



Cross-Platform

Flávio Moreni



Agenda

Aula anterior

- Singleton
- SQLite (SQLite para Flutter)

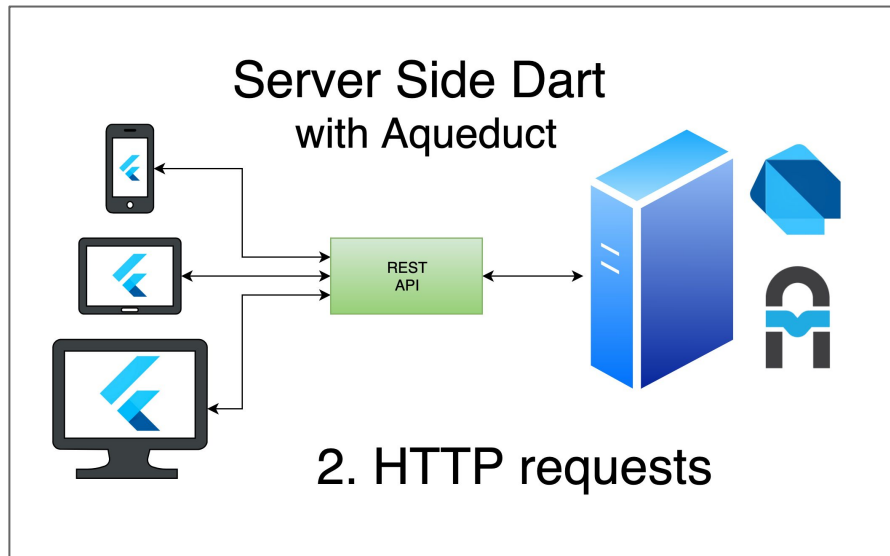
Aula de hoje

- WebServices
- Api Rest
- Lib Dio (Http Client)



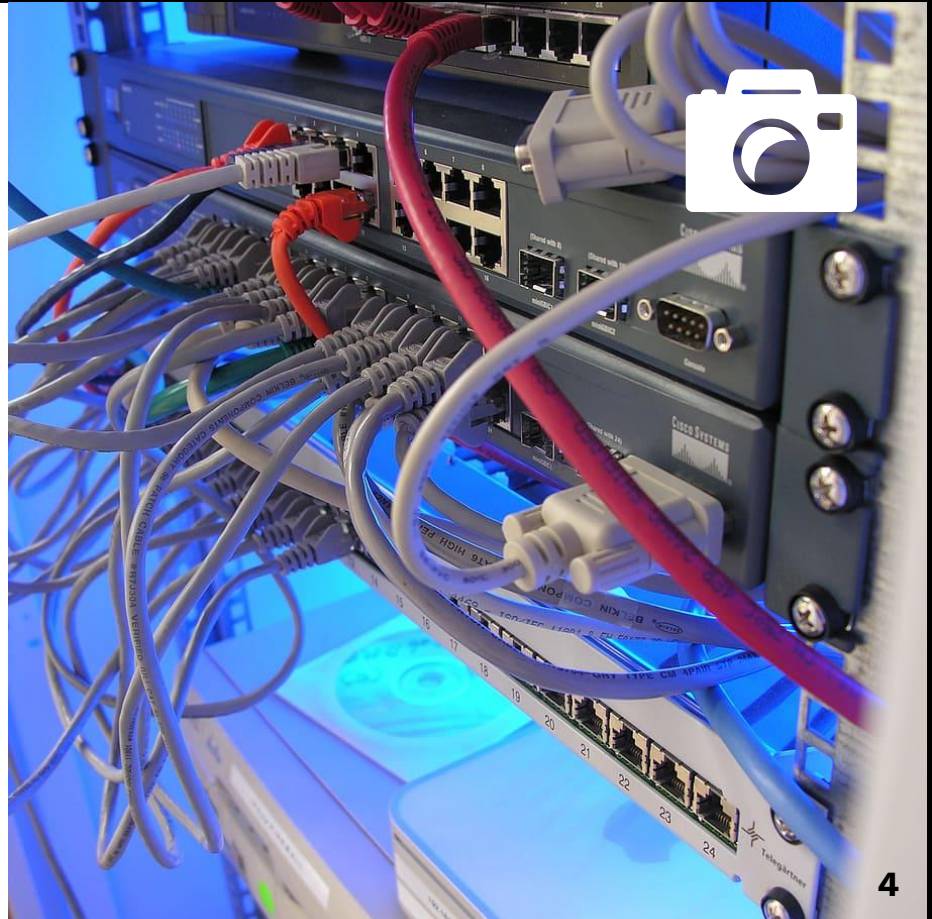
Objetivo

Efetuar a persistência de dados por meio de WebApi, utilizando a lib Dio para efetuar as requisições Http.



