

Curso de Testes de API Rest

Julio de Lima

Vídeos: <https://www.youtube.com/watch?v=VqVQ7vHY32o&list=PLf8x7B3nFTI17WeEVj405tHlstiq1kNBX>

Aula 01 - Introdução aos testes de API Rest

Saber manipular um software web não é o suficiente para que você seja capaz de testar o software.

Se você souber manipular requisições, isso por si só não é suficiente para que você possa testar uma API Rest.

O curso é dividido em duas partes:

Primeira parte: manipular API's Rest

Segunda parte: testar API's Rest

Plano de ensino

Primeira parte (manipular API's Rest):

1. Introdução a API's Rest
2. Service Repository e o tal de Controller
3. Request & Response
4. Headers
5. Parâmetros
6. Autenticação e Autorização
7. Insomnia
8. Postman

Segunda parte (testar API's Rest):

9. RestAssured
10. Estratégias de Testes de API Rest
11. Cobertura de Testes
12. Testes Funcionais
13. Testes de Performance
14. Testes de Compatibilidade
15. Mountebank
16. JSONSchema
17. Leitura de Logs
18. Dicas para Entrevistas

Aula 02 - Introdução a API's Rest

Um estilo de arquitetura

API Rest é a junção de dois conceitos:

API que é uma interface de programação de aplicações que traz consigo um objetivo muito claro que é expor um serviço sem que as pessoas que vão consumir esse serviço precisem conhecer qual é a sua estrutura.

O objetivo primário da API é poder expor os seus serviços sem que as pessoas que vão consumi-lo tenham acesso aos recursos internos dentro dele.

API: comunicação entre dois softwares.

UI: comunicação entre um software e o usuário.

Quando falamos em uma UI a característica primária é a **usabilidade**. Em uma API a característica primária dela é a **interoperabilidade** (um sub-característica de qualidade de software).

Para realizar a comunicação entre dois softwares é necessário seguir um protocolo, uma recomendação (padrão oficial), um estilo de arquitetura.

REST e **SOAP** são estilos de arquitetura.

API's Rests são muito utilizadas atualmente.

Aula 03 - O que está por detrás da API Rest

O **backend** está por detrás da API

O **frontend** são interfaces gráficas que fazem a interface entre o usuário e o software.

API é uma interface entre dois softwares.

UI é uma interface com o usuário (interface entre um software e um ser-humano).

Service e Repository

Service é uma camada dentro do backend da aplicação que geralmente armazena as regras de negócio.

Repository é uma outra camada da aplicação que é responsável por trafegar as informações entre a regra de negócio e o banco de dados, por exemplo, ou sistemas de armazenamento que você possui. Em resumo, é a interface com o banco de dados.

Exemplo:

Quero um software capaz de buscar todas as viagens ou filtrar por região.

No Service são implementados dois métodos:

- Um para buscar todas as viagens.

```
public List<Viagem> listar() {  
    return viagemRepository.findAll();  
}
```

- Outro para filtrar as viagens por região

```
public List<Viagem> buscarViagensPorRegiao(String regiao) {  
    List<Viagem> viagens = viagemRepository.findAllByRegiao(regiao);  
  
    if (viagens.isEmpty()) {  
        throw new ViagemServiceException("Não existem viagens cadastradas para esta Região!");  
    }  
  
    return viagens;  
}
```

ViagemRepository

```
package com.montanha.gerenciador.repositories;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.montanha.gerenciador.entities.Viagem;

import java.util.List;

@Repository
public interface ViagemRepository extends JpaRepository<Viagem, Long> {

    Viagem findByLocalDeDestino(String localDeDestino);

    List<Viagem> findAllByRegiao(String regiao);

}
```

Controller

O controller usa a implementação da API Rest buscando as informações através dos serviços.

O controller é o intermediador entre quem chama e os services e repositorys.

O controller faz a intermediação entre a aplicação que está consumindo a API Rest e os serviços que existem no backend.

O Postman chama o controller. O controller determina o que vem na requisição e qual o método que ele deve chamar. E aí chama os services. O Service chama o repository, o repository interage com o banco de dados que devolve a informação para o repository, que devolve a informação para o service, que devolve a informação para o controller. E o controller devolve a informação para a interface gráfica que renderiza esses dados.

```
@ApiOperation(value = "Retorna todas as viagens")
@RequestMapping(value = "/v1/viagens", method = RequestMethod.GET, produces = "application/json")
@PreAuthorize("hasAnyRole('USUARIO')")
public ResponseEntity<Response<List<Viagem>>> listar(@RequestParam(value = "regiao", required = false) String
regiao, @RequestHeader String Authorization) {
    List<Viagem> viagens = null;

    if (regiao == null) {
        viagens = viagemService.listar();
    } else {
        viagens = viagemService.buscarViagensPorRegiao(regiao);
    }

    Response<List<Viagem>> viagensResponse = new Response<>();
    viagensResponse.setData(viagens);
    return ResponseEntity.status(HttpStatus.OK).body(viagensResponse);
}
```

Pense em uma requisição como uma consulta.

Só quem pode consultar as viagens é alguém do tipo usuário e que tenha as credenciais válidas (um administrador, por exemplo, não pode).

- **regiao** não é um parâmetro obrigatório, ou seja, pode ou não ser passado.
- O método **GET** é chamado de **safe** porque não altera o estado do servidor.
- Na resposta (**response**) é devolvido um **status code**, que no caso de um processamento com sucesso, é **200**.
- Por recomendação do Rest, toda resposta vem no formato **json**. O formato json é reconhecido pela maioria das linguagens de programação.

Aula 04 - O que são requests e responses

Uma conversa entre dois softwares

Requisição e Resposta, é através desses dois elementos que os softwares conseguem se comunicar.

cURL

Permite enviar requisições via HTTP.

