            .read(userPlacesProvider.notifier)
            .addPlace(enteredTitle, _selectedImage!);
        Navigator.of(context).pop();
    }
}
```

Aggiornamento dello screen AddPlaceScreen

```
@override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: const Text('Aggiungi un luogo'),
        ), // AppBar
        body: SingleChildScrollView(
            padding: const EdgeInsets.all(12),
            child: Column(
                children: [
                    TextField(
                        decoration: const InputDecoration(labelText: 'Titolo'),
                        controller: _titleController,
                        style: TextStyle(
                            color: Theme.of(context).colorScheme.onSurface,
                        ), // TextStyle
                    ), // TextField
                    ImageInput(
                        onPickImage: (image) {
                            _selectedImage = image;
                        },
                    ), // ImageInput
                    const SizedBox(height: 16),
                ],
            ),
        ),
    );
}
```





Adeguiamo lista e
dettaglio per
mostrare
l'anteprima

Modifichiamo il widget PlacesList

```
3
4     return ListView.builder(
5         itemCount: places.length,
6         itemBuilder: (ctx, index) => ListTile(
7             leading: CircleAvatar(
8                 radius: 26,
9                 backgroundImage: FileImage(places[index].image),
0             ), // CircleAvatar
1             title: Text(
2                 places[index].title,
3                 style: Theme.of(context).textTheme.titleMedium!.copyWith(
4                     color: Theme.of(context).colorScheme.onSurface,
5                 ),
6             ), // Text
7             onTap: () {
8                 Navigator.of(context).push(
9                     MaterialPageRoute(
0                         builder: (ctx) => PlaceDetailScreen(place: places[index]),
1                     ), // MaterialPageRoute
2                 );
3             },

```

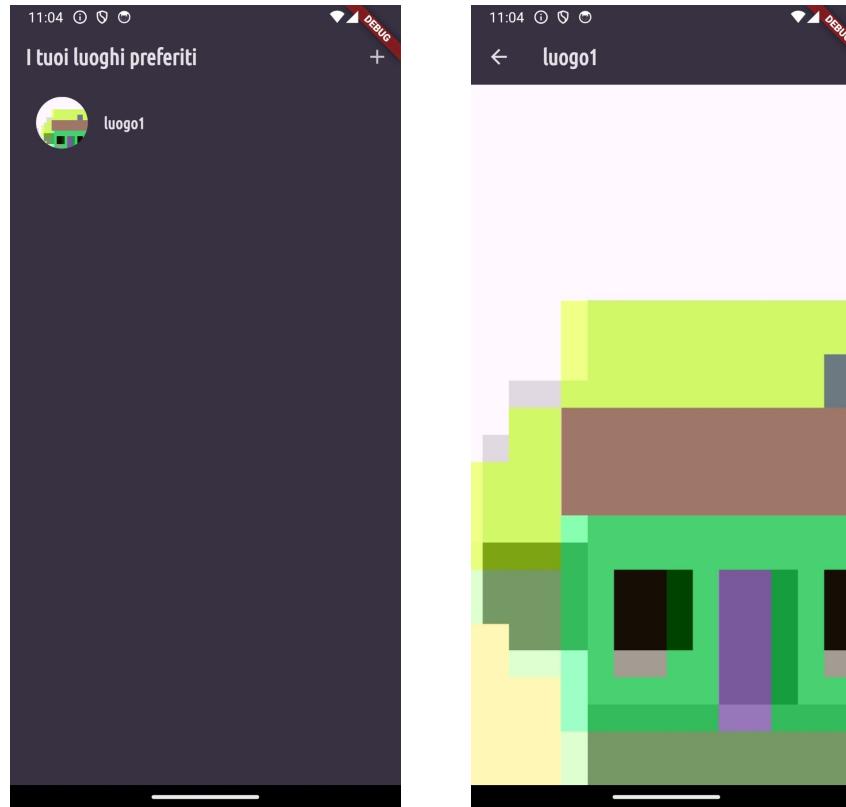


Aggiungiamo foto nel dettaglio su PlaceDetailScreen

```
lib > screens > place_detail.dart > ...
1  import 'package:flutter/material.dart';
2
3  import 'package:devicenative/models/place.dart';
4
5  class PlaceDetailScreen extends StatelessWidget {
6    const PlaceDetailScreen({super.key, required this.place});
7
8    final Place place;
9
10   @override
11   Widget build(BuildContext context) {
12     return Scaffold(
13       appBar: AppBar(
14         title: Text(place.title),
15       ), // AppBar
16       body: Stack(
17         children: [
18           Image.file(
19             place.image,
20             fit: BoxFit.cover,
21             width: double.infinity,
22             height: double.infinity,
23           ), // Image.file
24           ],
25         ), // Stack
26       // body: Center(
27       //   child: Text(
28       //     place.title,
29       //     style: Theme.of(context).textTheme.bodyLarge!.copyWith(
30       //       color: Theme.of(context).colorScheme.onSurface,
31       //     ),
32       //   ),
33       // ),
34     ); // Scaffold
}
```



Il risultato





Integriamo
l'accesso alla
posizione GPS

Come si integra l'accesso alla location dell'utente?

Lo abbiamo già visto!

Ma intanto...

Creiamo un nuovo widget: LocationInput

```
1 import 'package:flutter/material.dart';
2
3 class LocationInput extends StatefulWidget {
4   const LocationInput({super.key});
5
6   @override
7   State<LocationInput> createState() {
8     return _LocationInputState();
9   }
10 }
11
12 class _LocationInputState extends State<LocationInput> {
13   @override
14   Widget build(BuildContext context) {
15     return Column(
16       children: [
17         Container(
18           height: 170,
19           width: double.infinity,
20           alignment: Alignment.center,
21           decoration: BoxDecoration(
22             border: Border.all(
23               width: 1,
24               color: Theme.of(context).colorScheme.primary.withOpacity(0.2),
25             ), // Border.all
26           ), // BoxDecoration
27           child: Text(
28             'Nessuna location selezionata',
29             textAlign: TextAlign.center,
30             style: Theme.of(context).textTheme.bodyLarge!.copyWith(
31               color: Theme.of(context).colorScheme.onSurface,
32             ),
33           ), // Text
34         ), // Container
35         Row(
36           mainAxisAlignment: MainAxisAlignment.spaceEvenly,
37           children: [
38             TextButton.icon(
39               icon: const Icon(Icons.location_on),
40               label: const Text('Posizione dell\'utente'),
41               onPressed: () {},
42             ), // TextButton.icon
43             TextButton.icon(
44               icon: const Icon(Icons.map),
45               label: const Text('Seleziona dalla Mappa'),
46               onPressed: () {},
47             ), // TextButton.icon
48           ],
49         ), // Row
50       ],
51     ); // Column
52 }
```

Creiamo un nuovo widget: LocationInput

- Cominciamo a costruire un'interfaccia utente composta da una Column con due figli.
- Il primo figlio è un Container che ha una determinata altezza e larghezza, e un bordo. All'interno del Container c'è un Text widget che mostra il messaggio 'Nessuna location selezionata', da mostrare quando ancora l'utente non effettua la sua scelta.
- Il secondo figlio della Column è una Row con due TextButton.icon widgets. Questi sono bottoni con icone e testo. Il primo bottone ha un'icona di una posizione (Icons.location_on) e il testo 'Posizione dell'utente'. Il secondo bottone ha un'icona di una mappa (Icons.map) e il testo 'Seleziona dalla Mappa'. Al momento, premere questi bottoni non fa nulla, perché la funzione onPressed è vuota.
- Abbiamo progettato questo widget per permettere all'utente di scegliere una posizione da due input differenti senza aver ancora integrato le funzionalità.

Integriamo LocationInput nella schermata AddPlaceScreen

```
decoration: const InputDecoration(labelText: 'Titolo'),
controller: _titleController,
style: TextStyle(
|   color: Theme.of(context).colorScheme.onSurface,
),
// TextStyle
), // TextField
ImageInput(
|   onPickImage: (image) {
|     _selectedImage = image;
},
), // ImageInput
const SizedBox(height: 10),
LocationInput(),
const SizedBox(height: 10),
ElevatedButton.icon(
|   onPressed: _savePlace,
|   icon: const Icon(Icons.add),
|   label: const Text('Aggiungi un luogo'),
),
// ElevatedButton.icon

// Column
SingleChildScrollView
caffold
```



Integriamo l'accesso al GPS in LocationInput

```
class _LocationInputState extends State<LocationInput> {
  Location? _pickedLocation;
  var _isGettingLocation = false;

  void _getCurrentLocation() async {
    Location location = Location();

    bool serviceEnabled;
    PermissionStatus permissionGranted;
    LocationData locationData;

    serviceEnabled = await location.serviceEnabled();
    if (!serviceEnabled) {
      serviceEnabled = await location.requestService();
      if (!serviceEnabled) {
        return;
      }
    }

    permissionGranted = await location.hasPermission();
    if (permissionGranted == PermissionStatus.denied) {
      permissionGranted = await location.requestPermission();
      if (permissionGranted != PermissionStatus.granted) {
        return;
      }
    }
  }
}
```

Settiamo variabili e permessi

Integriamo l'accesso al GPS in LocationInput

```
    permissionGranted = await location.requestPermission();
    if (permissionGranted != PermissionStatus.granted) {
        return;
    }

    setState(() {
        _isGettingLocation = true;
    });

    locationData = await location.getLocation();

    setState(() {
        _isGettingLocation = false;
    });

    print(locationData.latitude);
    print(locationData.longitude);
}
```

Acquisiamo la
posizione

