VOLTAR

Projeto de um app consumindo JSON de uma API real

Nessa aula vamos aprender a criar um aplicativo que irá obter dados da internet, através de uma consulta a um serviço aberto de temperaturas de cidades. Vamos aprender muitas coisas nessa aula, como nos cadastrar no site que usaremos, entender o que é um JSON, e montar a interface completa de nossa aplicação, com consultas assíncronas.

NESTE TÓPICO

Marcar tópico

O que são: APIs e JSON?

Sistemas diferentes, desenvolvidos por fabricantes distintos e em linguagens totalmente avessas precisam se comunicar de alguma forma. Uma das melhores maneiras de fazer isso é utilizando textos. Isso mesmo, textos, assim como nós seres humanos, de nações diferentes podemos nos comunicar com textos (que podem ser traduzidos) sistemas distintos e que "não se conhecem" também podem.

Para que isso ocorra existe um padrão de comunicação universal, que é como um texto que pode ser interpretado em praticamente qualquer linguagem de programação moderna e, claro, o Dart com Flutter não são diferentes. São os famosos JSON.

JSON é acrônimo de *Javascript Object Notation* (Notação de Objetos Javascript) e apesar de ter em seu nome o termo Javascrip não precisaremos nos preocupar com essa linguagem. É que em Javascript os objetos são representados no mesmo padrão de um JSON e isso acabou se tornando um modelo universal.

Mas o que é um JSON? É um arquivo de texto que pode ser gerado por uma aplicação com informações para que podem ser utilizadas por outras aplicações, assim como você pode também enviar um JSON para que a aplicação alvo receba dados para tratamento e armazenamento, por exemplo. O conceito é realmente simples mesmo: Um simples arquivo de texto, transmitido via web que pode ser lido ou recebido por uma aplicação qualquer.

E para consigamos acessar essas informações devemos utilizar uma chamada web para a aplicação alvo. Isso normalmente é feito através de uma API, que é acrônimo de "Application Programming Interface" ou, ainda, "Interface de Programação de Aplicações" e são nada mais do que trechos de código expostos na web capazes de gerar um JSON.

MAS POR QUE ESTAMOS FALANDO DISSO TUDO?

Bom, nessa aula vamos desenvolver uma pequena aplicação de previsão do tempo que recupera os dados do clima através da chamada de uma API pública de um serviço gratuito e tal chamada é retornada para a aplicação em formato JSON.

Para entender um pouco melhor o que é uma API e como é feita essa resposta em formato JSON, veja o vídeo abaixo que mostra exatamente o projeto que desenvolveremos nessa aula e ensina como se prepara para utilizar a API do "openweathermap" (mapa de clima aberto). Muita atenção no vídeo inicial pois seu cadastro é primordial neste site, além claro é muito importante que você tenha sua própia chave de API.

Criando e configurando o projeto

Neste projeto vamos utilizar vários recursos novos como pacotes externos que precisaremos instalar em nossa aplicação, dividiremos o projeto em arquivos específicos (cada um com sua responsabilidade) etc. Temos bastante trabalho pela frente. Para isso, o vídeo abaixo mostra a criação e início do projeto.

Como você pode observar no vídeo acima, estamos criando este projeto já aplicando alguns conceitos mais interessantes como tema para a aplicação e dividindo nossos arquivos em pastas, baseado na responsabilidade de cada um dentro do projeto, para que nosso projeto fique, inclusive, muito mais organizado.

