



# Google Maps

**N**esta aula, vamos introduzir o uso do Google Maps no Flutter. Exploraremos o que é o Google Maps, suas principais funcionalidades e benefícios. Discutiremos como o Google Maps pode ser integrado em aplicativos Flutter para fornecer mapas interativos, rotas e locais. Esta aula fornecerá uma visão geral sobre a importância e as aplicações do Google Maps nos aplicativos modernos.

## Introdução ao Google Maps no Flutter

### O que é o Google Maps?

Google Maps é um serviço de mapeamento desenvolvido pela Google que oferece imagens de satélite, mapas de ruas, vistas panorâmicas de ruas e condições de trânsito em tempo real.

### Principais Funcionalidades do Google Maps

- **Mapas Interativos:** Permite a visualização e interação com mapas.
- **Rotas e Navegação:** Fornece direções e rotas entre diferentes locais.
- **Visualização de Tráfego:** Mostra condições de trânsito em tempo real.
- **Pontos de Interesse:** Exibe locais de interesse como restaurantes, hospitais, etc.

### Benefícios de Integrar Google Maps no Flutter

- **Interatividade:** Melhora a experiência do usuário com mapas interativos.

- **Informações em Tempo Real:** Acesso a dados atualizados de tráfego e locais.
- **Funcionalidade Ampla:** Possibilidade de implementar navegação, rotas e muito mais.

Nesta aula, aprenderemos como exibir mapas usando o Google Maps no Flutter. Veremos como configurar as propriedades básicas do mapa, como inicializar o Google Maps e personalizar sua aparência. Abordaremos também a configuração das chaves de API necessárias para utilizar o Google Maps.

## Exibindo Mapas e Configurando Propriedades

### Passos para Exibir um Mapa

#### 1. Adicionar Dependências ao pubspec.yaml

*dependencies:*

flutter:

  sdk: flutter

  google\_maps\_flutter: ^2.0.6

#### 2. Obter Chaves de API do Google Maps

- Acesse o Google Cloud Console.
- Crie um projeto e ative a API do Google Maps.
- Gere uma chave de API.

#### 3. Configurar o AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
```

```
package="com.example.google_maps_app">
```

```
<application>
```

```
<meta-data
```

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

```
android:value="YOUR_API_KEY"/>
```

```
</application>
```

```
</manifest>
```

#### 4. Configurar o Info.plist para iOS

```
<key>GMSApiKey</key>
```

```
<string>YOUR_API_KEY</string>
```

#### 5. Código para Exibir o Mapa

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

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

```
void main() {
```

```
  runApp(MyApp());
```

```
}
```

```
class MyApp extends StatelessWidget {
```

```
  @override
```

```
Widget build(BuildContext context) {  
  
  return MaterialApp(  
  
    home: MapScreen(),  
  
  );  
  
}  
  
}  
  
class MapScreen extends StatefulWidget {  
  
  @override  
  
  _MapScreenState createState() => _MapScreenState();  
  
}  
  
class _MapScreenState extends State<MapScreen> {  
  
  late GoogleMapController mapController;  
  
  final LatLng _center = const LatLng(45.521563, -122.677433);  
  
  void _onMapCreated(GoogleMapController controller) {  
  
    mapController = controller;  
  
  }  
  
  @override  
  
  Widget build(BuildContext context) {  
  
    return Scaffold(  

```

```
appBar: AppBar(  
  
  title: Text('Google Maps no Flutter'),  
  
  backgroundColor: Colors.green[700],  
  
),  
  
body: GoogleMap(  
  
  onMapCreated: _onMapCreated,  
  
  initialCameraPosition: CameraPosition(  
  
    target: _center,  
  
    zoom: 11.0,  
  
  ),  
  
),  
  
);  
  