O cURL é utilizado via linha de comando através de um terminal (por exemplo: prompt de comandos).

O token é uma série de caracteres alfa-numéricos que irá permitir que consigamos ter acesso a um determinado endpoint.

Fazendo a requisição com o curl

```
curl -X GET http://localhost:8089/api/v1/viagens
```

O `/api` para a API do Antonio Montanha é um endereço padrão (base).

Essas informações podem ser obtidas com o uso Swagger, que veremos na próxima aula.

```
D:\github\gerenciador-viagens>curl -X GET http://localhost:8089/api/v1/viagens
{"timestamp": "2022-05-21T12:09:52.125+0000", "status": 401, "error": "Unauthorized", "message": "Acesso negado. Você deve estar autenticado no sistema para acessar a URL solicitada.", "path": "/api/v1/viagens"}
```

É necessário um token de usuário para visualizar as viagens.

Obtendo um token de administrador

```
curl -X POST -i http://localhost:8089/api/v1/auth -d '{ "email": "admin@email.com", "senha": "654321" }' -H 'Content-Type: application/json'
```

```
Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (main)
$ curl -X POST -i http://localhost:8089/api/v1/auth -d '{ "email": "admin@email.com", "senha": "654321" }' -H 'Content-Type: application/json'
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload Total   Spent    Left  Speed
100  305     0  256  100    49   1075    205 --:--:-- --:--:-- 1286HT
TP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 21 May 2022 13:29:21 GMT

{"data": {"token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbkBlbWFpbC5jb20iLCJyb2x1IjoiUk9MRV9BRE1JTlIsImNyZWFOZlQiOjE2NTMzMzk3NjExNjUsImV4cCI6MTY1MzIzOTc2MH0._YSqBNTAP8qqb807id7e1VNiMYnQLVz6_U9xgXrNrzcw3KK1bwFMhdDluncbFV27xYgk9CXmG6mTtc3nNP0ZvQ"}, "errors": []}
```

token:

```
eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJhZG1pbkBlbWFpbC5jb20iLCJyb2xIjoiUk9MRV9BRE1JTilslmNyZWFOZWQiOjE2NTMx
Mzk3NjExNjUsImV4cCI6MTY1MzlzOtC2MH0._YsqBNTAP8qqb8O7iD7e1VNiMYnQLVz6_U9xgXrNrzcu3KKlbWMhdDlu
ncbFV27xYgk9CXmG6mTtc3nNP0ZvQ
```

Cadastrando viagem como administrador

Para cadastrar viagens é necessário ser administrador (usuário não tem autorização)

```
curl -X POST -is http://localhost:8089/api/v1/viagens -d '{ "acompanhante": "Isabelle", "dataPartida": "2021-01-23",
"dataRetorno": "2021-02-23", "localDeDestino": "Manaus", "regiao": "Norte" }' -H 'Content-Type: application/json' -H
'Authorization:
eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJhZG1pbkBlbWFpbC5jb20iLCJyb2xIjoiUk9MRV9BRE1JTilslmNyZWFOZWQiOjE2NTMx
Mzk3NjExNjUsImV4cCI6MTY1MzlzOtC2MH0._YsqBNTAP8qqb8O7iD7e1VNiMYnQLVz6_U9xgXrNrzcu3KKlbWMhdDlu
ncbFV27xYgk9CXmG6mTtc3nNP0ZvQ'
```

```
Roberto@DESKTOP-HGDUAQT MINGW64 /d/github/gerenciador-viagens (main)
$ curl -X POST -is http://localhost:8089/api/v1/viagens -d '{ "acompanhante": "Isabelle", "dataPartida": "2021-01-23", "dataRetorno": "2021-02-23", "localDeDestino": "Manaus", "regiao": "Norte" }' -H 'Content-Type: application/json' -H 'Authorization: eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJhZG1pbkBlbWFpbC5jb20iLCJyb2xIjoiUk9MRV9BRE1JTilslmNyZWFOZWQiOjE2NTMxMzk3NjExNjUsImV4cCI6MTY1MzlzOtC2MH0._YsqBNTAP8qqb8O7iD7e1VNiMYnQLVz6_U9xgXrNrzcu3KKlbWMhdDlu
ncbFV27xYgk9CXmG6mTtc3nNP0ZvQ'
HTTP/1.1 201
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Location: http://localhost:8089/api/v1/viagens/1
Content-type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 21 May 2022 13:37:24 GMT
[{"data": {"id": 1, "localDeDestino": "Manaus", "dataPartida": "2021-01-23", "dataRetorno": "2021-02-23", "acompanhante": "Isabelle", "regiao": "Norte"}, "errors": []}]
```

Obtendo um token de usuário

```
curl -X POST -i http://localhost:8089/api/v1/auth -d '{ "email": "usuario@email.com", "senha": "123456" }' -H
'Content-Type: application/json'
```

```
Roberto@DESKTOP-HGDUAQT MINGW64 /d/github/gerenciador-viagens (main)
$ curl -X POST -i http://localhost:8089/api/v1/auth -d '{ "email": "usuario@email.com", "senha": "123456" }' -H 'Content-Type: application/json'
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100  312     0  261  100  51 1095  214 --:--:--:--:--:-- 1316HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 21 May 2022 13:39:48 GMT
[{"data": {"token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJ1c3VhcmhvQGVtYWlsLmNvbSIsInJvbGUiOiJST0xFX1VTUFSSU8iLCJjcmVhdGVkIjoxNjUzMzg4OTE3LCJ1eHA1OjE2NTMyNDAzODd9.B4VBbjubW-wSBh0L8QmwGhX0cFsxDdkq1d534YtV7YR740RUTALKjz6IHQiz76ktS6vMr7C91JYYhDCglbgXeew"}, "errors": []}]
```

token:

```
eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJ1c3VhcmhvQGVtYWlsLmNvbSIsInJvbGUiOiJST0xFX1VTUFSSU8iLCJjcmVhdGVkIjoxNjUzMzg4OTE3LCJleHAIoje2NTMyNDAzODd9.B4VBbjubW-
wSBh0L8QmwGhX0cFsxDdkq1d534YtV7YR740RUTALKjz6IHQiz76ktS6vMr7C91JYYhDCglbgXeew
```