```
D/EGL_emulation(30876): app_time
D/EGL_emulation(30876): app_time
I/flutter (30876): 37.4219983
I/flutter (30876): -122.084
```

Integriamo l'accesso al GPS in LocationInput

```
Widget build(BuildContext context) {
  Widget previewContent = Text(
    'Nessuna location selezionata',
    textAlign: TextAlign.center,
    style: Theme.of(context).textTheme.bodyLarge!.copyWith(
      color: Theme.of(context).colorScheme.onSurface,
    ),
  );
  if (_isGettingLocation) {
    previewContent = const CircularProgressIndicator();
  }
  return Column(
    children: [
      Container(
        height: 170,
        width: double.infinity,
        alignment: Alignment.center,
        decoration: BoxDecoration(
          border: Border.all(
            width: 1,
            color: Theme.of(context).colorScheme.primary.withOpacity(0.2),
          ), // Border.all
        ), // BoxDecoration
        child: previewContent,
      ),
    ],
  );
}
```

Integriamo uno spinner che funziona mentre vengono acquisite le coordinate

<https://developers.google.com/maps?hl=it>

The screenshot shows the Google Maps Platform homepage in Italian. The URL in the address bar is `https://developers.google.com/maps?hl=it`. The page features a large heading "Crea fantastiche app grazie alla conoscenza di Google del mondo reale". Below it, a subtext reads "Crea esperienze reali e in tempo reale con le funzionalità più recenti di Maps, Routes e Places di Google Maps Platform. Creato dal team di Google per gli sviluppatori di tutto il mondo." There are two buttons: "inizia" (blue) and "Leggi la documentazione" (white). A section titled "Esplora gli argomenti principali" contains four cards: "Anteprima Google I/O '24" (with a thumbnail of a city skyline), "Guarda la nostra sessione tecnica" (with a thumbnail of a person speaking), "Segui il workshop" (with a thumbnail of two people), and "Componenti delle reazioni" (with a thumbnail of a map with reaction lines).

Billing

Get started on Google Maps Platform

You're all set to develop! Here's the API key that you would need for your implementation. API key can be referenced in the Credentials section.

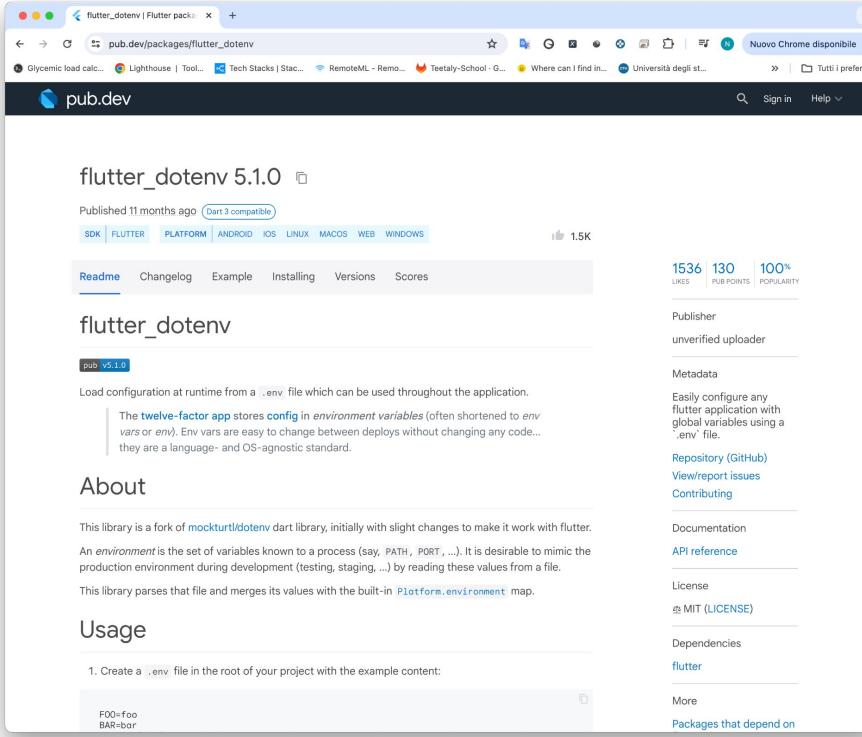
Your API key copy

Enable all Google Maps APIs for this project ?

Create budget alerts to help me stay on top of my spending and notify me when I am about to exceed the \$200 monthly Google Maps credit ?

[GO TO GOOGLE MAPS PLATFORM](#)

flutter_dotenv



The screenshot shows the flutter_dotenv package page on pub.dev. The page title is "flutter_dotenv 5.1.0". It was published 11 months ago and is Dart 3 compatible. The package has 1.5K GitHub stars. The main navigation tabs include Readme, Changelog, Example, Installing, Versions, and Scores. The "Readme" tab is currently selected. The "About" section describes the package as a fork of mockturtle_dotenv, designed to work with Flutter. It explains that an environment is a set of variables known to a process and that the package parses .env files to merge their values with the Platform.environment map. The "Usage" section provides instructions for creating a .env file. The right sidebar displays statistics: 1536 likes, 130 pub points, and 100% popularity. It also shows the publisher as an unverified uploader and links to the repository on GitHub, issues, and contributing guidelines. Other sections include Documentation, API reference, License (MIT), Dependencies (flutter), and More (Packages that depend on).

flutter_dotenv 5.1.0

Published 11 months ago Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS 1.5K

Readme Changelog Example Installing Versions Scores

1536 130 100%

Likes Pub Points Popularity

Publisher unverified uploader

Metadata

Easily configure any flutter application with global variables using a .env file.

Repository (GitHub) View/report issues Contributing

About

This library is a fork of [mockturtle_dotenv](#) dart library, initially with slight changes to make it work with flutter.

An *environment* is the set of variables known to a process (say, PATH, PORT, ...). It is desirable to mimic the production environment during development (testing, staging, ...) by reading these values from a file.

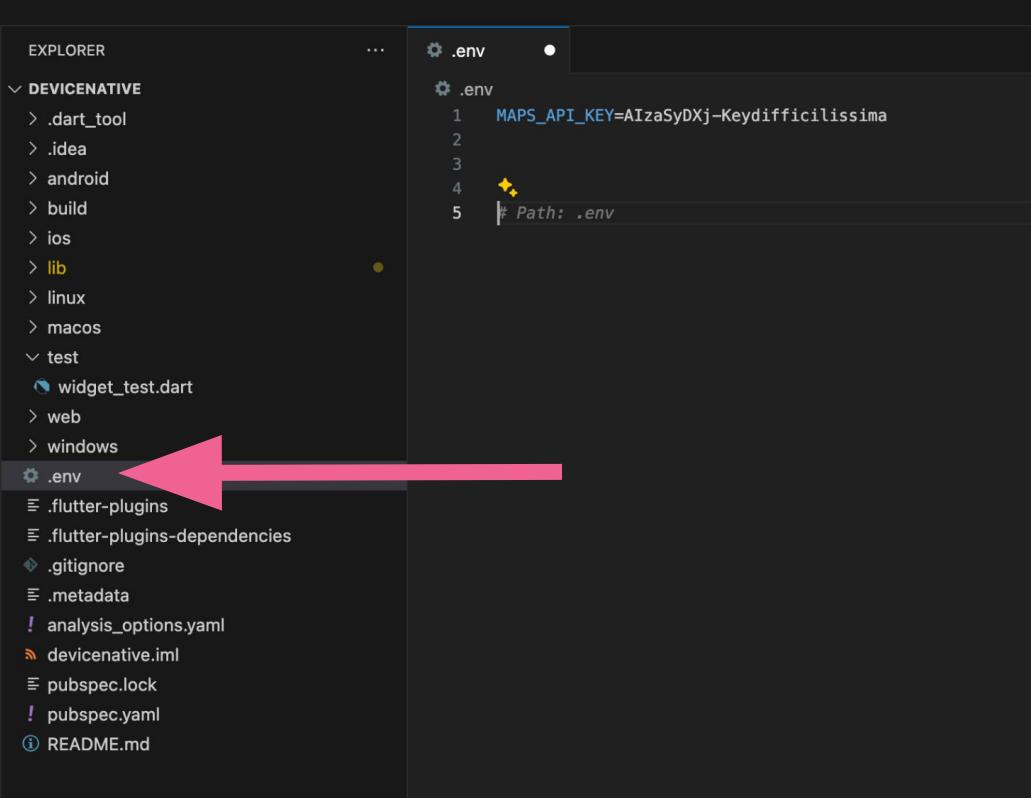
This library parses that file and merges its values with the built-in `Platform.environment` map.

Usage

1. Create a `.env` file in the root of your project with the example content:

```
FOO=foo  
BAR=bar
```

Create il vostro file .env

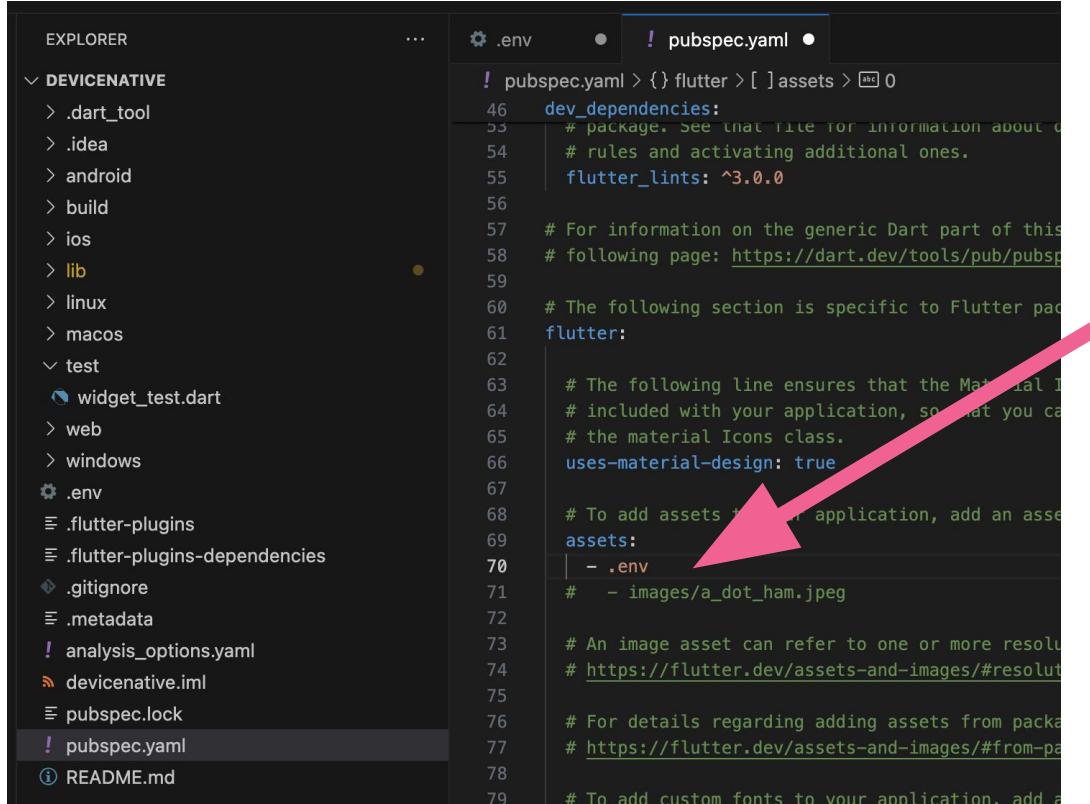


```
MAPS_API_KEY=AIzaSyDXj-Keydifficilissima
```

Path: .env

Se usate git per il vostro progetto ricordate di mettere il file .env nel .gitignore. Un malintenzionato potrebbe usare la vostra API Key e addebitare a voi i costi dei suoi servizi!!!!!!

Aggiungete il file alla disponibilità degli assets

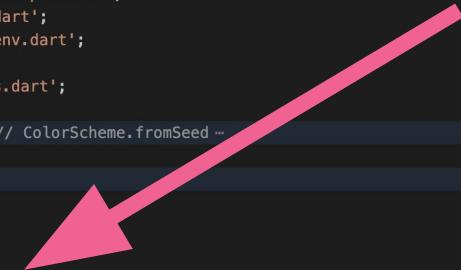


The screenshot shows the VS Code interface with the file `pubspec.yaml` open in the editor. The left sidebar displays the project structure under `EXPLORER`, including `DEVICENATIVE`, `.env`, `lib`, `test`, `widget_test.dart`, `web`, `windows`, `.env`, `.flutter-plugins`, `.flutter-plugins-dependencies`, `.gitignore`, `.metadata`, `analysis_options.yaml`, `devicenative.iml`, `pubspec.lock`, `pubspec.yaml`, and `README.md`. The right pane shows the content of the `pubspec.yaml` file:

```
! pubspec.yaml > {} flutter > [ ] assets > abc 0
46 dev_dependencies:
53 # package. See that file for information about d
54 # rules and activating additional ones.
55 flutter_lints: ^3.0.0
56
57 # For information on the generic Dart part of this
58 # following page: https://dart.dev/tools/pub/pubsp
59
60 # The following section is specific to Flutter pa
61 flutter:
62
63 # The following line ensures that the Material I
64 # included with your application, so that you can
65 # the material Icons class.
66 uses-material-design: true
67
68 # To add assets to your application, add an asse
69 assets:
70 | - .env
71 |   - images/a_dot_ham.jpeg
72
73 # An image asset can refer to one or more resolu
74 # https://flutter.dev/assets-and-images/#resolut
75
76 # For details regarding adding assets from packa
77 # https://flutter.dev/assets-and-images/#from-pa
78
79 # To add custom fonts to your application, add a
```

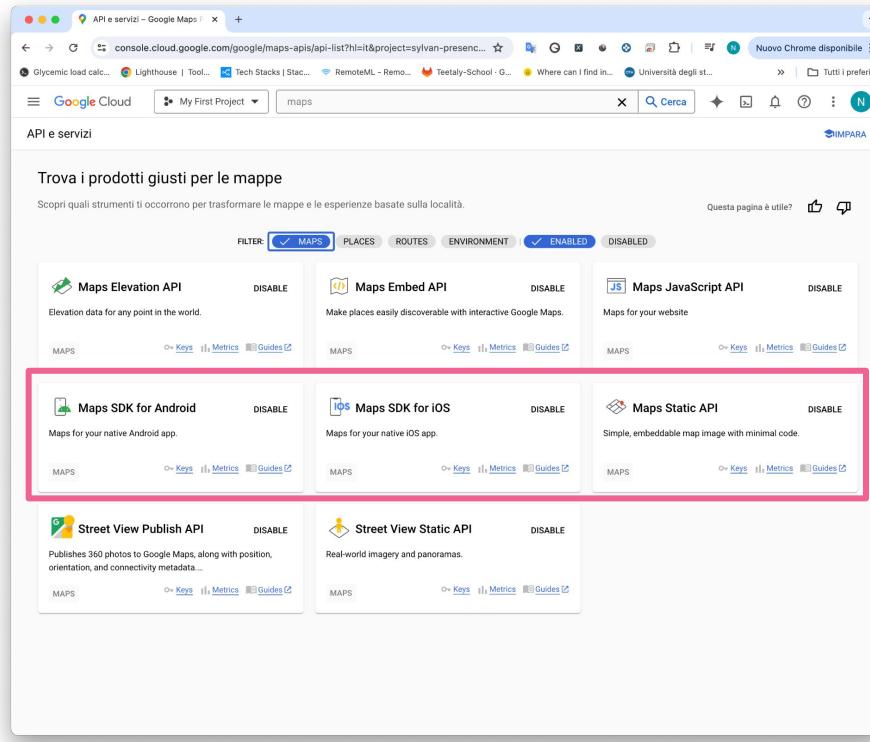
A pink arrow points to the `assets:` section at line 69.

Rendiamo asincrono il nostro main()

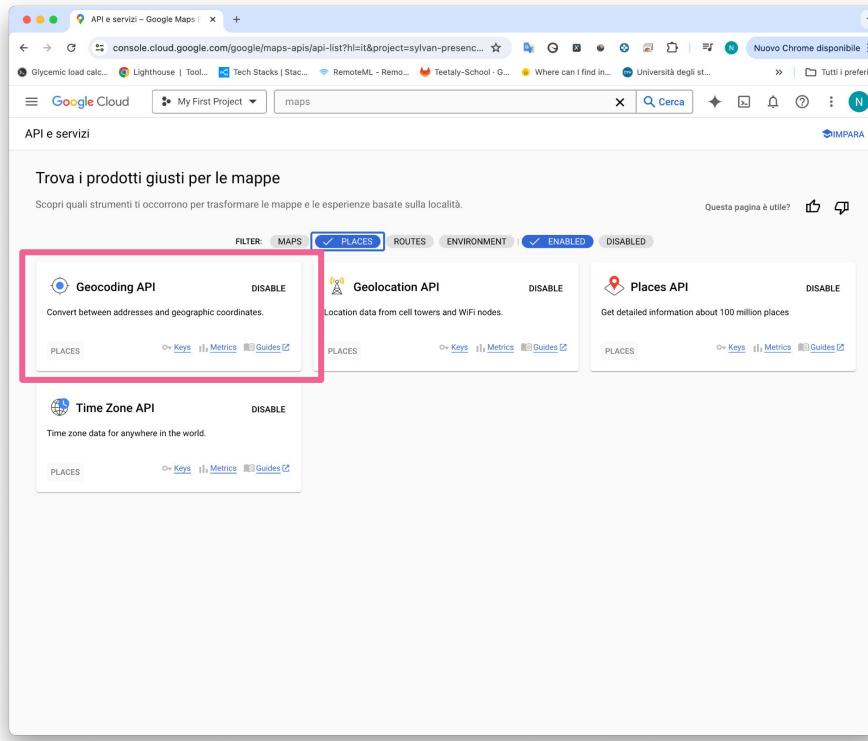