Criando a listagem e seleção de cidades

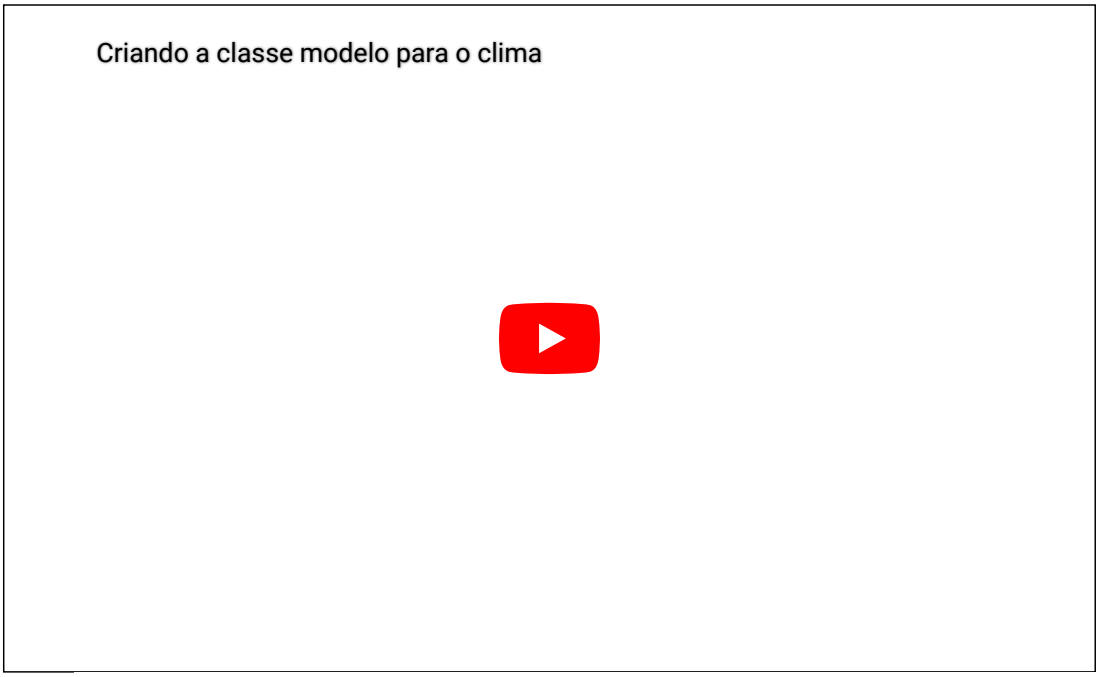
Agora que nosso projeto já está devidamente configurado, vamos a desenvolve a tela principal. Para isso, nada melhor do que começar com a caixa de seleção da cidade, que utiliza um componente externo e precisa ser "instalado" no projeto, o que é realmente muito simples no Flutter.

Para desenvolver instalar e desenvolver usando este componente, veja o vídeo abaixo que inclui, ainda, uma pequena correção no tema da aplicação, do vídeo anterior.

Criando a classe de Modelo

Agora que já temos uma parte da aplicação pronta, podemos começar a montar a lógica da obtenção dos dados através da API. Para isso precisamos ter uma classe que representará a previsão do tempo, ou seja, um objeto que terá seus valores atribuídos através da consulta à API externa.

Para montarmos essa classe nada melhor do que um vídeo a explicando, que pode ser visto abaixo:



Este projeto está ficando muito legal não é mesmo? Esperamos realmente que você esteja conseguindo fazer tudo direitinho.

Widget de tempo

Agora que já temos nossa classe de modelo, podemos começar a desenvolver nosso Widget que mostrará as informações do clima.

Para isso, acompanhe o vídeo abaixo:

Criando o Widget de clima

Obtenção dos dados da API externa

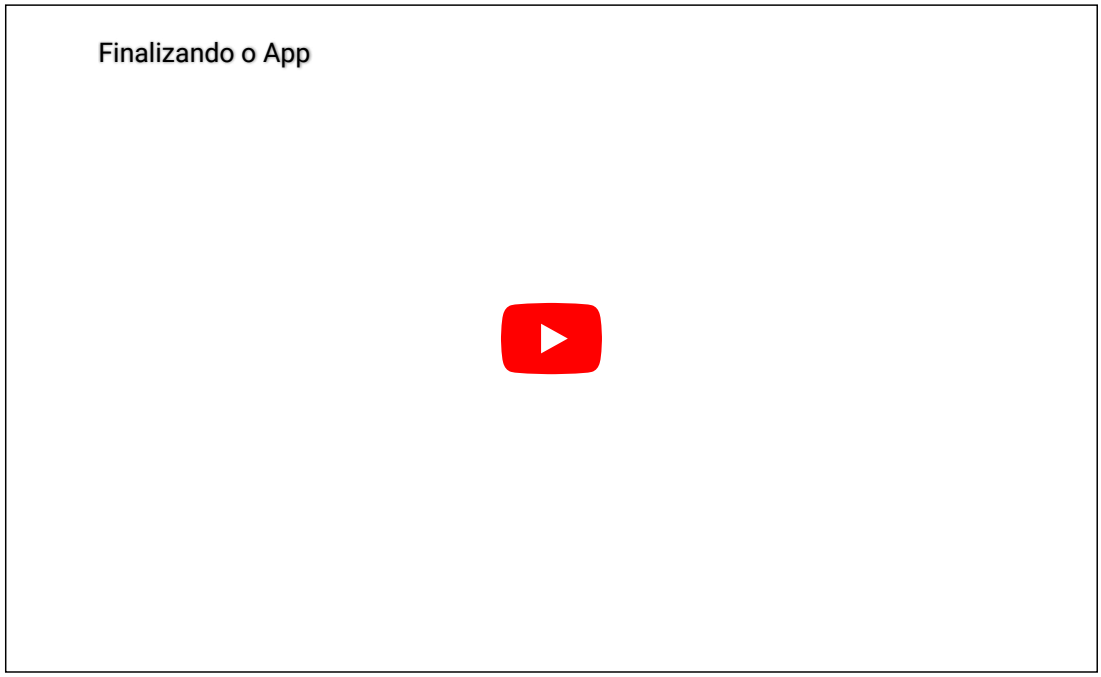
Agora que a estrutura do Widget de tempo já está prontinha, vamos começar a fazer o método que será responsável por obter os dados do clima, através do retorno em JSON da API que usamos no início do projeto.

Para isso, não deixe de assistir o vídeo abaixo.

Criando o método de obtenção dos dados da API

Finalizando o App

Esperamos que você esteja bastante empolgado(a) com essa parte: A finalização do aplicativo! Com o vídeo abaixo você verá como finalizar a construção deste aplicativo, já usando o Widget que construímos anteriormente.



Prontinho! Terminamos nosso App de clima. Se você precisar consultar os códigos, não deixe de ver os blocos abaixo que mostram, separadamente, a codificação de cada uma das classes, ok?

Arquivo "main.dart":

```
import 'package:flutter/material.dart';
import 'package:tempo_app/screens/home.dart';
import 'package:tempo_app/theme/theme.dart';

void main() {
  runApp(PrevisaoTempo());
}

class PrevisaoTempo extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Home(),
      title: 'Previsão do Tempo',
      debugShowCheckedModeBanner: false,
      theme: lightTheme(),
      darkTheme: darkTheme(),
      themeMode: ThemeMode.system,
    );
  }
}
```

Arquivo /theme/theme.dart

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

ThemeData darkTheme() {
  return ThemeData(
    brightness: Brightness.dark,
    scaffoldBackgroundColor: Colors.grey[800],
    textTheme: TextTheme(
      headline4: TextStyle(fontSize: 23.0, fontWeight: FontWeight.bold),
      headline5: TextStyle(fontSize: 23.0, fontWeight: FontWeight.w200),
    )
  );
}

ThemeData lightTheme() {
  return ThemeData(
      brightness: Brightness.light,
      scaffoldBackgroundColor: Colors.white,
      appBarTheme: AppBarTheme(color: Colors.deepPurple[700]),
      textTheme: TextTheme(
        headline4: TextStyle(fontSize: 23.0, fontWeight: FontWeight.bold),
        headline5: TextStyle(fontSize: 23.0, fontWeight: FontWeight.w200),
      )
  );
}
```

Arquivo model/tempoModel.dart