Listar viagens como usuário

Apenas usuários podem listar viagens (o administrador não pode).

```
curl -X GET -i http://localhost:8089/api/v1/viagens -H 'Content-Type: application/json' -H 'Authorization:
eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJ1c3VhcmhvQGVtYWlsLmNvbSIsInJvbGUiOiJST0xFX1VTUFSSU8iLCJjcmVhdGVkIjoxNjUzMzg4OTE3LCJleHAIoje2NTMyNDAzODd9.B4VBbjubW-
wSBh0L8QmwGhX0cFsxDdkq1d534YtV7YR740RUTALKjz6IHQiz76ktS6vMr7C91JYYhDCglbgXeew'
```

```
Roberto@DESKTOP-HGDU4QT MINGW64 /d/github/gerenciador-viagens (main)
$ curl -X GET -i http://localhost:8089/api/v1/viagens -H 'Content-Type: application/json' -H 'Authorization: eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ1c3Vhcm1vQGVtYwlsLmNvbSIsInJvbGUiOiJST0xFX1VTVuFSSU8iLCJjcmVhdGvkJioxNjUzMTQwMzg40TE3LCJtEHA0JE2NTMyNDAxODd9.B4V8jubW-wS8hOL8QmwchX0cfFsxDdkq1d534YtV7YR740RUTALKJz6IHQiz76ktS6vMr7C9IJYyhDg1bgXeeW'
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload Upload Total Spent   Left Speed
100  154     0  154     0      0  2044      0  ---:--- --:--- --:--- 2081HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 21 May 2022 13:44:50 GMT
>{"data":[{"id":1,"localDeDestino":"Manaus","dataPartida":"2021-01-23","dataRetorno":"2021-02-23","acompanhante":"Isabelle","regiao":"Norte"}],"errors":[]}
```

Aula 05 - Instalando uma API Rest de Exemplo para Testes

Instalando o Java

java jdk 8 download

<https://www.oracle.com/br/java/technologies/javase/javase8-archive-downloads.html>

Baixar o Java SE Development Kit 8u202 para Windows

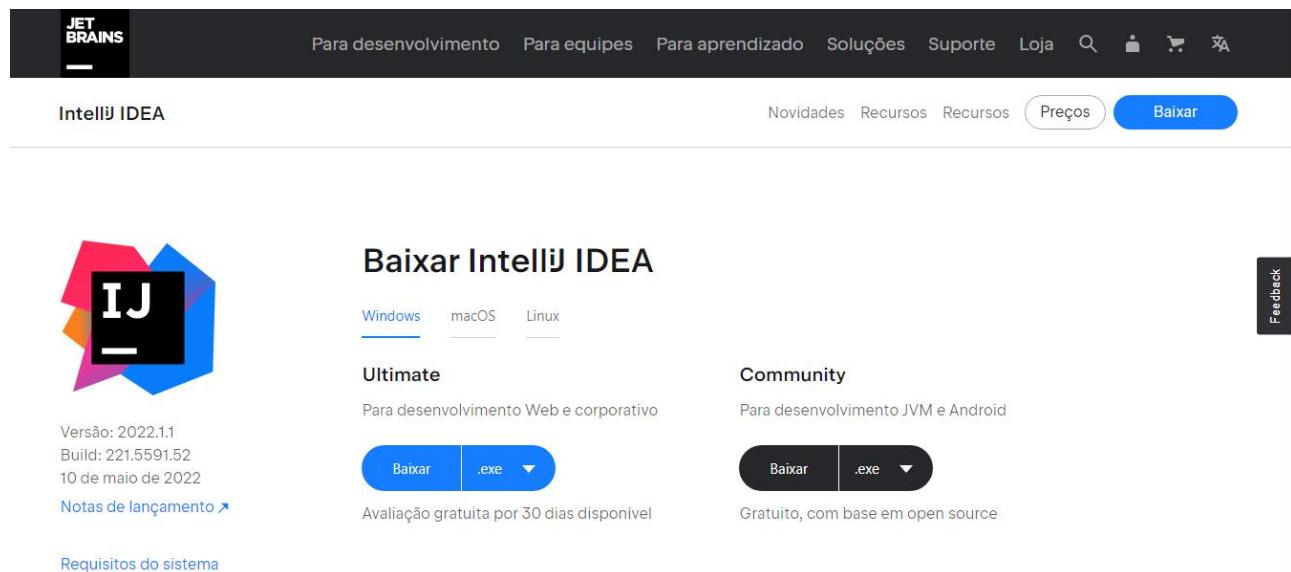
jdk-8u202-windows-x64.exe

java -version

```
C:\Users\Roberto>java -version
java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)
```

Instalando o intelliJ IDEA

<https://www.jetbrains.com/pt-br/idea/download/#section=windows>



The screenshot shows the IntelliJ IDEA download page. At the top, there's a navigation bar with links for 'Para desenvolvimento', 'Para equipes', 'Para aprendizado', 'Soluções', 'Suporte', 'Loja', and icons for search, user profile, cart, and feedback. Below the navigation, there's a main heading 'Baixar IntelliJ IDEA' with tabs for 'Windows' (selected), 'macOS', and 'Linux'. Under the 'Ultimate' tab, it says 'Para desenvolvimento Web e corporativo' and has a 'Baixar .exe' button. Under the 'Community' tab, it says 'Para desenvolvimento JVM e Android' and has a 'Baixar .exe' button. A note below the Ultimate tab says 'Avaliação gratuita por 30 dias disponível'. At the bottom left, there's a note about system requirements: 'Requisitos do sistema'. On the right side, there's a 'Feedback' button.

