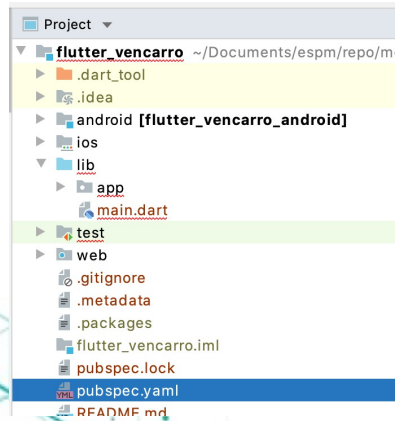
PM – PROGRAMAÇÃO MOBILE

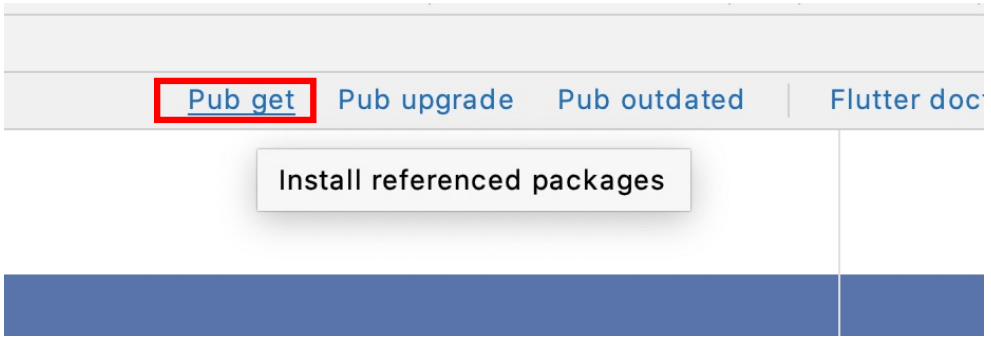
FLUTTER VENCARRO

edson.sensato@espm.br

Criando a Splashscreen




```
dependencies:
  flutter:
    sdk: flutter
  splashscreen:
```



Criando a Splashscreen

```
import 'package:flutter/material.dart';
import 'package:flutter_vencarro/app/ListaVeiculo.dart';
import 'package:splashscreen/splashscreen.dart';

class Splash extends StatelessWidget {

  @override
  Widget build(BuildContext context) {

    return MaterialApp(home: SplashScreen(seconds: 14,
        title: Text('Vencarro', style: new TextStyle(
            fontWeight: FontWeight.bold,
            fontSize: 20.0
        )),
        backgroundColor: Colors.white,
        image: Image.network('https://cdn.pixabay.com/photo/2012/04/12/19/39/volga-30332_1280.png'),
        navigateAfterSeconds: ListaVeiculo(),
      photoSize: 100.0,
    ));
  }
}
```

flutter run --no-sound-null-safety

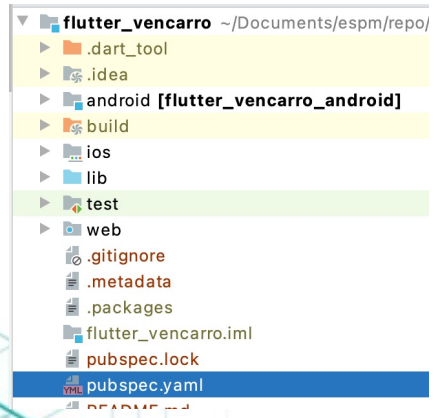
ListView

```
final List<String> itens = <String>['Item 1', 'Item 2', 'Item 3'];

@override
Widget build(BuildContext context) {

  return MaterialApp(home: Scaffold(
      appBar: AppBar(title: Text("Vencarro")),
      body:Center(child:
      ListView.builder(
          padding: const EdgeInsets.all(8),
          itemCount: itens.length,
          itemBuilder: (context, index) {
            return ListTile(
                title: Text('${itens[index]}'),
                onTap: () {
                  print(itens[index]);
                }
            );
          })
  )));
}
```

Requisições HTTP




```
dependencies:
  flutter:
    sdk: flutter
  http:
  splashscreen:
```

```
import 'package:http/http.dart' as http;


final URL = 'https://parallelum.com.br/fipe/api/v1/carros/marcas';
http.get(Uri.parse(URL)).then((resposta) => print (resposta.body));
```