```
lib > main.dart > ...
1 import 'package:flutter/material.dart';
2 import 'package:flutter_riverpod/flutter_riverpod.dart';
3 import 'package:google_fonts/google_fonts.dart';
4 import 'package:flutter_dotenv/flutter_dotenv.dart';
5
6 import 'package:devicenative/screens/places.dart';
7
8 > final colorScheme = ColorScheme.fromSeed( // ColorScheme.fromSeed ...
13
14 > final theme = ThemeData().copyWith( ...
29
Run | Debug | Profile
30 Future main() async {
31   await dotenv.load(fileName: ".env");
32   runApp(
33     const ProviderScope(child: MyApp()),
34   );
35 }
36
37 class MyApp extends StatelessWidget {
38   const MyApp({Key? key}) : super(key: key);
39
40   @override
41   Widget build(BuildContext context) {
42     return MaterialApp(
43       title: 'Great Places',
44       theme: theme,
45       home: const PlacesScreen(),
46     ); // MaterialApp
47   }
48 }
```

Quali useremo



Quali useremo



Geocoding API

The screenshot shows a web browser window displaying the Google Maps Platform documentation for the Geocoding API. The URL in the address bar is developers.google.com/maps/documentation/geocoding/overview?hl=it. The page title is "Panoramica dell'API Geocod...". The navigation bar includes links for "Panoramica", "Prodotti", "Documentazione", "Altro", and "Cerca". The language is set to "Italiano".

The main content area is titled "Panoramica dell'API Geocoding". It features a video thumbnail with the text "How to convert addresses to coordinates" and "Geocoding / Reverse geocoding". Below the video, there is a note about using Plus Codes. The section "Perché utilizzare l'API Geocoding" explains the service's purpose for web and mobile applications.

A sidebar on the right is titled "Su questa pagina" and lists several sections: "Perché utilizzare l'API Geocoding", "Cosa puoi fare con l'API Geocoding", "Come funziona l'API Geocoding", "Risorse", "Come utilizzare l'API Geocoding", "Librerie client disponibili", and "Passaggi successivi".

The left sidebar contains a navigation tree under "Web Services > Geocoding API" with categories like "Guide", "Assistenza", "Filtro", "API Maps Geocoding", "Panoramica" (which is selected), "Inizia", "Configurazione", "Guide per gli sviluppatori", "Best practice", and "Fatturazione e report".

Geocoding API - Reverse Geocoding

The screenshot shows a web browser window displaying the Google Maps Platform documentation for the Geocoding API. The URL in the address bar is `developers.google.com/maps/documentation/geocoding/overview?hl=it`. The page has a sidebar on the left containing links to various Google Maps Platform services like Panoramic, Configuration, and Best practices. The main content area features a heading "Come funziona l'API Geocoding" and a sub-section "L'API Geocoding esegue sia la geocodifica sia la geocodifica inversa". It includes two bullet points: "Geocodifica" (which converts addresses like "1600 Amphitheatre Parkway, Mountain View, CA" into coordinates) and "Geocodifica inversa" (which converts coordinates or place IDs into readable addresses). Below this, there's a map of Africa and the Middle East with a search bar overlay that says "Inserisci una query da geocodificare o fai clic sulla mappa per invertire la codifica." To the right of the map, there's a sidebar titled "Su questa pagina" with links to related topics like "Perché utilizzare l'API Geocoding", "Cosa puoi fare con l'API Geocoding", and "Risorse". At the bottom of the page, there's a footer with links to "Norme e termini", "Criteri", and "Termini di servizio".

Analizzare l'endpoint e la sua risposta

The screenshot shows a browser window with the URL developers.google.com/maps/documentation/geocoding/requests-reverse-geo.... The page is titled "Google Maps Platform" and has a navigation bar with links to "Panoramica", "Prodotti", "Documentazione", "Altro", and "Cerca". The "Documentazione" tab is active. On the left, there's a sidebar with links to "API Maps Geocoding", "Panoramica", "Inizia", "Configurazione", "Configurare il progetto Google Cloud", "Utilizzo di chiavi API", and "Guide per gli sviluppatori". The main content area displays a JSON response for a reverse geocoding request. The JSON object includes fields like "long_name", "short_name", "types", "formatted_address", "geometry", "location", "lat", "lng", "location_type", "viewport", and "northeast". A sidebar on the right lists "Su questa pagina" sections: "Richieste di geocodifica inversa", "Parametri obbligatori", "Parametri facoltativi", "Esempio di geocodifica inversa", "Geocodifica inversa filtrata per tipo", and "Risposte di".

```
{
  "address_components": [
    {
      "long_name": "11211",
      "short_name": "11211",
      "types": [ "postal_code" ]
    }
  ],
  "formatted_address": "277 Bedford Avenue, Brooklyn, NY 11211, USA",
  "geometry": {
    "location": {
      "lat": 40.714232,
      "lng": -73.9612889
    },
    "location_type": "ROOFTOP",
    "viewport": {
      "northeast": {
        "lat": 40.715442,
        "lng": -73.960079
      },
      "southwest": {
        "lat": 40.713022,
        "lng": -73.962501
      }
    }
  }
}
```

Integriamo LocationInput con la chiamata al servizio

```
setState(() {
  _isGettingLocation = true;
});

locationData = await location.getLocation();
final lat = locationData.latitude;
final lng = locationData.longitude;
final key = dotenv.env['MAPS_API_KEY'];

if (lat == null || lng == null) {
  return;
}

final url = Uri.parse(
  'https://maps.googleapis.com/maps/api/geocode/json?latlng=$lat,$lng&key=$key');
final response = await http.get(url);
final resData = json.decode(response.body);
final address = resData['results'][0]['formatted_address'];
print(address);

setState(() {
  _isGettingLocation = false;
});
```

Il risultato

```
D/EGL_emulation(30876): app_time_stats: avg=22.39ms min=10.92ms max=33.97ms count=45
D/EGL_emulation(30876): app_time_stats: avg=25.57ms min=14.64ms max=88.53ms count=40
I/flutter (30876): 1600 Amphitheatre Pkwy Building 43, Mountain View, CA 94043, USA
```

Esercizio

Predisponete, se vi va, il vostro account developer Google ed usate la vostra API Key all'interno del codice location.zip per verificarne il funzionamento, eventualmente facendo la build su un dispositivo fisico.

Mi raccomando, attenzione all'uso delle API Key se usate il progetto su Git!!!!!!

Visualizzazione della location su mappa

Aggiorno il mio data model per supportare le coordinate

```
lib > models > place.dart > ...
1   import 'dart:io';
2
3   import 'package:uuid/uuid.dart';
4
5   const uuid = Uuid();
6
7   class PlaceLocation {
8     const PlaceLocation({
9       required this.latitude,
10      required this.longitude,
11      required this.address,
12    });
13
14     final double latitude;
15     final double longitude;
16     final String address;
17   }
18
19   class Place {
20     Place({
21       required this.title,
22       required this.image,
23       required this.location,
24     }) : id = uuid.v4();
25
26     final String id;
27     final String title;
28     final File image;
29     final PlaceLocation location;
30   }
31 }
```



Adeguo LocationInput per salvare il risultato su un oggetto PlaceLocation

```
class _LocationInputState extends State<LocationInput> {
  PlaceLocation? _pickedLocation;
  var _isGettingLocation = false;

  void _getCurrentLocation() async {
    Location location = Location();

    bool serviceEnabled;
    PermissionStatus permissionGranted;
    LocationData locationData;

    serviceEnabled = await location.serviceEnabled();
    if (!serviceEnabled) { ...
    }

    permissionGranted = await location.hasPermission();
    if (permissionGranted == PermissionStatus.denied) { ...
    }

    setState(() { ...
    }

    locationData = await location.getLocation();
    final lat = locationData.latitude;
    final lng = locationData.longitude;
    final key = dotenv.env['MAPS_API_KEY'];

    if (lat == null || lng == null) { ...
    }

    final url = Uri.parse('...
    final response = await http.get(url);
    final resData = json.decode(response.body);
    final address = resData['results'][0]['formatted_address'];
    print(address);

    setState(() {
      _pickedLocation = PlaceLocation(
        latitude: lat,
        longitude: lng,
        address: address,
      );
      _isGettingLocation = false;
    });
  }
}
```

API Maps Static

Panoramica | Maps Static API +

developers.google.com/maps/documentation/maps-static/overview?hl=it

Glycemic load calc... Lighthouse | Tool... Tech Stacks | Stack... RemoteML - Remo... Teetally-School - G... Where can I find in... Università degli st...

Nuovo Chrome disponibile

Google Maps Platform

Panoramica Prodotti Documentazione Altro Cerca Italiano

Filtra

API Maps Static

Panoramica Inizia

Configurazione

Configura il progetto Google Cloud

Usa chiavi API

Usa una firma digitale

Personalizza con la personalizzazione delle mappe basata su cloud

Personalizzare con lo stile JSON

Servizi web

Best practice

Fatturazione e report

Utilizzo e fatturazione

Report e monitoraggio

Norme e termini

Termini di servizio

Un rapido esempio

L'esempio seguente contiene l'URL di un'immagine dell'API Maps Static del centro di New York, visualizzata di seguito:

https://maps.googleapis.com/maps/api/staticmap?center=Brooklyn+Bridge,New+York,NY&zoom=13&size=640x480&markers=color:blue%7Clabel:S%7C40.702147,-74.015794&markers=color:green%7Clabel:G%7C40.711614,-74.012984&key=YOUR_API_KEY&signature=YOUR_SIGNATURE

Tieni presente che non devi fare nulla di "speciale" per far sì che l'immagine venga visualizzata nella pagina. Non è necessario JavaScript. Dovendo solo creare un URL e inserirlo all'interno di un tag . Puoi inserire un'API Google Maps Static in qualsiasi punto della pagina web dove inserire un'immagine.

Autenticazione, quote, prezzi e criteri

Autenticazione

Per utilizzare l'API Maps Static, devi prima abilitarla e ottenere le credenziali di autenticazione corrette. Per maggiori informazioni, consulta la [guida introduttiva a Google Maps Platform](#).

Su questa pagina

Un rapido esempio

Autenticazione, quote, prezzi e criteri

Autenticazione

Quote e prezzi

Criteri

Scopri di più

Integro la funzione per mostrare l'immagine della mappa in PlaceLocation

```
class _LocationInputState extends State<LocationInput> {
  PlaceLocation? _pickedLocation;
  var _isGettingLocation = false;
  final key = dotenv.env['MAPS_API_KEY'];