Baixe e instale a versão "Community".

Instalando o Maven

O Maven é um gerenciador de pacotes e gerador de processos automatizados a serem executados.

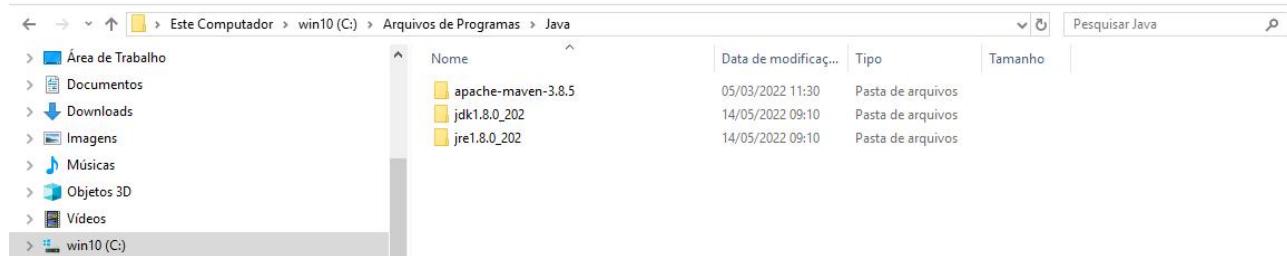
maven java download

<https://maven.apache.org/download.cgi>

	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.8.5-bin.tar.gz	apache-maven-3.8.5-bin.tar.gz.sha512	apache-maven-3.8.5-bin.tar.gz.asc
Binary zip archive	apache-maven-3.8.5-bin.zip	apache-maven-3.8.5-bin.zip.sha512	apache-maven-3.8.5-bin.zip.asc
Source tar.gz archive	apache-maven-3.8.5-src.tar.gz	apache-maven-3.8.5-src.tar.gz.sha512	apache-maven-3.8.5-src.tar.gz.asc
Source zip archive	apache-maven-3.8.5-src.zip	apache-maven-3.8.5-src.zip.sha512	apache-maven-3.8.5-src.zip.asc

Baixe apache-maven-3.8.5-bin.zip

Recorte e cole a pasta do Maven em C:\Program Files\Java



Cole no path (variáveis de ambiente) o endereço:

C:\Program Files\Java\apache-maven-3.8.5\bin

mvn -v

```
C:\Users\Roberto>mvn -v
Apache Maven 3.8.5 (3599d3414f046de2324203b78ddcf9b5e4388aa0)
Maven home: C:\Program Files\Java\apache-maven-3.8.5
Java version: 1.8.0_202, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jre1.8.0_202
Default locale: pt_BR, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Git

Baixe e instale o Git.

Repositório Github de Antonio Montanha

Baixe o repositório [gerenciador viagens](#) (código fonte da API Rest).

<https://github.com/AntonioMontanha/gerenciador-viagens>

Na pasta do github execute o comando:

```
git clone https://github.com/AntonioMontanha/gerenciador-viagens.git
```

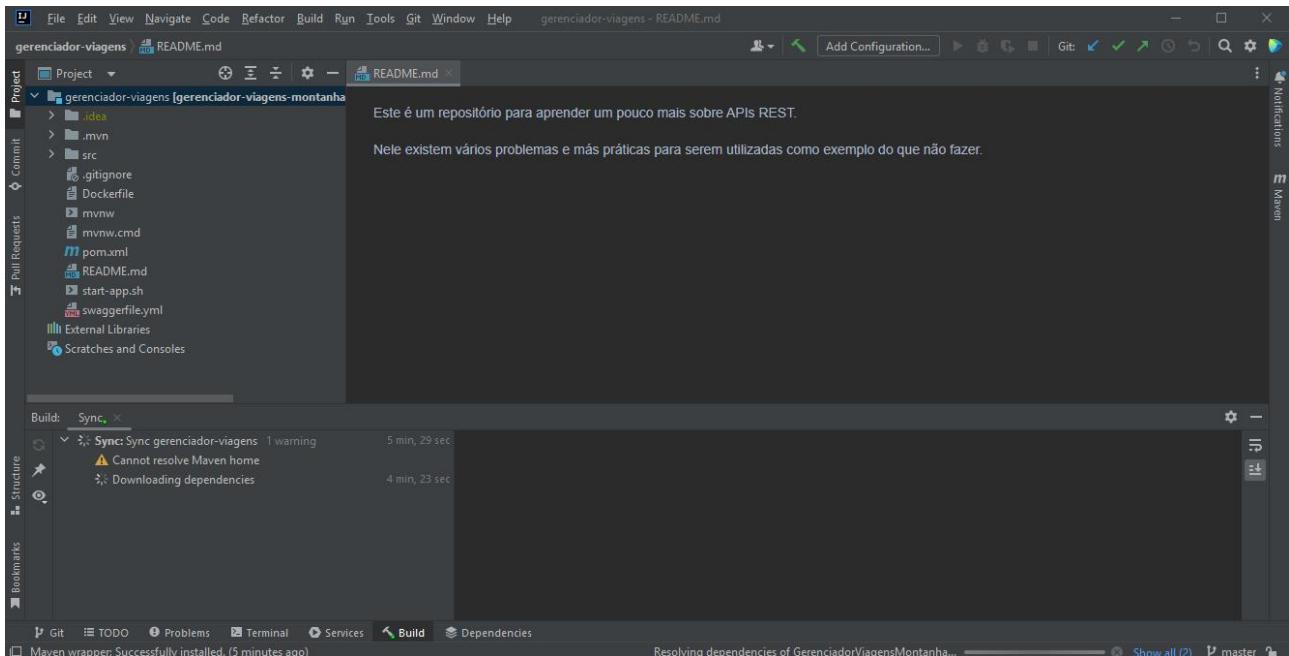
```