Web Api

Uma web API server-side é uma interface programática consistente de um ou mais endpoints publicamente expostos para um sistema definido de mensagens pedido-resposta, tipicamente expressado em JSON ou XML, que é exposto via a internet - por meio de um servidor web baseado em HTTP.




DIO HttpClient - Install



1. Abra o arquivo **pubspec.yaml**.
2. Na sessão **dependencies** abaixo do **flutter sdk** adicione a biblioteca **dio: ^3.0.9**
3. Salve o arquivo.
4. Execute o comando **flutter pub get**
5. Para usar o Sqlite faça a o **import 'package:dio/dio.dart';**

```
dependencies:  
  flutter:  
    sdk: flutter  
  sqflite: ^1.3.0  
  dio: ^3.0.9
```



CursoModel

```
factory CursoModel.fromMap(Map<Stri  
    id: ·  
    ······ (json["id"] is String) ?  
    ······ int.parse(json["id"]) :  
    ······ json["id"],  
    nome: json["nome"],  
    nivel: json["nivel"],  
    percentualConclusao: json["perc  
    preco: json["preco"],  
    conteudo: json["conteudo"],  
);
```



Cursos WebApi

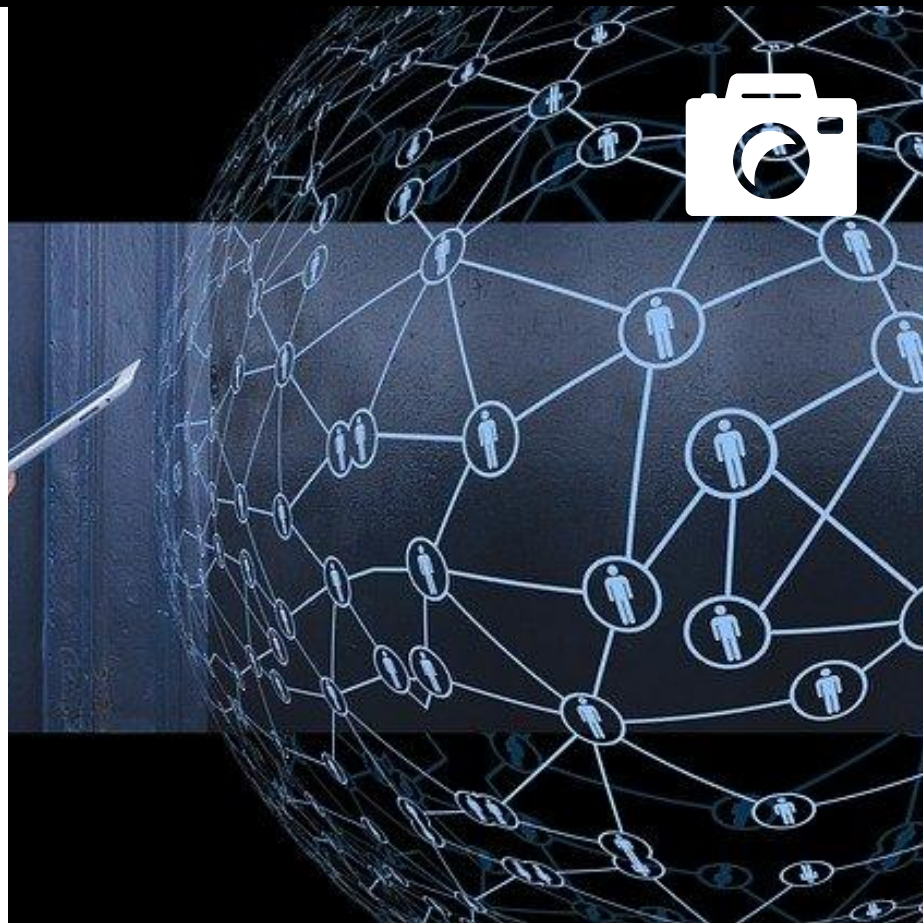
Endpoint:

<https://5cb544bd07f233001424ceb8.mockapi.io/fiap/>

Recurso: **curso**

Verbos Http: **get, post, put, delete**

ON	GET	→ /curso	<input type="text" value="\$mockData"/>
ON	GET	→ /curso/:id	<input type="text" value="\$mockData"/>
ON	POST	→ /curso	<input type="text" value="\$mockData"/>
ON	PUT	→ /curso/:id	<input type="text" value="\$mockData"/>
ON	DELETE	→ /curso/:id	<input type="text" value="\$mockData"/>



```
if ( _doubt )
```

```
{
```

```
    askNow();
```

```
}
```



WebApi - Dio HttpClient

Criando classe de configuração e os interceptadores



Classe **ServiceConfig**

Exemplo:

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

class ServiceConfig {
  String _url = "";
  int timeout = 5000;

  ServiceConfig(this._url, {this.timeout = 5000} );
}
```

Método Create:

```
Dio create() {
  Dio dio = Dio(
    BaseOptions(
      connectTimeout: timeout,
      baseUrl: _url,
    ),
  );
  dio.interceptors.add(
    InterceptorsWrapper(
      onRequest: (RequestOptions request) async {
        //request.headers["token"] = "sa09f0dfkjfkashd";
        return request;
      },
      onResponse: (Response response) async {
        return response;
      },
      onError: (DioError error) async {
        return error;
      },
    ),
  );
  return dio;
}
```

WebApi - Curso Services

Criando classe para manipular os dados do curso



```
import 'package:dio/dio.dart';
import 'package:lista_cursos/models/curso_model.dart';
import 'package:lista_cursos/services/Service_config.dart';
```

```
class CursoService {
  static final String _endpoint =
    "https://5cb544bd07f233001424ceb8.mockapi.io/fiap";

  static final String _resource = 'curso';

  final ServiceConfig service = new
  ServiceConfig(_endpoint);
}
```

Método findAll:

```
Future<List<CursoModel>> findAll() async {
  List<CursoModel> lista = new List<CursoModel>();
  try {
    Response response = await service.create().get(_resource);
    if (response.statusCode == 200) {
      response.data.forEach(
        (value) {
          lista.add(
            CursoModel.fromMap(value),
          );
        },
      );
    }
  } catch (error) {
    throw error;
  }
  return lista;
}
```

WebApi - Curso Services

Criando classe para manipular os dados do curso



Método getByID:

```
Future<CursoModel> getById(int id) async {  
  try {  
    String endpoint = _resource + "/" + id.toString();  
    Response response = await service.create().get(endpoint);  
  
    if (response.statusCode == 200) {  
      var retorno = CursoModel.fromMap(response.data);  
      return retorno;  
    }  
  
  } catch (error) {  
    print("Service Error: $error");  
    throw error;  
  }  
}
```

Método create:

```
Future<int> create(CursoModel cursoModel) async {  
  try {  
    cursoModel.id = 0;  
    Response response = await service.create().post(  
      _resource,  
      data: cursoModel.toMap(),  
    );  
    if (response.statusCode == 201) {  
      var retorno = (response.data["id"] is String) ?  
        int.parse(response.data["id"]) :  
        response.data["id"];  
      return retorno;  
    }  
  } catch (error) {  
    print("Service Error: $error");  
    throw error;  
  }  
}
```

WebApi - Curso Services

Criando classe para manipular os dados do curso



Método update:

```
Future<int> update(CursoModel cursoModel) async {  
  try {  
    String endpoint = _resource + "/" + cursoModel.id.toString();  
    Response response = await service.create().put(  
      endpoint,  
      data: cursoModel.toMap(),  
    );  
  
    var retorno = ( response.data["id"] is String ) ?  
      int.parse(response.data["id"]) :  
      response.data["id"];  
    return retorno;  
  } catch (error) {  
    print("Service Error: $error ");  
    throw error;  
  }  
}
```

Método delete:

```
Future<void> delete(CursoModel cursoModel) async {  
  try {  
    String endpoint = _resource + "/" + cursoModel.id.toString();  
  
    Response response = await service.create().delete(  
      endpoint,  
    );  
  
    if (response.statusCode != 200) {  
      throw Exception("Não foi possível excluir o recurso!");  
    }  
  } catch (error) {  
    throw error;  
  }  
}
```



```
if ( _doubt )  
{  
    askNow();  
}
```

Prática Services

Chegou a hora de eliminar todas as chamadas de persistência local e fazer as informações serem gravadas remotamente.

1. Na camada View, altere todas as chamadas aos métodos do Repository para métodos da camada service.
2. Simule um erro na lista de cursos (basta trocar a url).
3. Na tela de listagem de cursos, implemente uma forma de exibir erros no componente **FutureBuilder**.





Questions?



Thanks!

A solução será publicada do portal