}
```

Explicação: Este código configura um mapa básico usando o Google Maps no Flutter, exibindo um mapa centrado em uma localização específica.

Nesta aula, veremos como adicionar marcadores e animações ao mapa no Flutter. Aprenderemos a criar e personalizar marcadores, adicionar animações de movimento de câmera e configurar eventos de interação com o mapa. Essas funcionalidades melhoram a experiência do usuário e tornam o mapa mais interativo.

## Marcadores e Animações no Mapa

### Adicionando Marcadores

#### 1. Adicionar um Marcador Simples

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

import 'package:google_maps_flutter/google_maps_flutter.dart';

void main() {

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: MapScreen(),

    );

  }

}

class MapScreen extends StatefulWidget {

  @override

  _MapScreenState createState() => _MapScreenState();

}
```

```
}
```

```
class _MapScreenState extends State<MapScreen> {

  late GoogleMapController mapController;

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

  final Set<Marker> _markers = {};

  void _onMapCreated(GoogleMapController controller) {

    mapController = controller;

    _markers.add(

      Marker(

        markerId: MarkerId('id-1'),

        position: _center,

        infoWindow: InfoWindow(

          title: 'Marcador Inicial',

          snippet: 'Descrição do marcador',

        ),

        icon: BitmapDescriptor.defaultMarker,

      ),

    );

    setState(() {});
```

```
}
```

```
@override
```

```
Widget build(BuildContext context) {
```

```
  return Scaffold(
```

```
    appBar: AppBar(
```

```
      title: Text('Google Maps no Flutter'),
```

```
      backgroundColor: Colors.green[700],
```

```
    ),
```

```
    body: GoogleMap(
```

```
      onMapCreated: _onMapCreated,
```

```
      initialCameraPosition: CameraPosition(
```

```
        target: _center,
```

```
        zoom: 11.0,
```

```
      ),
```

```
      markers: _markers,
```

```
    ),
```

```
  );
```

```
}
```

```
}
```



Explicação: Este código adiciona um marcador ao mapa na posição especificada, com uma janela de informações.

## Adicionando Animações de Movimento de Câmera

### 1. Animação para Nova Posição

```
void _moveCamera() {  
  
    mapController.animateCamera(  
  
        CameraUpdate.newCameraPosition(  
  
            CameraPosition(  
  
                target: LatLng(37.7749, -122.4194),  
  
                zoom: 14.0,  
  
                bearing: 45.0,  
  
                tilt: 45.0,  
  
            ),  
  
        ),  
  
    );  
  
}  
  
@override  
  
Widget build(BuildContext context) {  
  
    return Scaffold(  

```

```
appBar: AppBar(  
  
  title: Text('Google Maps no Flutter'),  
  
  backgroundColor: Colors.green[700],  
  
),  
  
body: GoogleMap(  
  
  onMapCreated: _onMapCreated,  
  
  initialCameraPosition: CameraPosition(  
  
    target: _center,  
  
    zoom: 11.0,  
  
  ),  
  
  markers: _markers,  
  
),  
  
floatingActionButton: FloatingActionButton(  
  
  onPressed: _moveCamera,  
  
  tooltip: 'Mover Câmera',  
  
  child: Icon(Icons.camera_alt),  
  
),  
  
);  
  
}
```

Explicação: Este código adiciona um botão flutuante que, ao ser pressionado, anima a câmera para uma nova posição com uma nova configuração de zoom, inclinação e direção.

Nesta aula, vamos explorar como integrar geolocalização e serviços de localização no Flutter. Aprenderemos a obter a localização atual do dispositivo, atualizar a posição em tempo real e usar serviços de localização para melhorar a funcionalidade do aplicativo. Isso é essencial para criar aplicativos que dependem de dados de localização.

## Geolocalização e Serviços de Localização

### Obtendo a Localização Atual

#### 1. Adicionar Dependências ao pubspec.yaml

*dependencies:*

flutter:

  sdk: flutter

  google\_maps\_flutter: ^2.0.6

  geolocator: ^7.7.0

#### 2. Solicitar Permissões de Localização

##### - AndroidManifest.xml

<uses-permission

  android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission

  android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

## - Info.plist

```
<key>NSLocationWhenInUseUsageDescription</key>
```

```
<string>Precisamos da sua localização para mostrar onde você está no mapa.</string>
```

