Biblioteca DIO

Como vimos no tópico anterior, o protocolo *HTTP* e seus verbos são essenciais para o uso de serviços *web*, sendo assim necessitamos de uma implementação em Dart que possa nos permitir realizar requisições *HTTP*, e é aí que entra a biblioteca *Dio*. Ela nos permitirá realizar nossas requisições *HTTP*.

Para que possamos utilizar essa biblioteca em nossos projetos Flutter, primeiramente necessitamos informá-la como uma dependência de nosso projeto. Para tal, devemos informar mais uma dependência em nosso arquivo *pubspec.yaml*. Veja um exemplo abaixo:

Agora que já informamos nossa mais nova dependência, podemos declarar nosso código Dart para realizar a requisição e receber a resposta de nosso web service. Para tal, vamos realizar uma consulta a uma API pública de CEPs, o Postmon (https://postmon.com.br/).

Para nosso exemplo enviaremos um CEP exemplo para a *API* do Postmon. O Cep será 01001000. E com base nele, a API nos fornecerá uma resposta *JSON* contendo os dados de tal CEP, entre eles teremos:

- Rua;
- Bairro;
- Cidade;
- E etc.

Agora que já entendemos como funcionará nosso exemplo prático, vamos ao código-

fonte para tal. Na figura 33 encontramos um exemplo de código em Dart que nos fornece além dos dados já citados, o código do status da requisição.

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

void main() async {
  var url = "https://api.postmon.com.br/v1/cep/01001000";
  var resposta = await Dio().get(url);
  print(resposta.data);
}

```

Figura - Exemplo de requisição HTTP

Como pode ter ficado claro no exemplo anterior, uma requisição *HTTP* é feita de forma assíncrona, retornando um resultado *Future*. Sendo assim, é necessário que utilizemos os conceitos aprendidos no módulo anterior para que possamos executar nosso código, já que sempre que realizarmos uma requisição, a mesma não terá uma resposta imediata.

Outro aspecto que vale a pena ressaltar é o fato da biblioteca *Dio* possuir métodos para requisições para todos os demais verbos *HTTP*.

Tratando os dados

Agora que já entendemos como requisitar informações de um serviço *web*, precisamos entender como tratar esses dados, afinal, *JSON* não é um formato suportado pela Dart, sendo assim, o Dart trata o *JSON* como uma *String*. Felizmente a biblioteca *Dio* já faz todo trabalho pesado para nós, convertendo o *JSON* em um objeto *Map*. Sendo assim, o tratamento do *JSON* é feito de forma automática, nos restando apenas armazenar esse mapa para uso posterior. Vejamos um exemplo na figura abaixo.

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

void main() async {
  var url = "https://api.postmon.com.br/v1/cep/01001000";
  Response<dynamic> resposta = await Dio().get(url);
  Map<String,dynamic> mapa = resposta.data;
  print("Chaves: ${mapa.keys}");
  print("Chaves: ${mapa.values}");
}

```

FutureBuilder

O *Widget FutureBuilder*, como o próprio nome já sugere, irá construir um *Widget* em um futuro, ou seja, após receber a resposta de uma função cujo retorno é *Future*, um determinado *Widget* será "bildado" na tela do dispositivo. Para que fique mais fácil de compreender, vamos a um exemplo prático.

Função buscarEndereco

```dart
Future<Map> buscarEndereco(String cep) async {
    var url = Uri.parse('https://api.postmon.com.br/v1/cep/$cep');
    var resposta = await http.get(url);
    return json.decode(resposta.body);
  }
```

Main.dart - Método Build

```dart
Widget build(BuildContext context) {
    return FutureBuilder(
      future: buscarEndereco("01001000"),
      builder: (BuildContext context, AsyncSnapshot snapshot) {
        switch (snapshot.connectionState) {
          // Requisição não iniciada
          case ConnectionState.none:
          // Aguardando resposta
          case ConnectionState.waiting:
            return Scaffold(
              body: Center(
                child: CircularProgressIndicator(),
              ),
            );
          default:
            return MainScreen(snapshot.data);
        }
      },
    );
  }
```

MainScreen.dart

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

class MainScreen extends StatelessWidget {
  final Map<String, dynamic> dados;
  const MainScreen(this.dados);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Aplicação exemplo"),
      ),
      body: Container(
        margin: EdgeInsets.all(16),
        child: ListTile(
          leading: Icon(
            Icons.place,
          ),
          title: Text(
            dados["logradouro"],
          ),
          subtitle: Text(
            "${dados["cidade"]} - ${dados["estado"]}",
          ),
        ),
      ),
    );
  }
}

```