

Assincronismo e APIs REST

Danilo Perez

Full Stack Developer

 **perez-danilo**

 **Fala devs**

Objetivo Geral

Objetivo deste módulo é aprender a acessar recursos através do protocolo REST, que nos permitirá obter dados de APIs. Para isso usaremos as bibliotecas do Flutter e seus pacotes, que facilitarão esse acesso, obtendo dados de APIs abertas e protegidas.

Aprenderemos sobre o que é REST, métodos HTTP, status code e comunicação assíncrona.

Paginaremos chamadas de APIs e muito mais

Percurso

- Etapa 1** Visão Geral Sobre APIs REST e Protocolo HTTP
- Etapa 2** Biblioteca "http" e o conceito de assincronismo (Future)
- Etapa 3** Criando um service padrão para as APIs
- Etapa 4** Acessando API autenticada
- Etapa 5** Usando outros verbos HTTP – CRUD de Tarefas

Etapa 1

Flutter

// Visão Geral Sobre APIs REST e Protocolo HTTP

O que é REST e HTTP?

A sigla REST — Representational State Transfer, significa “Transferência de Estado Representacional”. A utilização da arquitetura REST, permite a comunicação entre aplicações.

O HTTP (HyperText Transfer Protocol) é o caminho mais conhecido nas transferências de dados. A maioria das APIs RESTful utilizam o HTTP como protocolo de comunicação oficial, uma vez que apresenta uma interface de operações padronizadas.

O que é REST e HTTP?

O HTTP permite criar, atualizar, pesquisar, executar e remover operações, atuando sob determinados recursos. Apresenta também um apanhado de respostas, guiando os clientes (navegadores ou APIs) nas suas ações diante de resposta específicas.

Ao abrir o navegador, ele estabelece uma conexão TCP/IP com o servidor de destino e envia uma requisição GET HTTP, com o endereço buscado.

O que é REST e HTTP?

O servidor, então, interpreta a requisição, retornando com uma resposta HTTP ao navegador. Essa resposta pode ser completa, com representações em formato HTML, ou apresentar erro, afirmando que o recurso solicitado não foi encontrado.

Os Web Services que adotam REST são mais leves e perfeitos na busca da metodologia ágil. Outro diferencial é a flexibilidade, sendo possível escolher o formato que melhor se encaixa para as mensagens do sistema. Os mais utilizados, além do texto puro, são Json e XML, dependendo da necessidade de cada momento.

Verbos HTTP

- GET – Utilizado para obter dados da API
- POST – Utilizado para criação de recursos
- PUT – Utilizado para criar ou atualizar dados em uma API
- PATCH – Utilizado para alteração de informações
- DELETE – Utilizado para exclusão de dados

Status Code

- 200 OK
- 201 Created
- 202 Accepted
- 204 No Content
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 500 Internal Server Error
- 502 Bad Gateway
- 503 Service Unavailable

Percurso

- Etapa 1** ~~Visão Geral Sobre APIs REST e Protocolo HTTP~~
- Etapa 2** Biblioteca "http" e o conceito de assincronismo (Future)
- Etapa 3** Criando um service padrão para as APIs
- Etapa 4** Acessando API autenticada
- Etapa 5** Usando outros verbos HTTP – CRUD de Tarefas

Etapa 2

Flutter

// Biblioteca "http" e o conceito de assincronismo (Future)

Flutter HTTP e Future

No Flutter para acessar dados de uma API precisamos da biblioteca HTTP, ela não vem diretamente com o Dart e precisaremos fazer a sua instalação.

As chamadas HTTP são realizadas de forma assíncrona, as quais a aplicação, por padrão, não fica esperando o resultado.

Quando temos um método que tem esse comportamento podemos fazer uso de um recurso chamado Future, o qual determina que esta função não terá seu retorno imediato e sim no futuro.

Flutter HTTP e Future

Para isso no fim da função podemos executar um comando `“.then”`, o qual será executado após a finalização da requisição.

Em contrapartida, podemos anotar a função com a palavra reservada `“async”` e quando queremos o programa ao invés de executar o próximo comando, ele aguarde o retorno da função assíncrona, usamos a palavra `await`.

Acessando endpoint de CEP

Conheceremos o site ViaCep, o qual nos disponibiliza de forma gratuita acesso a um endpoint o qual possui informações de vários endereços do Brasil, através de uma chama REST, onde podemos informar apenas o CEP.

Esse recurso é processado no servidor e pode ser retornado para nós em formato JSON.

<https://viacep.com.br/>

Conhecendo JsonPlaceholder

Conheceremos o site JsonPlaceholder que nos fornece uma API REST para usarmos como teste para nossas aplicações, nela podemos simular a criação de um blog, lista de tarefas, álbuns, fotos e usuários.

Com ele criaremos uma lista de posts e ao clicar a lista de comentários de um Post.