  String get locationImage {
    if (_pickedLocation == null) {
      return '';
    }
    final lat = _pickedLocation!.latitude;
    final lng = _pickedLocation!.longitude;
    return 'https://maps.googleapis.com/maps/api/staticmap?center=$lat,$lng&zoom=16&size=600x300&maptype=roadmap&markers=color:red%7Clabel:A%7C$lat,$lng&key=$key';
  }
}
```

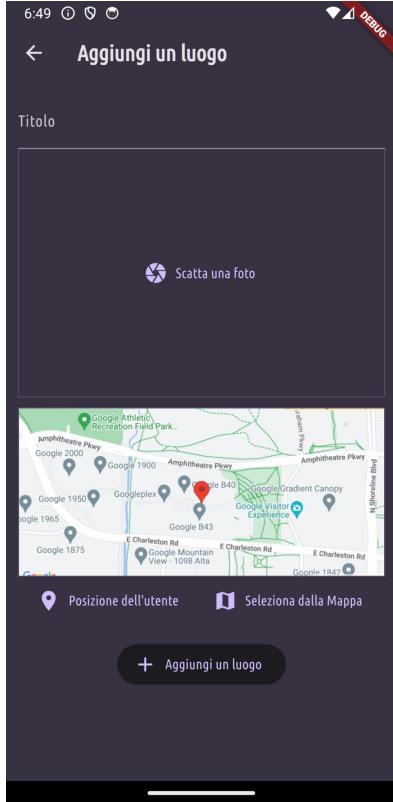


Integro la funzione per mostrare l'immagine della mappa in PlaceLocation

```
@override
Widget build(BuildContext context) {
    Widget previewContent = Text(
        'Nessuna location selezionata',
        textAlign: TextAlign.center,
        style: Theme.of(context).textTheme.bodyLarge!.copyWith(
            color: Theme.of(context).colorScheme.onSurface,
        ),
    );
    if (_pickedLocation != null) {
        previewContent = Image.network(
            locationImage,
            fit: BoxFit.cover,
            width: double.infinity,
            height: double.infinity,
        );
    }
    if (_isGettingLocation) {
        previewContent = const CircularProgressIndicator();
    }
}
```



Il risultato



Per questa build ho
dovuto
temporaneamente
commentare le
aggiunte sul data
model

Aggiornamento Riverpod

Aggiornamento del provider Riverpod

```
lib > providers > user_places.dart > ...
1  import 'package:flutter_riverpod/flutter_riverpod.dart';
2  import 'dart:io';
3
4  import 'package:devicenative/models/place.dart';
5
6  class UserPlacesNotifier extends StateNotifier<List<Place>> {
7    UserPlacesNotifier() : super(const []);
8
9    void addPlace(String title, File image, PlaceLocation location) {
10      // questo potete scriverlo come volete, è a vostra discrezione
11      // potete passare anche un oggetto Place direttamente
12      final newPlace = Place(title: title, image: image, location: location);
13      // aggiorna lo stato con il nuovo luogo e il resto della lista "spalmata"
14      state = [newPlace, ...state];
15    }
16  }
17
18  final userPlacesProvider =
19  |  | StateNotifierProvider<UserPlacesNotifier, List<Place>>(
20  |  |   (ref) => UserPlacesNotifier(),
21  |  ); // StateNotifierProvider
```

Ma ovviamente
dobbiamo anche
adeguare widget e
screen come
abbiamo fatto
quando abbiamo
aggiunto l'immagine!

Adeguamento di LocationInput



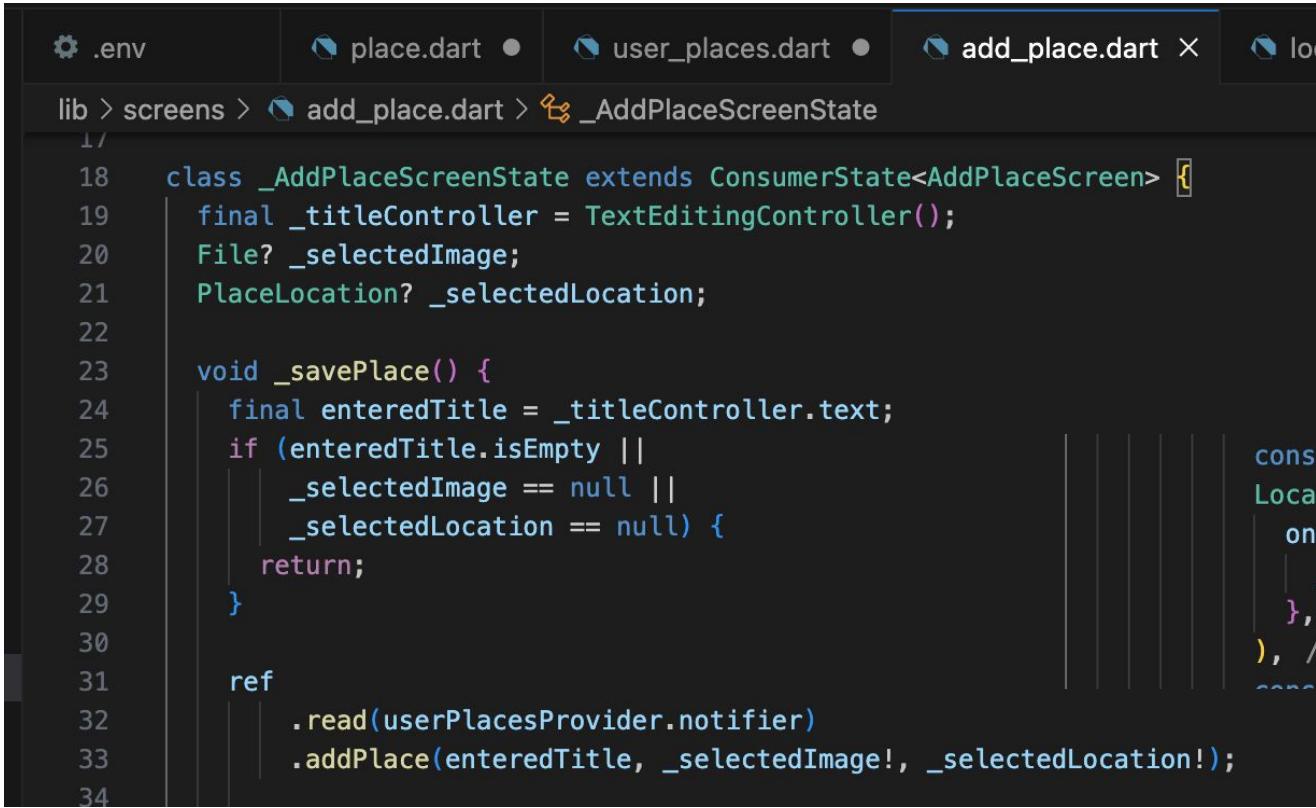
```
lib > widgets > location_input.dart > _LocationInputState
1 import 'package:flutter/material.dart';
2 import 'package:http/http.dart' as http;
3 import 'dart:convert';
4 import 'package:location/location.dart';
5 import 'package:flutter_dotenv/flutter_dotenv.dart';
6 import 'package:devicenative/models/place.dart';
7
8 class LocationInput extends StatefulWidget {
9   const LocationInput({super.key, required this.onSelectLocation});
10
11   final void Function(PlaceLocation location) onSelectLocation;
12 }
```

Adeguamento di LocationInput



```
lib > widgets > location_input.dart > _LocationInputState
19   class _LocationInputState extends State<LocationInput> {
33     void _getCurrentLocation() async {
70       final response = await http.get(url);
71       final resData = json.decode(response.body);
72       final address = resData['results'][0]['formatted_address'];
73
74       setState(() {
75         _pickedLocation = PlaceLocation(
76           latitude: lat,
77           longitude: lng,
78           address: address,
79         );
80         _isGettingLocation = false;
81       });
82       widget.onSelectLocation(_pickedLocation!);
83     }
84   }
```

Lifting up state: AddPlaceScreen

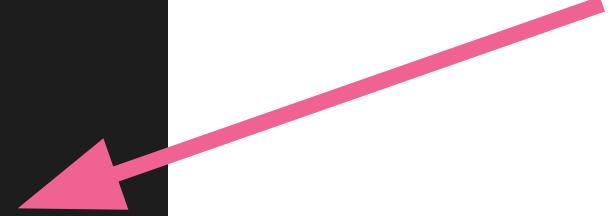


The screenshot shows a code editor with several tabs at the top: '.env', 'place.dart', 'user_places.dart', 'add_place.dart' (which is the active tab), and 'location.dart'. Below the tabs, a breadcrumb navigation bar indicates the file structure: 'lib > screens > add_place.dart > _AddPlaceScreenState'. The main content area displays the code for the `_AddPlaceScreenState` class.