```dart
class TempoData {
  final double temp;
  final double tempMax;
  final double tempMin;
  final String descTemp;
  final String icone;
  final int umidade;

  TempoData(
      {this.temp,
      this.tempMax,
      this.tempMin,
      this.descTemp,
      this.icone,
      this.umidade});

  factory TempoData.fromJson(Map<String, dynamic> json) {

    String capitalize(String s) => s[0].toUpperCase() + s.substring(1);

    return TempoData(
      temp: json['main']['temp'].toDouble(),
      tempMax: json['main']['temp_max'].toDouble(),
      tempMin: json['main']['temp_min'].toDouble(),
      descTemp: capitalize(json['weather'][0]['description']),
      icone: json['weather'][0]['icon'],
      umidade: json['main']['humidity'].toInt()
    );
  }
}
```

Arquivo widgets/tempo_widget.dart:

```dart
import 'package:flutter/material.dart';
import 'package:tempo_app/model/tempoModel.dart';


class Tempo extends StatelessWidget {

  final TempoData temperatura;

  Tempo({Key key, @required this.temperatura}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Image.network(
          'http://openweathermap.org/img/wn/${temperatura.icone}.png',
          fit: BoxFit.fill,
          width: 80.0,
        ),
        Text(
          '${temperatura.temp.toStringAsFixed(0)} ºC',
          style: TextStyle(fontSize: 50.0),
        ),
        Text(
          temperatura.descTemp,
          style: TextStyle(fontSize: 30.0),
          textAlign: TextAlign.center,
        ),
        SizedBox(
          height: 20.0,
        ),
        Text(
          "Min. do dia: ${temperatura.tempMin.toStringAsFixed(0)}",
          style: TextStyle(fontSize: 16.0),
          textAlign: TextAlign.center,
        ),
        Text(
          "Max. do dia: ${temperatura.tempMax.toStringAsFixed(0)}",
          style: TextStyle(fontSize: 16.0),
          textAlign: TextAlign.center,
        ),
        Text(
          "Umidade do ar: ${temperatura.umidade.toString()}%"
        )
      ],
    );
  }
}
```

Arquivo screens/home.dart:

```dart
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:searchable_dropdown/searchable_dropdown.dart';
import 'package:http/http.dart' as http;
import 'package:tempo_app/model/tempoModel.dart';
import 'package:tempo_app/widgets/tempo_widget.dart';

class Home extends StatefulWidget {
  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State<Home> {
  bool isLoading = false;
  TempoData tempoData;

  List<String> _cidades = [
    'Aracaju',
    'Belém',
    'Belo Horizonte',
    'Boa Vista',
    'Brasilia',
    'Campo Grande',
    'Cuiaba',
    'Curitiba',
    'Florianópolis',
    'Fortaleza',
    'Goiânia',
    'João Pessoa',
    'Macapá',
    'Maceió',
    'Manaus',
    'Natal',
    'Palmas',
    'Porto Alegre',
    'Porto Velho',
    'Recife',
    'Rio Branco',
    'Rio de Janeiro',
    'Salvador',
    'São Luiz',
    'São Paulo',
    'Teresina',
    'Vitoria'
  ];

  String _cidadeSelecionada = "São Paulo";

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

  carregaTempo() async {
    setState(() {
      isLoading = true;
    });

    String appid = 'XXXX'; //Coloque a sua chave de api aqui!
    String lang = 'pt_br';
    String units = 'metric';

    final tempoReponse = await http.get(
        'https://api.openweathermap.org/data/2.5/weather?q=$_cidadeSelecionada&appid
=$appid&units=$units&lang=$lang');

    print('Url montada: ' + tempoReponse.request.url.toString());