Roberto@DESKTOP-HGDUAAQT MINGW64 /d/github
$ git clone https://github.com/AntonioMontanha/gerenciador-viagens.git
Cloning into 'gerenciador-viagens'...
remote: Enumerating objects: 614, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 614 (delta 0), reused 1 (delta 0), pack-reused 608
Receiving objects: 100% (614/614), 124.10 KiB | 1.94 MiB/s, done.
Resolving deltas: 100% (225/225), done.

Roberto@DESKTOP-HGDUAAQT MINGW64 /d/github
$ |

```

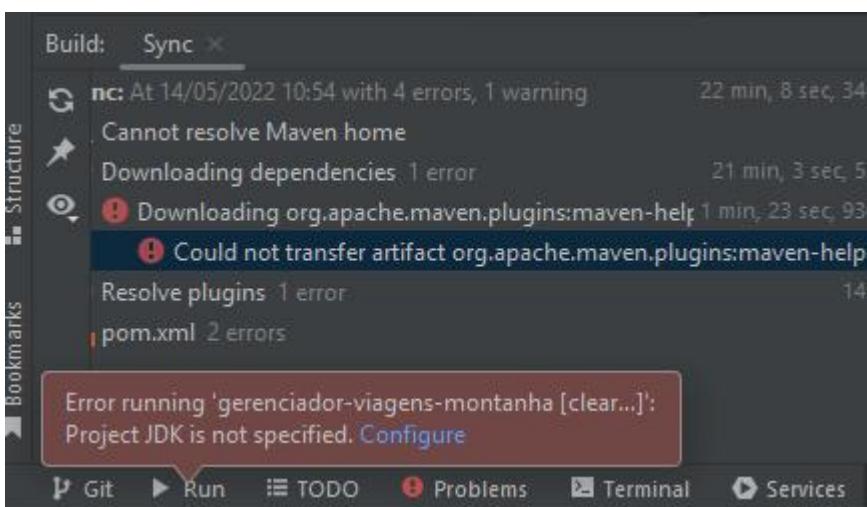
No IntelliJ, clique no botão "Open", selecione a pasta "gerenciador-viagens" e aguarde o carregamento.



No lado direito da tela clique na aba do "Maven":

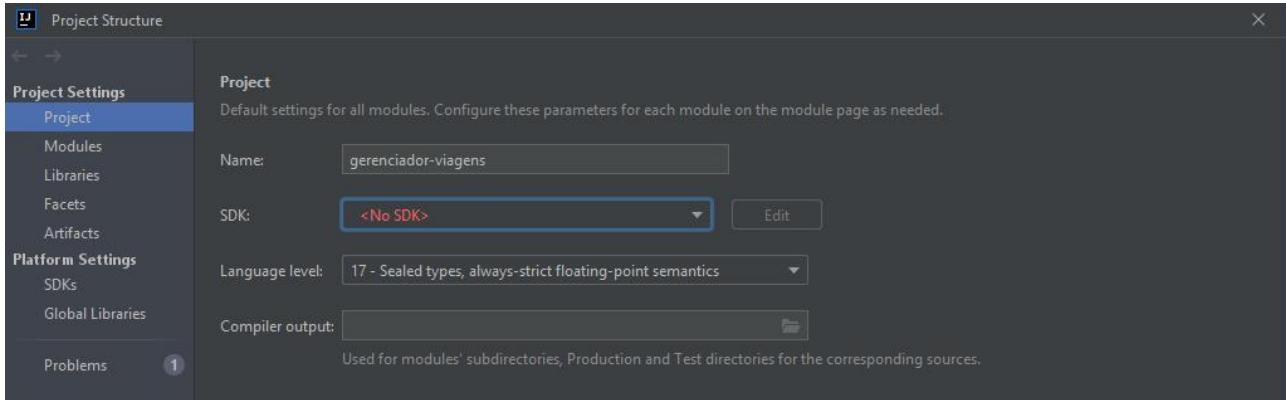
Clique no botão "Execute Maven Goal" e entre com o comando:

mvn clean install

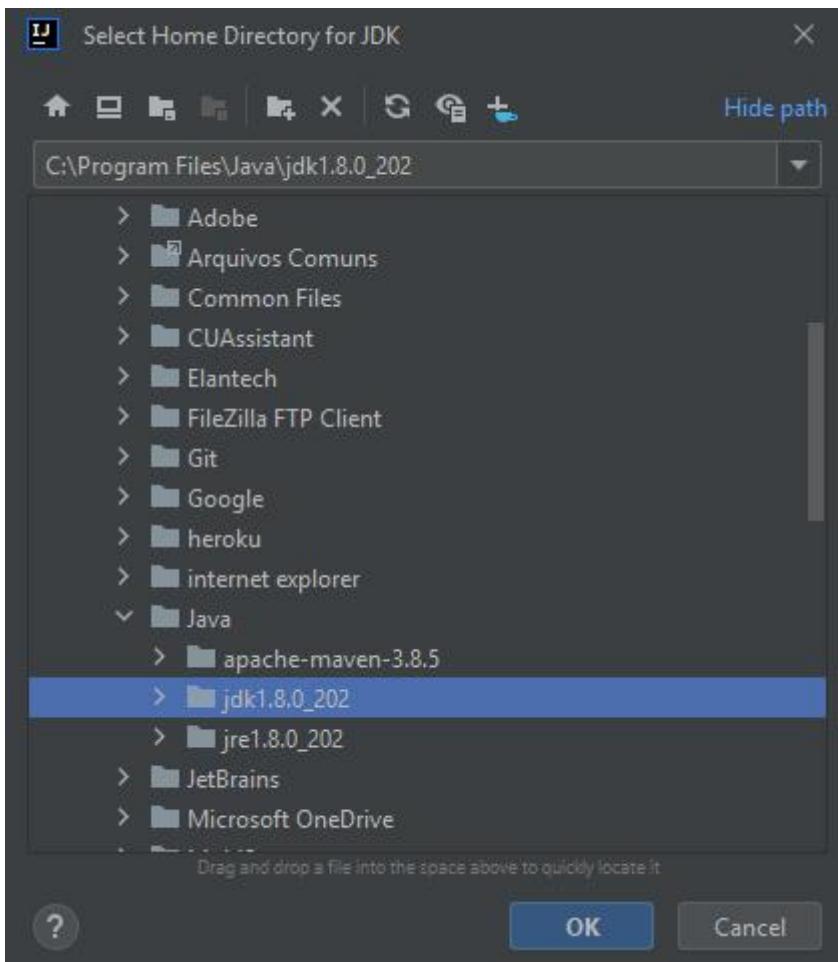


Não basta apenas instalar, é necessário configurar o JDK.

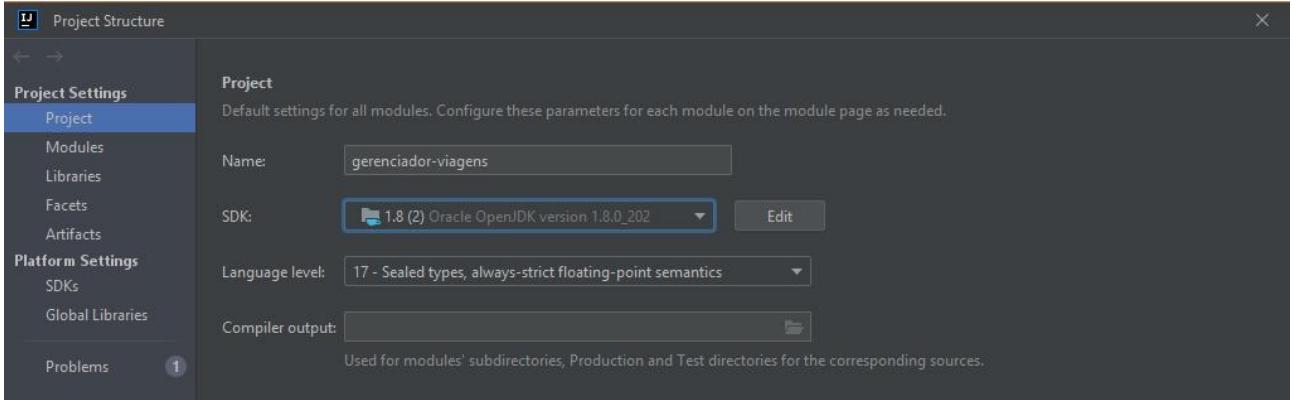
Clique no link "Configure"



Em SDK, clique em "Add SDK => JDK":

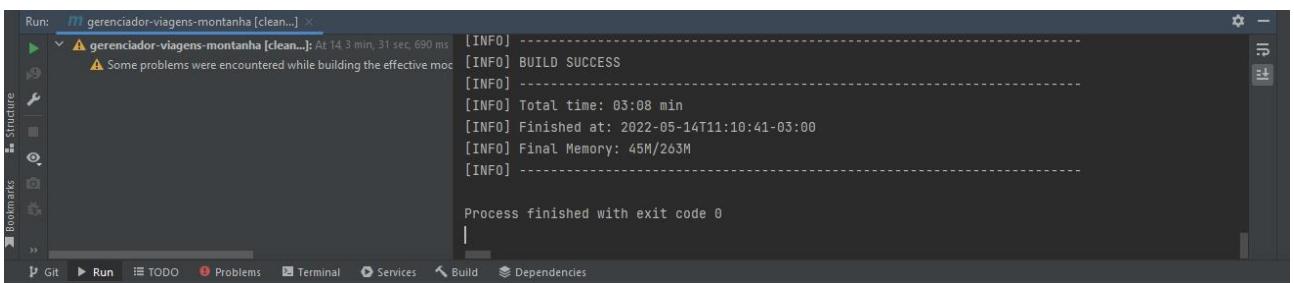


E clique no botão "OK".



Execute o comando:

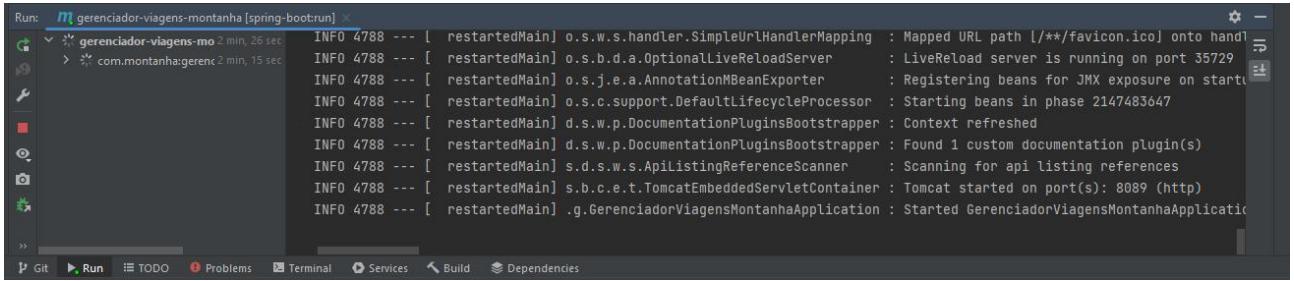
`mvn clean install`



Fazendo com que a API seja executada

Importante: Entre com o comando:

`mvn spring-boot:run`



No gitbash entre com o comando:

`curl -X GET http://localhost:8089/api/v1/status`



Aula 06 - Como usar Headers, Swagger e Verbos em APIs Rest

