Flutter: Http

Rogério B. de Andrade

Método "GET" para realizar requisições Web

O método GET é uma das maneiras mais comuns de enviar e receber dados em aplicativos.

Visão geral do método GET

- O método GET é um dos métodos HTTP utilizados para solicitar recursos em uma aplicação;
- É utilizado para recuperar informações de um servidor, como dados de formulários ou parâmetros de URL;
- A principal característica do método GET é que os dados são enviados como parte da URL.

Método "GET" para realizar requisições Web

Para realizar requisições HTTP em Flutter, é necessário adicionar a dependência **http** ao arquivo **pubspec.yaml**.

Abra o arquivo **pubspec.yaml** em seu projeto Flutter e adicione a seguinte dependência:

```
dependencies:
  http: ^0.13.0
```

Método "GET" para realizar requisições Web

Após adicionar a dependência, importe a biblioteca HTTP no arquivo Dart em que deseja realizar a requisição.

No topo do arquivo, adicione o seguinte import:

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

Método "GET" para realizar requisições Web

```dart
// realiza a requisição
final response = await http.get(Uri.parse('https://demo4107708.mockable.io/data'));
// verifica êxito da requisição
if (response.statusCode == 200) {
  // converte resposta em objeto json
  final jsonResponse = json.decode(response.body);
  // atualiza state
  setState(() {
    dataList = jsonResponse['data'];
  });
} else {
  // erro na requisição
  print('Request failed with status: ${response.statusCode}.');
}
```

Future

No contexto do Flutter, para fazer uma requisição GET no futuro, você pode usar a classe **Future** juntamente com a biblioteca **http** para realizar chamadas HTTP.

O **Future** é uma construção que permite trabalhar com valores que podem não estar disponíveis imediatamente, como no caso de requisições assíncronas.

Future

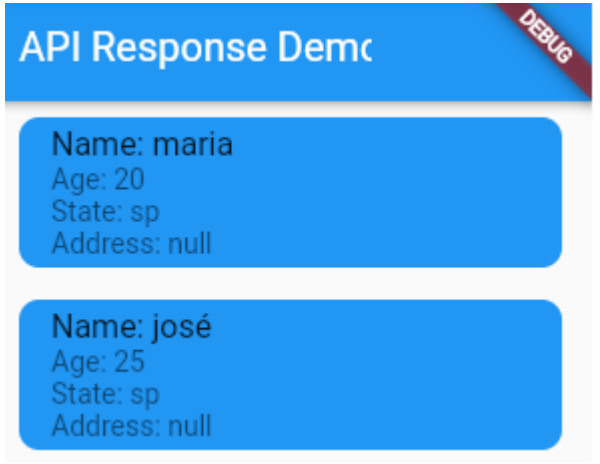
```dart
class MyAppState extends State<MyApp> {
  List<dynamic> dataList = [];
  // método assíncrono para consumir informações de uma api
  Future<void> fetchData() async {
    // realiza a requisição
    final response = await http.get(Uri.parse('https://demo4107708.mockable.io/data'));
    // verifica êxito da requisição
    if (response.statusCode == 200) {
      // converte resposta em objeto json
      final jsonResponse = json.decode(response.body);
      // atualiza state
      setState(() {
        dataList = jsonResponse['data'];
      });
    } else {
      // erro na requisição
      print('Request failed with status: ${response.statusCode}.');
    }
  }
```

EXECUTA DE FORMA ASSÍNCRONA O MÉTODO GET

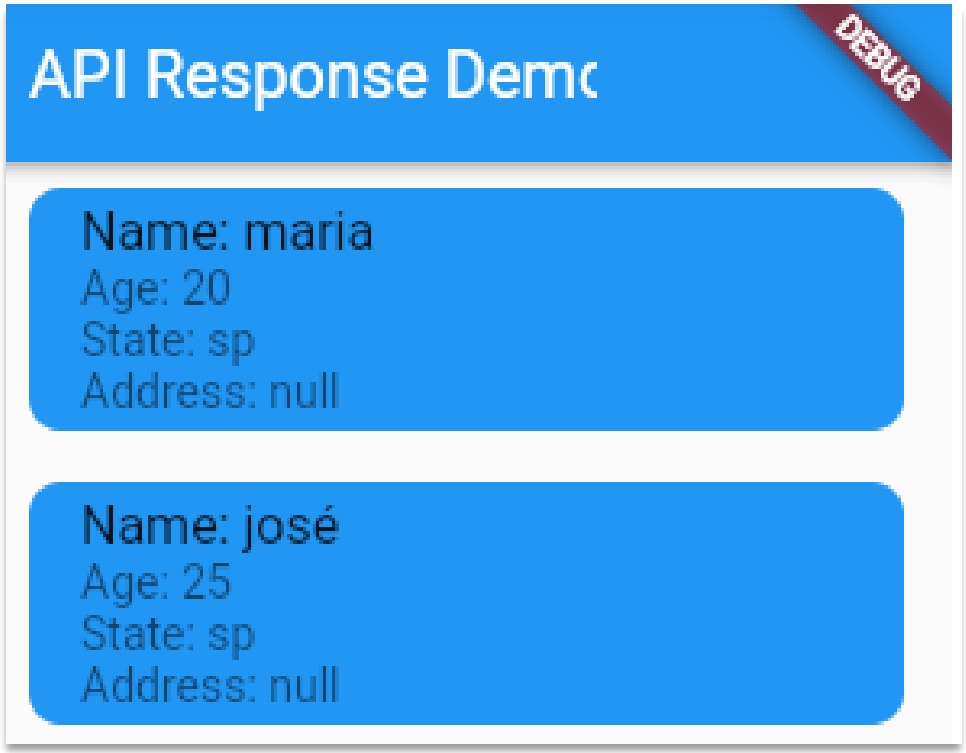
AGUARDA RESPOSTA DO MÉTODO GET

Renderização no ListView

```dart
body: ListView.builder( Build para renderizar o listView
  itemCount: dataList.length, Obtém número de itens da lista (dataList)
  itemBuilder: (BuildContext context, int index) { Acesso ao item listView
    final item = dataList[index]; item atual a ser renderizado
    return Container( Container para renderizar retângulo arrendondado
        margin: Margem
            const EdgeInsets.symmetric(vertical: 8.0, horizontal: 16.0),
        decoration: BoxDecoration( Configuração do retângulo (Box)
          color: Colors.blue, // Cor de fundo do retângulo
          borderRadius: BorderRadius.circular(10.0), // Raio
        ),
        child: ListTile( Renderização das informações da lista
          title: Text('Name: ${item['name']}'), Título
          subtitle: Column( Subtítulo (restante das informações
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text('Age: ${item['age']}'),
              Text('State: ${item['state']}'),
              Text('Address: ${item['address']}'),
            ],
          ),
        ));
  },
),
```



Código disponível em:



https://docs.google.com/document/d/1WgEEqMQSxi9L1Itn2XeFx-sHLGBFnPqZJr2ivjVxhQM/edit?usp=sharing