```
17
18 class _AddPlaceScreenState extends ConsumerState<AddPlaceScreen> {
19   final _titleController = TextEditingController();
20   File? _selectedImage;
21   PlaceLocation? _selectedLocation;
22
23   void _savePlace() {
24     final enteredTitle = _titleController.text;
25     if (enteredTitle.isEmpty ||
26         _selectedImage == null ||
27         _selectedLocation == null) {
28       return;
29     }
30
31     ref
32       .read(userPlacesProvider.notifier)
33       .addPlace(enteredTitle, _selectedImage!, _selectedLocation!);
34 }
```

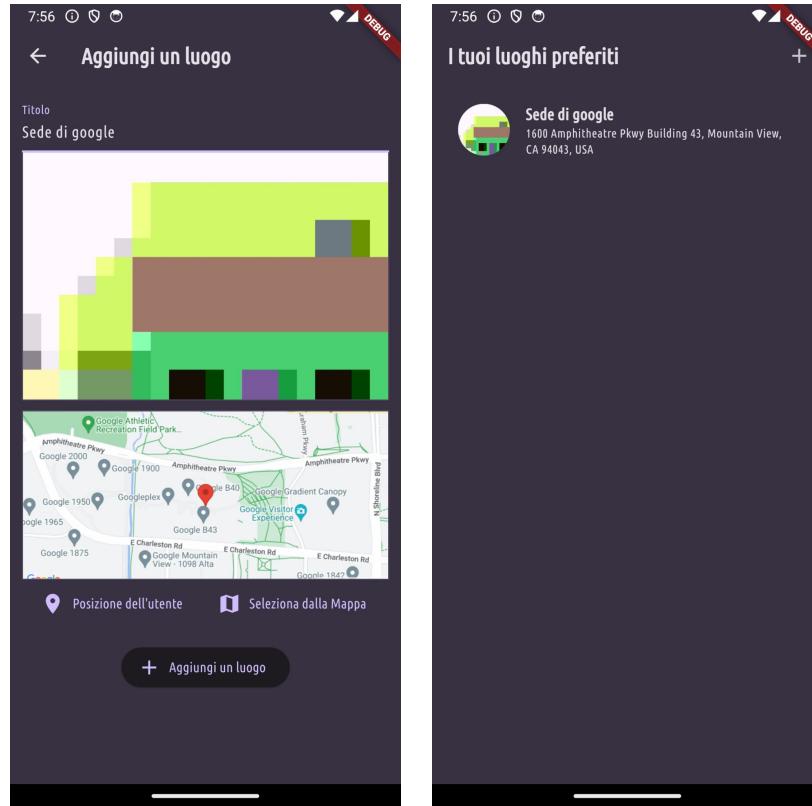
The code implements a state class for the `AddPlaceScreen`. It contains fields for a title controller, selected image, and selected location. A `_savePlace` method checks if the title is empty or if the selected image or location are null, returning if any are. Otherwise, it reads from the `userPlacesProvider` and adds a new place with the provided title, image, and location.

Modifichiamo PlacesList per verificare il funzionamento



```
lib > widgets > places_list.dart > PlacesList > build
  6   class PlacesList extends StatelessWidget {
12     Widget build(BuildContext context) {
13
14       return ListView.builder(
15         itemCount: places.length,
16         itemBuilder: (ctx, index) => ListTile(
17           leading: CircleAvatar(
18             radius: 26,
19             backgroundImage: FileImage(places[index].image),
20           ), // CircleAvatar
21           title: Text(
22             places[index].title,
23             style: Theme.of(context).textTheme.titleMedium!.copyWith(
24               color: Theme.of(context).colorScheme.onSurface,
25             ),
26           ), // Text
27           subtitle: Text(
28             places[index].location.address,
29             style: Theme.of(context).textTheme.bodySmall!.copyWith(
30               color: Theme.of(context).colorScheme.onSurface,
31             ),
32           ), // Text
33           onTap: () {
34             Navigator.of(context).push(
35               MaterialPageRoute(
36                 builder: (ctx) => PlaceDetailScreen(place: places[index]),
37               ), // MaterialPageRoute
38             );
39           },
40         ), // ListTile
41       ); // ListView.builder
42     }
43   }
44 }
```

Il Risultato



E se volessimo
aggiungere una
mappa anche nella
schermata di
dettaglio?

Aggiornamento dello screen PlaceDetailScreen (1)

```
.lib > screens > place_detail.dart > PlaceDetailScreen > build
1 import 'package:flutter/material.dart';
2 import 'package:flutter_dotenv/flutter_dotenv.dart';
3 import 'package:devicenative/models/place.dart';
4
5 class PlaceDetailScreen extends StatelessWidget {
6   const PlaceDetailScreen({super.key, required this.place});
7
8   final Place place;
9
10  String get locationImage {
11    final key = dotenv.env['MAPS_API_KEY'];
12    final lat = place.location.latitude;
13    final lng = place.location.longitude;
14    return 'https://maps.googleapis.com/maps/api/staticmap?center=\$lat,\$lng&zoom=16&size=600x300&maptyle=roadmap&markers=color:red%7Clabel:location%7C\$lat,\$lng&key=\$key';
15  }
16
17  @override
18  Widget build(BuildContext context) {
19    return Scaffold(
20      appBar: AppBar(
21        title: Text(place.title),
22      ). // AppBar
```

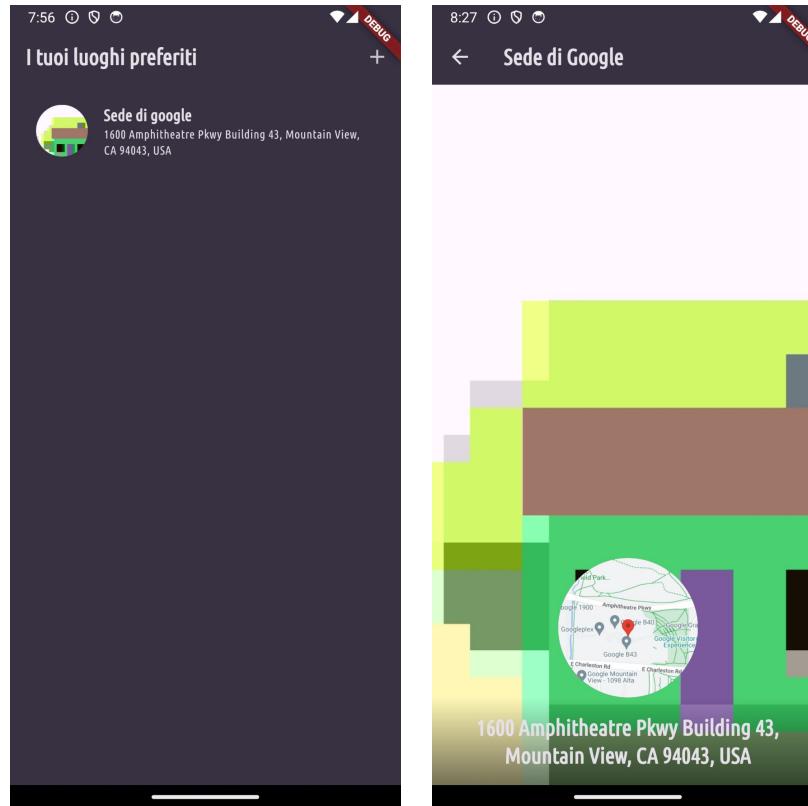
Aggiornamento dello screen PlaceDetailScreen (2)