Swagger Editor (em <https://editor.swagger.io/>)

Um lugar especial para informações adicionais

Quem é você, qual tipo de dado está enviando, etc. São informações que podem ser enviadas no Header.

Swagger é basicamente uma documentação da interface em si com a qual se está interagindo.

localhost:8089/api/swagger-ui.html

The screenshot shows the Swagger UI interface with the following sections:

- authentication-controller**: Authentication Controller
 - POST /v1/auth**: Gera um Token de acesso
- gerenciador-viagem-controller**: Gerenciador Viagem Controller
 - GET /v1/viagens**: Retorna todas as viagens
 - POST /v1/viagens**: Cadastra uma viagem
 - GET /v1/viagens/{id}**: Retorna uma viagem específica
 - PUT /v1/viagens/{id}**: Atualiza uma viagem específica
 - DELETE /v1/viagens/{id}**: Apaga uma viagem específica
- status-aplicacao-controller**: Status Aplicacao Controller
 - GET /v1/status**: verificaStatusAplicacao

At the bottom left, there is a "Models" section with a right-pointing arrow.

Api Documentation 1.0

[Base URL: localhost:8089/api]
<http://localhost:8089/api/v2/api-docs>

Api Documentation

[Terms of service](#)

[Apache 2.0](#)

authentication-controller Authentication Controller

POST /v1/auth Gera um Token de acesso

Parameters

Name	Description
authenticationDto <small>* required</small>	authenticationDto (body)

Example Value | Model