    if (tempoReponse.statusCode == 200) {
```

```
      return setState(() {
        tempoData = TempoData.fromJson(jsonDecode(tempoReponse.body));
        isLoading = false;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(_cidadeSelecionada),
        centerTitle: true,
      ),
      body: Center(
        child: Column(
          children: [
            SearchableDropdown.single(
              items: _cidades
                  .map((e) => DropdownMenuItem(value: e, child: Text(e)))
                  .toList(),
              onChanged: (value) {
                setState(() {
                  _cidadeSelecionada = value;
                  carregaTempo();
                });
              },
              displayClearIcon: false,
              value: _cidadeSelecionada,
              icon: Icon(Icons.location_on),
              isExpanded: true,
              closeButton: "Fechar",
            ),
            Expanded(
                child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Padding(
                  padding: EdgeInsets.all(6.0),
                  child: isLoading
                      ? CircularProgressIndicator(
                          strokeWidth: 4.0,
                          valueColor: new AlwaysStoppedAnimation(Colors.blue),
                        )
                      : tempoData != null
                          ? Tempo(temperatura: tempoData)
                          : Container(
                              child: Text(
                                "Sem dados para exibir!",
                                style: Theme.of(context).textTheme.headline4,
                              ),
                            ),
                ),
                Padding(
                  padding: EdgeInsets.all(8.0),
                  child: isLoading
                      ? Container(
                          child: Text(
                          "Carregando...",
                          style: Theme.of(context).textTheme.headline5,
                        ))
                      : IconButton(
                          icon: Icon(Icons.refresh),
                          iconSize: 40.0,
                          tooltip: 'Recarregar',
                          onPressed: carregaTempo,
                          color: Colors.blue,
                        ),
                )
```

```
140.                 ],
141.               ))
142.             ],
143.           ),
144.         ),
145.       );
146.     }
147.   }
```

E também, se você precisar, poderá baixar o projeto completo finalizado abaixo.

MATERIAL COMPLEMENTAR

(Https://Img.Uninove.Br/Static/0/0/0/0/0/0/9/0/0/0/2/9000266/Tempo-App.7z)

Quiz

Exercício Final

Projeto de um app consumindo JSON de uma API real

INICIAR

Referências

DART. **Dart documentation**. *Site*. Disponível em: https://dart.dev/. Acesso em: 08 dez. 2020.

MATERIAL DESIGN. **Material Design documentation**. *Site*. Disponível em: https://material.io/. Acesso em: 08 dez. 2020.

FLUTTER. **Flutter docs.** *Site*. Disponível em: https://flutter.dev. Acesso em: 09 dez. 2020.

WINDMILL, Eric. **Flutter in action**. Nova Iorque: Manning publications, 2020. *E-book.* Disponível em: https://learning.oreilly.com/library/view/flutter-in-action/9781617296147/. Acesso em: 08 dez. 2020.

SINHA, Sanjib . **Quick start guide to Dart programming**: create high performance applications for the web and mobile. Lompoc, CA, EUA: Apress, 2019. *E-book*. Disponível em: https://learning.oreilly.com/library/view/quick-start-guide/9781484255629/. Acesso em: 09 dez. 2020.

ALESSANDRIA, Simone. **Flutter projects**: a practical, project-based guide to building real-words cross-platform mobile applications and games. Birmingham, Reino Unido: Packt Publishing, 2020. *E-book*. Disponível em: https://learning.oreilly.com/library/view/flutter-projects/9781838647773/. Acesso em: 09 dez. 2020.

ZACCAGNINO, Carmine. **Programming Flutter**. [s.l.]: The Pragmatic Bookshelf, 2020. *E-book*. Disponível em: https://learning.oreilly.com/library/view/programming-flutter/9781680507621/. Acesso em: 09 dez. 2020.

Avalie este tópico

ANTERIOR
Primeiro projeto completo com FlutterPrimeiro projeto completo com Flutter

Biblioteca (https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove (http://www.uninove.br)
Mapa do Site

Índice

PRÓXIMO
Projeto de um App com armazenamento Interno - ToDo

Ajuda? (https://ava.uninove.br/seu/AVA/topico/topico.php?idCurso=)