<https://jsonplaceholder.typicode.com/>

Percurso

Etapa 1 ~~Visão Geral Sobre APIs REST e Protocolo HTTP~~

Etapa 2 ~~Biblioteca "http" e o conceito de assincronismo (Future)~~

Etapa 3 Criando um service padrão para as APIs

Etapa 4 Acessando API autenticada

Etapa 5 Usando outros verbos HTTP – CRUD de Tarefas

Etapa 3

Flutter

// Criando um service padrão para as APIs

Conhecendo a biblioteca Dio

Seremos apresentados a um outro pacote de chamadas HTTP, chamado Dio. Ele é muito bem visto pela comunidade, pois ele possui várias funcionalidade que ajudam o desenvolvimento.

Podemos criar classes que injetarão informações dentro de todas chamadas HTTP de nossa aplicação, criação de logs

Conhecendo a dotenv

O dotenv é um pacote muito conhecido no meio dos desenvolvedores, onde criamos variáveis dentro de arquivos e podemos usar os mesmos dentro de nossa aplicação.

Usado principalmente para setar informações que podem mudar dependendo ambiente usado QA, PROD e DEV.

Também usado para não subir informações de chaves e senhas nos controles de versão de fonte, exemplo github.

Criando interface

Como aprendemos a usar tanto a biblioteca http, quanto a Dio, criaremos uma interface a qual as duas classes implementarão.

Dessa forma podemos chavear entre a utilização de Dio e http a hora que quisermos.

Passando a vantagem de uso de interfaces.

Percurso

- Etapa 1** ~~Visão Geral Sobre APIs REST e Protocolo HTTP~~
- Etapa 2** ~~Biblioteca "http" e o conceito de assincronismo (Future)~~
- Etapa 3** ~~Criando um service padrão para as APIs~~
- Etapa 4** Acessando API autenticada
- Etapa 5** Usando outros verbos HTTP – CRUD de Tarefas

Etapa 4

Flutter

// Acessando API autenticada

O que é autenticação e autorização

A autorização e a autenticação se referem a parte de segurança, não só de APIs como páginas web.

A autenticação diz respeito ao usuário ter as credenciais para logar naquela aplicação. Quando não logados recebemos geralmente um 401 - Unauthorized

A autorização se o usuário em questão tem autorização para acessar determinado recurso. Quando não temos permissão recebemos um 403 - forbidden

Marvel Developer Portal

Para acesso a APIs que dependem de um certo privado, diferentemente do ViaCep.

No site da Marvel poderemos criar uma conta e receber credenciais de acesso para consumir informações de personagens, quadrinhos e muito mais.

Nesta parte aprenderemos acessar um API autenticada e aprenderemos a utilizar a biblioteca crypto do Flutter para criptografar os dados.

Criando paginação

Após listar os dados dos personagens da Marvel, aprenderemos a listar os dados de forma paginada.

Primeiramente será usado paginação com ação em um botão

Na sequencia faremos a utilização do recurso de paginação infinita.

Percurso

- Etapa 1** ~~Visão Geral Sobre APIs REST e Protocolo HTTP~~
- Etapa 2** ~~Biblioteca "http" e o conceito de assincronismo (Future)~~
- Etapa 3** ~~Criando um service padrão para as APIs~~
- Etapa 4** ~~Acessando API autenticada~~
- Etapa 5** Usando outros verbos HTTP – CRUD de Tarefas

Etapa 5

Flutter

// Usando outros verbos HTTP – CRUD de Tarefas

Aplicando verbos HTTP

Até o momento acessamos APIs apenas fazendo uso do verbo HTTP GET, agora aprenderemos a usar os outros métodos.

POST

PUT

DELETE

Conhecendo o Back4App

Como ferramenta de apoio conheceremos o Back4App o qual nos possibilita uma gama de recursos, como banco de dados, acesso via API, SDKs, controle de usuários e etc.

Criaremos uma classe de lista de tarefas e faremos todo o gerenciamento da mesma usando requisições HTTP.

Faremos uso de classes customizada de acesso a dados, dotenv e interceptors para envio de cabeçalhos de autenticação.

Links Úteis

- [digitalinnovationone/dio-flutter \(github.com\)](https://github.com/digitalinnovationone/dio-flutter)
- [JSONPlaceholder - Free Fake REST API \(typicode.com\)](https://typicode.com/)
- [Back4App - Low-code backend to build modern apps](#)
- [Marvel Developer Portal](#)
- [JSON to Dart \(javiercbk.github.io\)](https://javiercbk.github.io/)

Para saber mais

Artigos e cursos da DIO

“Fala Devs” youtube

Dúvidas?

- > Fórum/Artigos
- > Comunidade Online (Discord)



Desafio – Cadastro de CEPs

- Criar uma aplicação Flutter
- Criar uma classe de CEP no Back4App
- Consulte um Cep no ViaCep, após retornado se não existir no Back4App, realizar o cadastro
- Listar os CEPs cadastrados em forma de lista, possibilitando a alteração e exclusão do CEP