JSON Decode

```dart
import 'dart:convert';


final JSON_EX = '{"nome": "Joao", "nota": 10}';
final json_object = jsonDecode(JSON_EX);
print(json_object);
print(json_object["nome"]);
print(json_object["nota"]);
```

Criando Veiculo

```
class Veiculo {

  var marcaid;
  var marca;
  var modeloid;
  var modelo;
  var idAno;
  var ano;
  var preco;

  Veiculo({this.marcaid,
           this.marca,
           this.modeloid = 0,
           this.modelo = "",
           this.idAno = 0,
           this.ano = "",
           this.preco = ""});
}
```

Carregando Marcas

```
List<Veiculo> veiculos = List<Veiculo>.empty(growable: true);

@override
void initState() {
  super.initState();

  final URL = 'https://parallelum.com.br/fipe/api/v1/carros/marcas';

  http.get(Uri.parse(URL)).then((resposta) {

    jsonDecode(resposta.body).forEach((item) {
      setState(() {
        veiculos.add(Veiculo(marca: item["fipe_name"], marcaid: item["id"]));
      });
    });

  }
  );
}
```

Custom ListTile

```
class CustomTile extends StatelessWidget {

  var marca = "";
  var modelo = "";

  CustomTile({this.marca, this.modelo});

  @override
  Widget build(BuildContext context) {

    return Container(child: Column(children: [Text(marca), Text(modelo)]));

  }
}
```

```
return ListTile(
    title: CustomTile(marca: '${veiculos[index].marca}', modelo: '${veiculos[index].modelo}')}
    ...
);
```

Tela Resumo

Criar uma variável de estado para armazenar o caminho da imagem obtida na foto

Utilizar **`Column`** e **`Rows`** para exibir marca, modelo, ano e preço

Cada elemento da **`Row`** deve ser encapsulado em um **`Container`** com as propriedades **`padding`** (**`EdgeInsets`**) e **`width`** definidas (100.0)

A imagem inicial (**`Image`**) que será substituída pela foto deve ser encapsulada em um **`Flexible`**:

```
Flexible(child: Image.file(File(caminho_imagem)))
```

Navegação

Incluir rotas na **main.dart**

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

  return MaterialApp(

      routes: {'/resumo': (context) => Resumo()}

  ));
}
```

Navegação – push / pop

`Navigator.of(context).pushNamed('/resumo')` → abre uma nova tela identificada pela rota

`Navigator.of(context).pop()` → fecha tela atual

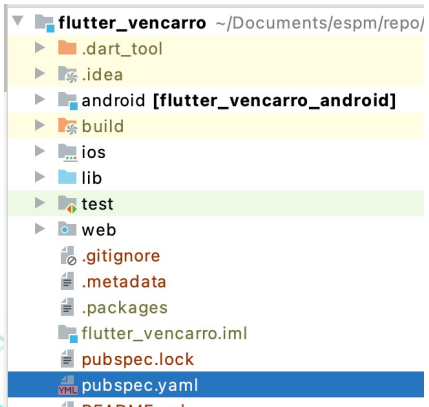
`Navigator.of(context).pushNamed('/resumo', arguments: {"nome": "joao"})` → passa parâmetros

`ModalRoute.of(context).settings.arguments` → obtém parâmetros passados

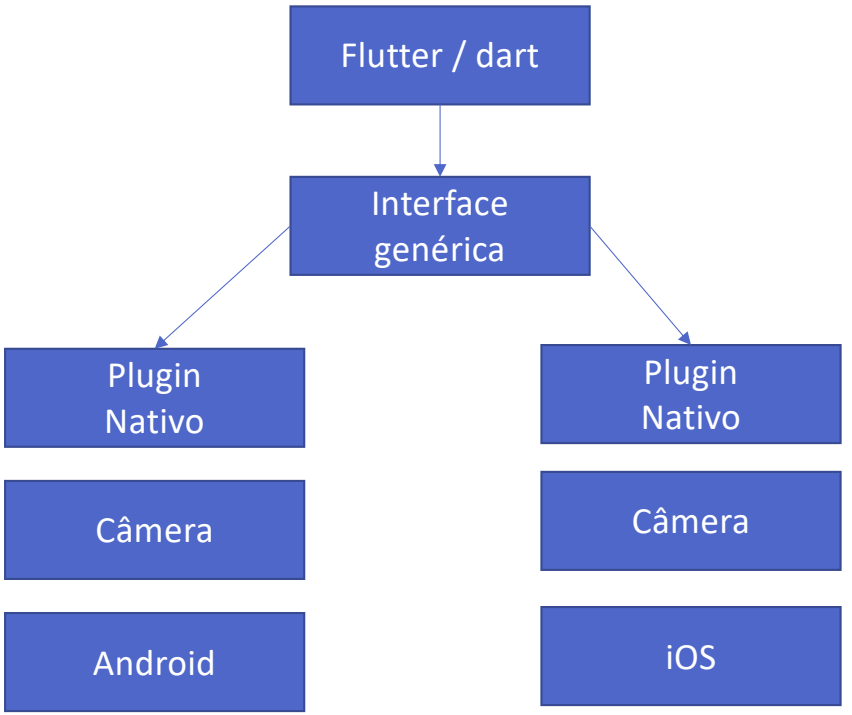
`Navigator.of(context).pop("Valor de retorno")` → fecha e retorna valores

`var ret = await Navigator.of(context).pushNamed('/resumo')` → recebe valores de volta

Câmera