```
{ "email": "string", "senha": "string" }
```

Parameter content type

application/json

Responses

Code	Description
200	OK
201	Created
401	Unauthorized
403	Forbidden
404	Not Found

Response content type

/

gerenciador-viagem-controller Gerenciador Viagem Controller



status-aplicacao-controller Status Aplicacao Controller



Models



```
curl -X POST -i http://localhost:8089/api/v1/auth -d '{ "email": "admin@email.com", "senha": "654321" }' -H 'Content-Type: application/json'
```

-d é o body

-H é o header

```
Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (master)
$ curl -X POST -i http://localhost:8089/api/v1/auth -d '{ "email": "admin@email.com", "senha": "654321" }' -H 'Content-Type: application/json'
% Total    % Received % Xferd  Average Speed   Time     Time     Time  Current
          Dload  Upload Total   Spent   Left  Speed
100  305     0  256  100    49  1251    239  --::-- --::-- --::--  1509HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Mon, 16 May 2022 18:26:05 GMT

{"data": {"token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbkBlbWFpbC5jb20iLCJyb2xIjoiUk9MRV9BRE1JTiIsImNyZWFOZWQiOjE2NTI3MjU1NjU5MTQsImV4cCI6MTY1MjgyNTU2NH0.OHwNXTokj6K2ut6wXBA0cIVX_edYxQW9ftDFYm9bv3uYmvs8kkKloF6gYqDk7j-pFhtXRV8fkAc7kA8KMJjrYw"}, "errors": []}

Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (master)
$ curl -X POST http://localhost:8089/api/v1/auth -d '{ "email": "admin@email.com", "senha": "654321" }' -H 'Content-Type: application/json'
% Total    % Received % Xferd  Average Speed   Time     Time     Time  Current
          Dload  Upload Total   Spent   Left  Speed
100  305     0  256  100    49  1116    213  --::-- --::-- --::--  1337 {"data": {"token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbkBlbWFpbC5jb20iLCJyb2xIjoiUk9MRV9BRE1JTiIsImNyZWFOZWQiOjE2NTI3MjUzNjcwMDcsImV4cCI6MTY1MjgyNTM2Nn0.h5pJuM-by6Pbu0HVb1xxNUhMwX9TRRoEmrLoelpz8pK57I1tryreuMdBzU2EzD2wIeJuB8i5v9ynJiU-6RN6QUA"}, "errors": []}
```

```
curl -X POST -is http://localhost:8089/api/v1/auth -d '{ "email": "admin@email.com", "senha": "654321" }' -H 'Content-Type: application/json'
```

```
Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (master)
$ curl -X POST -is http://localhost:8089/api/v1/auth -d '{ "email": "admin@email.com", "senha": "654321" }' -H 'Content-Type: application/json'
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Mon, 16 May 2022 18:56:00 GMT

{"data": {"token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbkBlbWFpbC5jb20iLCJyb2xIjoiUk9MRV9BRE1JTiIsImNyZWFOZWQiOjE2NTI3MjczNjAwMzMzMsImV4cCI6MTY1MjgyNzM1OX0.0BNT8lwNed8Spm8bT9F5JgvHeGu5xWG1_7XdRdy8SI1JJxeVbKAt8YbowAlrzCkodDrwRpnVqlW7xXRd1j8cL1Q"}, "errors": []}
```

Aula 07 - Autenticação e Autorização em APIs Rest

Ambos são termos muito comuns quando se fala em desenvolvimento de software em si, mas em APIs Rest são extremamente importantes.