## 3. Código para Obter a Localização Atual

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

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

```
import 'package:geolocator/geolocator.dart';
```

```
void main() {
```

```
  runApp(MyApp());
```

```
}
```

```
class MyApp extends StatelessWidget {
```

```
  @override
```

```
  Widget build(BuildContext context) {
```

```
    return MaterialApp(
```

```
      home: MapScreen(),
```

```
    );
```

```
  }
```

```
}
```

```
class MapScreen extends StatefulWidget {  
  
  @override  
  
  _MapScreenState createState() => _MapScreenState();  
  
}  
  
class _MapScreenState extends State<MapScreen> {  
  
  late GoogleMapController mapController;  
  
  Position? _currentPosition;  
  
  final LatLng _initialPosition = const LatLng(45.521563, -122.677433);  
  
  final Set<Marker> _markers = {};  
  
  void _onMapCreated(GoogleMapController controller) {  
  
    mapController = controller;  
  
    _getCurrentLocation();  
  
  }  
  
  _getCurrentLocation() async {  
  
    Position position = await Geolocator.getCurrentPosition(desiredAccuracy:  
LocationAccuracy.high);  
  
    setState(() {  
  
      _currentPosition = position;  
  
      mapController.animateCamera(  

```

```
CameraUpdate.newLatLng(LatLng(position.latitude,
position.longitude)),

);

_markers.add(

Marker(

  markerId: MarkerId('currentLocation'),

  position: LatLng(position.latitude, position.longitude),

  infoWindow: InfoWindow(title: 'Você está aqui'),

),

);

});

}

@override

Widget build(BuildContext context) {

  return Scaffold(

    appBar: AppBar(

      title: Text('Google Maps no Flutter'),

      backgroundColor: Colors.green[700],

    ),
```

```
body: GoogleMap(  
  
  onMapCreated: _onMapCreated,  
  
  initialCameraPosition: CameraPosition(  
  
    target: _initialPosition,  
  
    zoom: 11.0,  
  
  ),  
  
  markers: _markers,  
  
),  
  
);  
  
}  
  
}
```

Explicação: Este código obtém a localização atual do dispositivo e atualiza o mapa para exibir a posição do usuário com um marcador.

Esses exemplos e explicações fornecem uma base sólida para iniciantes em Flutter aprenderem a integrar e usar o Google Maps em seus aplicativos. Para mais detalhes, consulte a documentação oficial do Flutter: [Flutter Documentation](#).

## Materiais Extras

Você pode realizar o download do arquivo contendo os materiais extras utilizados ao longo das aulas por meio do seguinte link: <https://drive.google.com/file/d/1mg7lqMl8Pt2zl0rHlsFS0Qew00YN-sEX/view?usp=sharing>.

## Conteúdo Bônus

Para aprofundar seus conhecimentos em Desenvolvimento Mobile com integração ao Google Maps, recomendo o seguinte recurso gratuito:

Curso “Desenvolvendo Aplicações Mobile com Android Studio”: Oferecido pela Fundação Bradesco, este curso aborda desde os conceitos básicos de desenvolvimento mobile até a criação de projetos no Android Studio, incluindo a utilização de layouts e requisições HTTP.

## Referências Bibliográficas

BOYLESTAD, R. L.; NASHELSKY, L. Dispositivos Eletrônicos e Teoria de Circuitos. 11. ed. Pearson, 2013.

DEITEL, P. J.; DEITEL, H. M. Ajax, Rich Internet Applications e Desenvolvimento Web para Programadores. Pearson, 2008.

DUARTE, W. Delphi para Android e iOS: Desenvolvendo Aplicativos Móveis. Brasport, 2015.

FELIX, R.; SILVA, E. L. da. Arquitetura para Computação Móvel. 2. ed. Pearson, 2019.

LEE, V.; SCHNEIDER, H.; SCHELL, R. Aplicações Móveis: Arquitetura, Projeto e Desenvolvimento. Pearson, 2005.

MARINHO, A. L.; CRUZ, J. L. da. Desenvolvimento de Aplicações para Internet. 2. ed. Pearson, 2019.

MOLETTA, A. Você na Tela: Criação Audiovisual para a Internet. Summus, 2019.

SILVA, D. (Org.) Desenvolvimento para dispositivos móveis. Pearson, 2017.



**Ir para exercício**