```
dependencies:
  flutter:
    sdk: flutter
  http: 0.13.3
  splashscreen:
  camera:
  path_provider:
  path:
```



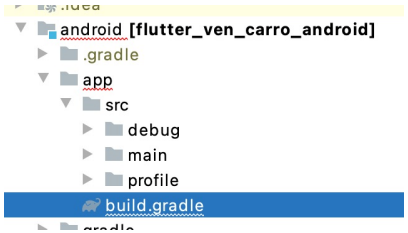
Câmera

```
var camera;

Future<void> main() async {
  runApp(Splash());

  WidgetsFlutterBinding.ensureInitialized();
  final cameras = await availableCameras();
  camera = cameras.first;

}
```



minSdkVersion 21

Câmera

Criar uma classe **Camera** do tipo **StatefullWidget**

```
late CameraController _controller;
late Future<void> _initializeControllerFuture;

@override
void initState() {
  super.initState();
  _controller = CameraController(camera, ResolutionPreset.medium);
  _initializeControllerFuture = _controller.initialize();
}
```

Câmera

```
body: FutureBuilder<void>(
  future: _initializeControllerFuture,
  builder: (context, snapshot) {
    if (snapshot.connectionState == ConnectionState.done) {
      return CameraPreview(_controller);
    } else {
      return Center(child: CircularProgressIndicator());
    }
  },
)
```

Câmera

Criar um botão para capturar a imagem (**floatingActionButton**) e retornar o caminho da imagem capturada (**image.path**) para a tela de resumo:

```
floatingActionButton: FloatingActionButton(
  child: Icon(Icons.camera_alt),
  onPressed: () async {
    // aguarda até que o controle da câmera seja iniciado
    await _initializeControllerFuture;
    // aguarda até que a imagem seja obtida...
    final image = await _controller.takePicture();
    // retorna para a tela de resumo com o caminho da imagem obtida
    Navigator.of(context).pop(image.path);
  }
)
```

Câmera

onPressed abaixo deve ser implementado na classe de resumo para acionar a câmera:

```
persistentFooterButtons: [IconButton(icon: Icon(Icons.camera_alt), onPressed: () async {

  // aguarda até que a imagem seja obtida...
  var caminho_imagem = await Navigator.of(context).pushNamed('/camera');
  // atualiza a variáveis de estado caminho_imagem para que a imagem seja visualizada na tela
  setState(() {

    this.caminho_imagem = caminho_imagem;