```
lib > screens > place_detail.dart > PlaceDetailScreen > build
5   class PlaceDetailScreen extends StatelessWidget {
6
7     @override
8     Widget build(BuildContext context) {
9       return Scaffold(
10         appBar: AppBar(
11           title: Text(place.title),
12         ), // AppBar
13         body: Stack(
14           children: [
15             Image.file(
16               place.image,
17               fit: BoxFit.cover,
18               width: double.infinity,
19               height: double.infinity,
20             ), // Image file
21             Positioned(
22               bottom: 0,
23               left: 0,
24               right: 0,
25               child: Column(
26                 children: [
27                   CircleAvatar(
28                     radius: 70,
29                     backgroundImage: NetworkImage(locationImage),
30                   ), // CircleAvatar
31                   Container(
32                     alignment: Alignment.center,
33                     padding: const EdgeInsets.symmetric(
34                       horizontal: 24,
35                       vertical: 16,
36                     ), // EdgeInsets.symmetric
37                     decoration: const BoxDecoration(
38                       gradient: LinearGradient(
39                         colors: [
40                           Colors.transparent,
41                           Colors.black54,
42                         ],
43                         begin: Alignment.topCenter,
44                         end: Alignment.bottomCenter,
45                       ), // LinearGradient
46                     ), // BoxDecoration
47                     child: Text(
48                       place.location.address,
49                       textAlign: TextAlign.center,
50                       style: Theme.of(context).textTheme.titleLarge!.copyWith(
51                         color: Theme.of(context).colorScheme.onSurface,
52                         ),
53                     ), // Text
54                   ), // Container
55                 ],
56               ), // Column
57             ) // Positioned
58           ],
59         ), // Stack
60       ); // Scaffold
61     }
62   }
63 }
```

Aggiornamento dello screen PlaceDetailScreen

- Il widget **PlaceDetailScreen** richiede un parametro place di tipo Place nel suo costruttore. Questo oggetto Place rappresenta un luogo e contiene la posizione geografica e un'immagine del luogo;
- Il widget ha un getter **locationImage** che costruisce un URL per una mappa statica di Google Maps. Questo URL include la latitudine e la longitudine del luogo, e utilizza una API key dal file di configurazione .env;
- Il metodo **build** del widget restituisce un **Scaffold** che contiene un **AppBar** con il titolo del luogo e un Stack nel corpo;
- Il **Stack** sovrappone più figli in un ordine di profondità. Il primo figlio è un'immagine del luogo che copre l'intero **Stack**. Il secondo figlio è un **Positioned** widget che contiene una **Column** con un **CircleAvatar** e un **Container**;
- Il **CircleAvatar** mostra un'immagine di una mappa statica del luogo, ottenuta dall'URL costruito dal getter **locationImage**;
- Il **Container** mostra l'indirizzo del luogo su uno sfondo sfumato che va da trasparente a nero. L'indirizzo è centrato nel Container e il suo stile è definito dal tema corrente dell'App.

Il risultato



Pausa

Scaricate locationmap.zip dalla repo, analizzate e provate il codice appena visto, focalizzandovi sulla comunicazione tra le componenti.

Integrazione di Google Maps

https://pub.dev/packages/google_maps_flutter

The screenshot shows the official package page for `google_maps_flutter` on `pub.dev`. The page is titled "google_maps_flutter 2.6.1". It includes a brief description: "A Flutter plugin that provides a Google Maps widget.", support information ("Support: SDK 20+, iOS 14+, Same as Flutter's"), and usage instructions. The "Usage" section notes that to use the plugin, it must be added as a dependency in the `pubspec.yaml` file. The "Getting Started" section lists steps for enabling Google Maps on both Android and iOS, including navigating to the Google Developers Console and selecting the "Maps" API. On the right side of the page, there are sections for "Publisher" (flutter.dev), "Metadata" (a Flutter plugin for integrating Google Maps in iOS and Android applications), "Repository" (GitHub), "Topics" (#google-maps #google-maps-flutter #map), "Documentation" (API reference), "License" (BSD-3-Clause), and "Dependencies" (flutter, google_maps_flutter_andROID, google_maps_flutter_iOS).

Published 32 days ago • [@ flutter.dev](#) (Dart 3 compatible)

SDK FLUTTER PLATFORM ANDROID IOS WEB

Readme Changelog Example Installing Versions Scores

3.9K

3967 LIKES 140 PUB POINTS 100% POPULARITY

Publisher [@ flutter.dev](#)

Metadata

A Flutter plugin for integrating Google Maps in iOS and Android applications.

Repository ([GitHub](#))

View/report issues

Contributing

Topics

#google-maps #google-maps-flutter #map

Documentation

[API reference](#)

License

BSD-3-Clause ([LICENSE](#))

Dependencies

flutter, google_maps_flutter_andROID, google_maps_flutter_iOS

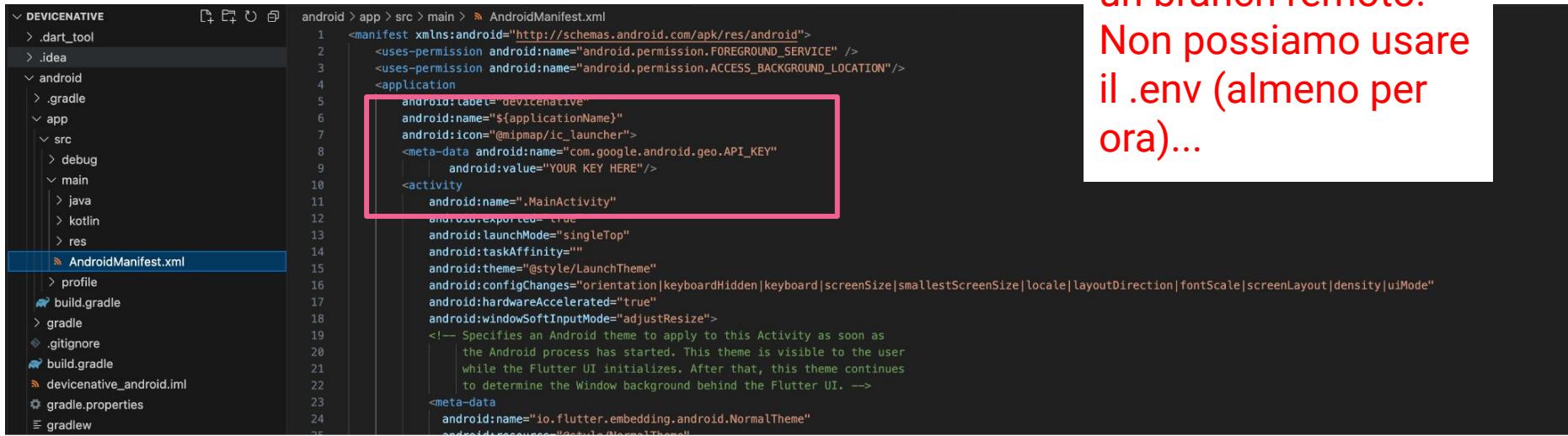
google_maps_flutter: Android minSdkVersion

```
✓ DEVICENATIVE
  > .dart_tool
  > .idea
  ✓ android
  > .gradle
  ✓ app
  > src
  ⚡ build.gradle
  > gradle
  ⚡ .gitignore
  ⚡ build.gradle
  ⚡ devicenative_android.iml
  ⚡ gradle.properties
  ⚡ gradlew
  ⚡ gradlew.bat
  ⚡ local.properties
  ⚡ settings.gradle
  > ios
  > lib
  > linux
  > macos
  ✓ test
  ⚡ widget_test.dart
  > web
```

```
android > app > ⚡ build.gradle
26   android {
27     minSdkVersion = flutter.minSdkVersion
28
29     compileOptions {
30       sourceCompatibility = JavaVersion.VERSION_1_8
31       targetCompatibility = JavaVersion.VERSION_1_8
32     }
33
34   }
35
36   defaultConfig {
37     // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).
38     applicationId = "com.example.devicenative"
39     // You can update the following values to match your application needs.
40     // For more information, see: https://docs.flutter.dev/deployment/android#reviewing-the-gradle-build-configuration.
41     minSdk = 20
42     targetSdk = flutter.targetSdkVersion
43     versionCode = flutterVersionCode.toInt()
44     versionName = flutterVersionName
45   }
46
47   buildTypes {
48     release {
49       // TODO: Add your own signing config for the release build.
50       // Signing with the debug keys for now, so `flutter run --release` works.
51       signingConfig = signingConfigs.debug
52     }
53   }
54 }
```

Se non funziona la build, usate 21 come minSdk

Aggiunta dell'API Key nell'Android Manifest



The screenshot shows the Android Studio project structure on the left and the `AndroidManifest.xml` file content on the right. A red box highlights the line where the API key is added:

```
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android">
2   <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
3   <uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
4   <application
5     android:label="@string/app_name"
6     android:name="${applicationName}"
7     android:icon="@mipmap/ic_launcher">
8       <meta-data android:name="com.google.android.geo.API_KEY"
9         android:value="YOUR KEY HERE"/>
10      <activity
11        android:name=".MainActivity"
12        android:exported="true"
13        android:launchMode="singleTop"
14        android:taskAffinity=""
15        android:theme="@style/LaunchTheme"
16        android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smallestScreenSize|locale|layoutDirection|fontScale|screenLayout|density|uiMode"
17        android:hardwareAccelerated="true"
18        android:windowSoftInputMode="adjustResize">
19          <!-- Specifies an Android theme to apply to this Activity as soon as
20              the Android process has started. This theme is visible to the user
21              while the Flutter UI initializes. After that, this theme continues
22              to determine the Window background behind the Flutter UI. -->
23          <meta-data
24            android:name="io.flutter.embedding.android.NormalTheme"
25            android:resource="@style/NormalTheme"/>
```

**Attenzione,
cancellate la chiave
prima di inviarla ad
un branch remoto.
Non possiamo usare
il .env (almeno per
ora)...**

Creazione dello screen MapScreen (1)

```
lib > screens > map.dart > _MapScreenState > build
  1 import 'package:flutter/material.dart';
  2 import 'package:google_maps_flutter/google_maps_flutter.dart';
  3
  4 import 'package:devicenative/models/place.dart';
  5
  6 class MapScreen extends StatefulWidget {
  7   const MapScreen({
  8     super.key,
  9     this.location = const PlaceLocation(
 10       latitude: 37.422,
 11       longitude: -122.084,
 12       address: '',
 13     ),
 14     this.isSelecting = true,
 15   });
 16
 17   final PlaceLocation location;
 18   final bool isSelecting;
 19
 20   @override
 21   State<MapScreen> createState() {
 22     return _MapScreenState();
 23   }
 24 }
 25 }
```

Creazione dello screen MapScreen (2)

```
25
26 class _MapScreenState extends State<MapScreen> {
27   @override
28   Widget build(BuildContext context) {
29     return Scaffold(
30       appBar: AppBar(
31         title: Text(widget.isSelecting
32             ? 'Scegli la tua Posizione'
33             : 'La tua Posizione'), // Text
34         actions: [
35           if (widget.isSelecting)
36             IconButton(
37               icon: const Icon(Icons.save),
38               onPressed: () {},
39             ), // IconButton
40           ], // AppBar
41       body: GoogleMap(
42         initialCameraPosition: CameraPosition(
43           target: LatLng(
44             widget.location.latitude,
45             widget.location.longitude),
46           zoom: 16,
47         ), // CameraPosition
48         markers: {
49           Marker(
50             markerId: const MarkerId('marker1'),
51             position: LatLng(
52               widget.location.latitude,
53               widget.location.longitude,
54             ), // LatLng
55             ), // Marker
56           },
57         ), // GoogleMap
58       ); // Scaffold
59     } // }
60   }
61 }
```

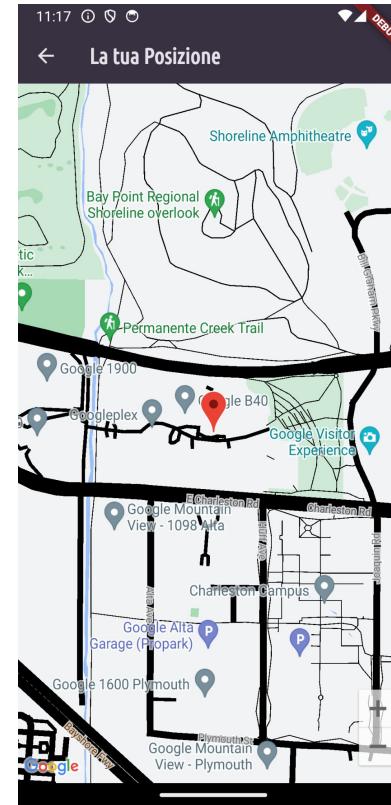
Creazione dello screen MapScreen

- Il widget **MapScreen** richiede due parametri nel suo costruttore: **location** e **isSelecting**. **location** è un oggetto **PlaceLocation** che rappresenta una posizione geografica con latitudine, longitudine e indirizzo. **isSelecting** è un booleano che indica se l'utente sta attualmente selezionando una posizione sulla mappa. Al momento abbiamo inserito intenzionalmente dei dati statici;
- La classe **_MapScreenState** ha un metodo build che restituisce un widget **Scaffold**. Questo **Scaffold** contiene un **AppBar** con un titolo che cambia a seconda del valore di **isSelecting**, e un **IconButton** che viene mostrato solo se **isSelecting** è true;
- Il corpo del **Scaffold** è un widget **GoogleMap** che mostra una mappa centrata sulla posizione specificata da **location**, con uno zoom di 16. La mappa ha un solo **Marker** che indica la posizione;
- L'**IconButton** nel **AppBar** ha un gestore **onPressed** vuoto, perché ancora non specifichiamo l'azione da compiere.

Modifico PlaceDetailScreen per raggiungere la mappa