  });
})
```

Mapa

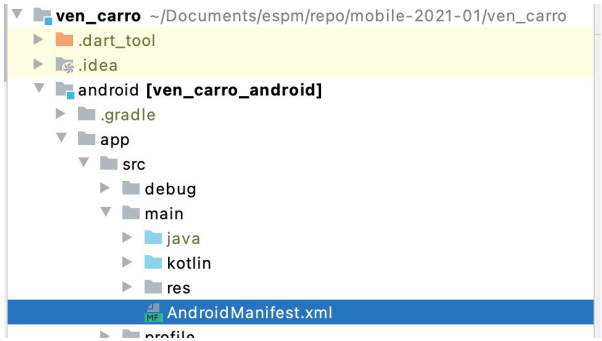
Para utilizar o serviço de mapas é necessário cadastrar uma chave de API junto à Google:

https://console.developers.google.com

Uma vez obtida a chave ela deve ser inserida no **`AndroidManifest.xml`**

```
<application
    android:label="ven_carro"
    android:icon="@mipmap/ic_launcher">

   <meta-data android:name="com.google.android.geo.API_KEY"
       android:value="<<CHAVE_AQUI>>"/>
...
</application>
```



Mapa

Criar um **`StateafulWidget`** para apresentar o mapa

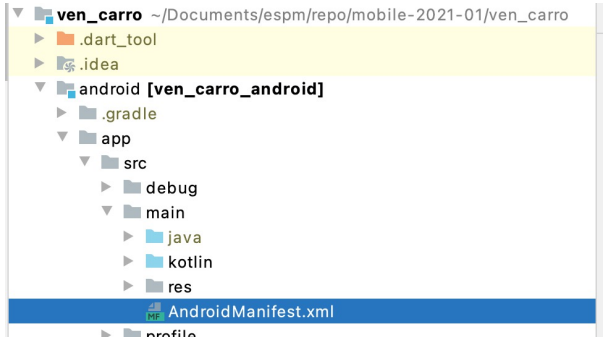
```
// Aguarda até que o controle da câmera seja obtido
Completer<GoogleMapController> controller = Completer();

static final CameraPosition inicio = CameraPosition(target: LatLng(0, 0), zoom: 10.0);
```

Iniciando o Map Controller

```
Widget build(BuildContext context) {
  return new Scaffold(
    body: GoogleMap(
      mapType: MapType.normal,
      initialCameraPosition: inicio,
      // Mapa criado e controle instanciado!
      onMapCreated: (GoogleMapController controller) {
       controller.complete(controller);
      }
    )
  );
}
```

Obter Posição Atual




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

```
posicaoAtual() async {
  Geolocator
      .getCurrentPosition(desiredAccuracy: LocationAccuracy.best, forceAndroidLocationManager: true)
      .then((Position position) async {
        print(position);
      }
}
```

Reposicionando Câmera

```
posicaoAtual() async {
  Geolocator
      .getCurrentPosition(desiredAccuracy: LocationAccuracy.best, forceAndroidLocationManager: true)
      .then((Position position) async {
    print(position);

    final GoogleMapController controller = await controller.future;
    CameraPosition atual = CameraPosition(
      target: LatLng(position.latitude, position.longitude),
      zoom: 10.0);
    controller.animateCamera(CameraUpdate.newCameraPosition(atual));
}
}
```

Adicionando Marcadores

Criar uma variável de estado para armazenar os marcadores (**Marker**)

```
var maracadores = Set<Marker>();
```

Adicionar propriedade **markers** junto ao *Widget* **GoogleMap** referenciando a variável **marcadores**

Criar os marcadores e adicioná-los à variável de estado **marcadores** (no exemplo, colocar em **posicaoAtual**)

```
var mAtual = Marker(infoWindow: InfoWindow(title: "Local Atual", snippet: "Sua localização atual"),
    markerId: MarkerId("ATUAL"),
    position: LatLng(position.latitude, position.longitude));

setState(() {
  maracadores.add(mAtual);
});
```

Adicionando Marcadores

É possível também adicionar marcadores ao clicar sobre o mapa implementando **onTap**:

```
onTap: (latlang) {

  var mAtual = Marker(infoWindow: InfoWindow(title: "Local Atual",
      snippet: "Sua localização atual"),
      markerId: MarkerId("ATUAL"),
      position: LatLng(latlang.latitude, latlang.longitude));

  setState(() {
    maracadores.add(mAtual);
  });
}
```