```
lib > screens > place_detail.dart > PlaceDetailScreen > build
7   class PlaceDetailScreen extends StatelessWidget {
20     Widget build(BuildContext context) {
36       right: 0,
37       child: Column(
38         children: [
39           GestureDetector(
40             onTap: () {
41               Navigator.of(context).push(
42                 MaterialPageRoute(
43                   builder: (ctx) => MapScreen(
44                     location: place.location,
45                     isSelecting: false,
46                     ), // MapScreen
47                   ), // MaterialPageRoute
48                 );
49             },
50             child: CircleAvatar(
51               radius: 70,
52               backgroundImage: NetworkImage(locationImage),
53             ), // CircleAvatar
54             ), // GestureDetector
55             Container(
56               alignment: Alignment.center,
57               padding: const EdgeInsets.symmetric(
```

Il risultato



Interazione con la mappa

Adeguo MapScreen per interagire con la mappa

```
class _MapScreenState extends State<MapScreen> {
  LatLng? _pickedLocation;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.isSelecting
          ? 'Scegli la tua Posizione'
          : 'La tua Posizione'), // Text
        actions: [
          if (widget.isSelecting)
            IconButton(
              icon: const Icon(Icons.save),
              onPressed: () {},
            ), // IconButton
        ], // AppBar
      ),
      body: GoogleMap(
        onTap: widget.isSelecting
          ? null
          : (position) {
              setState(() {
                _pickedLocation = position;
              });
            },
        initialCameraPosition: CameraPosition(
          target: LatLng(
            widget.location.latitude,
            widget.location.longitude,
          ), // LatLng
          zoom: 16,
        ), // CameraPosition
        markers: (_pickedLocation == null && widget.isSelecting)
          ? {}
          : [
            Marker(
              markerId: const MarkerId('m1'),
              position: _pickedLocation ??
                LatLng(
                  widget.location.latitude,
                  widget.location.longitude,
                ), // LatLng
            ), // Marker
          ],
      ),
    );
  }
}
```

In questo modo
al tocco su
mappa ottengo
una nuova
posizione

Adeguo MapScreen per interagire con la mappa

- Il widget MapScreen richiede due parametri nel suo costruttore: location e **isSelecting**. location è un oggetto PlaceLocation che rappresenta una posizione geografica con latitudine, longitudine e indirizzo. **isSelecting** è un booleano che indica se l'utente sta attualmente selezionando una posizione sulla mappa;
- La classe _MapScreenState ha un metodo build che restituisce un widget Scaffold. Questo Scaffold contiene un AppBar con un titolo che cambia a seconda del valore di **isSelecting**, e un IconButton che viene mostrato solo se **isSelecting** è true;
- Il corpo del Scaffold è un widget GoogleMap che mostra una mappa centrata sulla posizione specificata da location, con uno zoom di 16. La mappa ha un solo Marker che indica la posizione;
- L'IconButton nell'AppBar ha un gestore onPressed che chiama Navigator.of(context).pop(_pickedLocation). Questo farà ritornare il widget MapScreen alla schermata precedente e passerà la posizione selezionata (_pickedLocation) come risultato della route;
- Se **isSelecting** è true, l'utente può toccare un punto sulla mappa per selezionare una posizione. Quando l'utente tocca la mappa, il gestore **onTap** del widget GoogleMap viene chiamato con la posizione toccata. Questo gestore chiama setState per aggiornare **_pickedLocation** con la posizione toccata;
- Se **isSelecting** è false, il gestore onTap è null, il che significa che toccare la mappa non farà nulla.

Adeguo LocationInput per la selezione da mappa

```
    TextButton.icon(  
      icon: const Icon(Icons.location_on),  
      label: const Text('Posizione dell\'utente'),  
      onPressed: _getCurrentLocation,  
    ), // TextButton.icon  
    TextButton.icon(  
      icon: const Icon(Icons.map),  
      label: const Text('Seleziona dalla Mappa'),  
      onPressed: _selectOnMap,  
    ), // TextButton.icon  
  ],  
) // Row
```



Adeguo LocationInput per la selezione da mappa

```
void _selectOnMap() async {
  final pickedLocation = await Navigator.of(context).push<LatLng>(
    MaterialPageRoute(
      builder: (ctx) => const MapScreen(),
    ), // MaterialPageRoute
  );

  if (pickedLocation == null) {
    return;
  }
}
```

Adeguo LocationInput per la selezione da mappa

```
Future<void> _savePlace(double latitude, double longitude) async {
  final url = Uri.parse(
    'https://maps.googleapis.com/maps/api/geocode/json?latlng=$latitude,$longitude&key=$key');
  final response = await http.get(url);
  final resData = json.decode(response.body);
  final address = resData['results'][0]['formatted_address'];

  setState(() {
    _pickedLocation = PlaceLocation(
      latitude: latitude,
      longitude: longitude,
      address: address,
    );
    _isGettingLocation = false;
  });

  widget.onSelectLocation(_pickedLocation!);
}
```

Adeguo LocationInput per la selezione da mappa

```
void _selectOnMap() async {
    final pickedLocation = await Navigator.of(context).push<LatLng>(
        MaterialPageRoute(
            builder: (ctx) => const MapScreen(),
        ), // MaterialPageRoute
    );

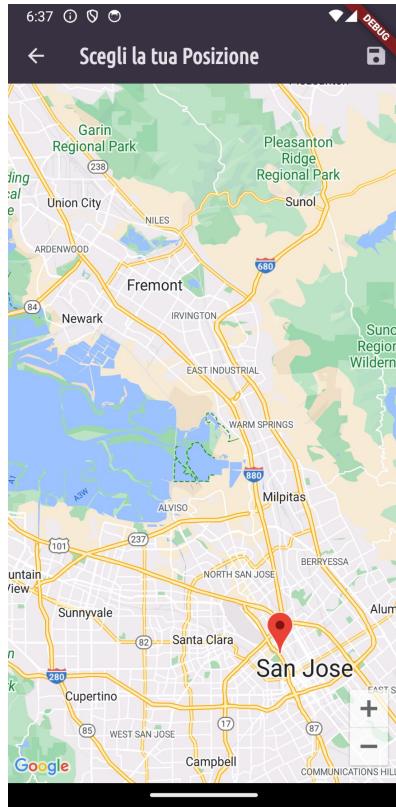
    if (pickedLocation == null) {
        return;
    }
    _savePlace(pickedLocation.latitude, pickedLocation.longitude);
}
```



Adeguo LocationInput per la selezione da mappa

- Il metodo `_selectOnMap` è responsabile per navigare verso la schermata della mappa e ottenere una posizione selezionata dall'utente;
- Quando viene chiamato, apre la schermata della mappa utilizzando `Navigator.of(context).push`. Questo metodo spinge una nuova route (in questo caso, la schermata della mappa) sullo stack di navigazione, che fa apparire la nuova schermata;
- Il metodo `push` restituisce un `Future` che si completerà con il valore che viene eventualmente "pop" da quella route. In questo caso, ci aspettiamo che la schermata della mappa restituisca un oggetto `LatLng` che rappresenta la posizione selezionata dall'utente;
- Se l'utente non seleziona una posizione (ad esempio, se torna indietro senza selezionare una posizione), `pickedLocation` sarà null e il metodo terminerà;
- Se l'utente seleziona una posizione, chiamiamo il metodo `_savePlace` con la latitudine e la longitudine della posizione selezionata;
- Il metodo `_savePlace` è responsabile per ottenere l'indirizzo della posizione selezionata e per salvare la posizione;
- Prima di tutto, costruisce un URL per l'API di geocodifica inversa di Google Maps, che può ottenere un indirizzo da una latitudine e una longitudine. Quindi, effettua una richiesta GET a quell'URL e decodifica la risposta JSON;
- "indirizzo viene estratto dalla risposta e viene utilizzato per creare un nuovo oggetto `PlaceLocation`, che viene poi salvato nello stato del widget;
- Infine, il metodo chiama il callback `onSelectLocation` passato al widget, che può essere utilizzato per informare un altro widget che una posizione è stata selezionata.

Il risultato



Esercizio

Il codice locationcompleto.zip contiene il progetto completo da usare. Potete provare ad esempio a salvare la lista su SQLite e le foto su file per verificare quanto imparato nelle scorse lezioni.