```
curl -X POST -is http://localhost:8089/api/v1/viagens -d '{ "acompanhante": "Isabelle", "dataPartida": "2021-01-23", "dataRetorno": "2021-02-23", "localDeDestino": "Manaus", "regiao": "Norte" }' -H 'Content-Type: application/json' -H 'Authorization: eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJhZG1pbkBlbWFpbC5jb20iLCJyb2xlIjoiUk9MRV9BRE1JTlsImNyZWFOZWQiOjE2NTI3MjcNjAwMzMslmV4cCI6MTY1MjgyNzM1OX0.0BNT8lwNed8Spm8bT9F5JgvHeGu5xWG1_7XdRdy8SI1JJxeVbKAt8YbowA1rzCkodDrwRpnVqW7xXRd1j8cL1Q'
```

```
Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (master)
$ curl -X POST -is http://localhost:8089/api/v1/viagens -d '{ "acompanhante": "Isabelle", "dataPartida": "2021-01-23", "dataRetorno": "2021-02-23", "localDeDestino": "Manaus", "regiao": "Norte" }' -H 'Content-Type: application/json' -H 'Authorization: eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJhZG1pbkBlbWFpbC5jb20iLCJyb2xlIjoiUk9MRV9BRE1JTlsImNyZWFOZWQiOjE2NTI3MjcNjAwMzMslmV4cCI6MTY1MjgyNzM1OX0.0BNT8lwNed8Spm8bT9F5JgvHeGu5xWG1_7XdRdy8SI1JJxeVbKAt8YbowA1rzCkodDrwRpnVqW7xXRd1j8cL1Q'
HTTP/1.1 201
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Location: http://localhost:8089/api/v1/viagens/1
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Mon, 16 May 2022 19:00:48 GMT

{"data": {"id": 1, "localDeDestino": "Manaus", "dataPartida": "2021-01-23", "dataRetorno": "2021-02-23", "acompanhante": "Isabelle", "regiao": "Norte"}, "errors": []}
Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (master)
```

Fazendo autenticação com um usuário que não é administrador

```
curl -X POST -i http://localhost:8089/api/v1/auth -d '{ "email": "usuario@email.com", "senha": "123456" }' -H 'Content-Type: application/json'
```

```
Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (master)
$ curl -X POST -i http://localhost:8089/api/v1/auth -d '{ "email": "usuario@email.com", "senha": "123456" }' -H 'Content-Type: application/json'
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
          Dload  Upload Total   Spent    Left  Speed
100  312    0  261  100    51   1104    215 --:--:-- --:--:-- --:--:--  1333HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Mon, 16 May 2022 19:08:26 GMT

{"data": {"token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ1c3Vhcm1vQGVtYW1sLmNvbSIsInJvbGUiOiJST0xFX1VTVUFSSU8iLCJjcmVhdGVkIjoxNjUyNzI4MTA2ODIzLCJleHAiOjE2NTI4MjgxMDV9.P_-6wJXYSaYiOrIVg-YpG008qWg817YRNHoPTkoamZrq5mp3s2Q6EIwTKJ2hvpvoFs7M-31kv0QEeq-h4Qrvrg"}, "errors": []}
```

Tentando cadastrar uma viagem como usuário

O usuário consegue se autenticar, porém não é autorizado a cadastrar viagens.

```
curl -X POST -is http://localhost:8089/api/v1/viagens -d '{ "acompanhante": "Isabelle", "dataPartida": "2021-01-23", "dataRetorno": "2021-02-23", "localDeDestino": "Manaus", "regiao": "Norte" }' -H 'Content-Type: application/json' -H 'Authorization: eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJ1c3Vhcm1vQGVtYWlsLmNvbSIsInJvbGUiOiJST0xFX1VTUFSSU8iLCJjcmVhdGVkIjoxNjUyNzI4MTA2ODIzLCJleHAiOjE2NTI4MjgxMDV9.P_-6wJXYSaYiOrIVg-YpG008qWG817YRNH0oPTkoamZrq5mp3s2Q6EIwTKJ2hvpvoFs7M-3lkv0QEeq-h4Qrvrg'
```

```
Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (master)
$ curl -X POST -is http://localhost:8089/api/v1/viagens -d '{ "acompanhante": "Isabelle", "dataPartida": "2021-01-23", "dataRetorno": "2021-02-23", "localDeDestino": "Manaus", "regiao": "Norte" }' -H 'Content-Type: application/json' -H 'Authorization: eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJ1c3Vhcm1vQGVtYWlsLmNvbSIsInJvbGUiOiJST0xFX1VTUFSSU8iLCJjcmVhdGVkIjoxNjUyNzI4MTA2ODIzLCJleHAiOjE2NTI4MjgxMDV9.P_-6wJXYSaYiOrIVg-YpG008qWG817YRNH0oPTkoamZrq5mp3s2Q6EIwTKJ2hvpvoFs7M-3lkv0QEeq-h4Qrvrg'
HTTP/1.1 403
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Mon, 16 May 2022 19:12:33 GMT

{"timestamp":"2022-05-16T19:12:33.886+0000","status":403,"error":"Forbidden","exception":"org.springframework.security.access.AccessDeniedException","message":"Acesso negado","path":"/api/v1/viagens"}
```

Aula 08 - Header, Query, Path e Body - Como usar parâmetros em APIs Rest

Descobrindo como enviar dados para a API Rest

Header, Body,Query e Path. Esses são os meios de envio de dados, mas como eles funcionam na prática?

Consultando as viagens

```
curl -X GET -is http://localhost:8089/api/v1/viagens -H 'Authorization:  
eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJc3Vhcm1vQGVtYWlsLmNvbSIsInJvbGUiOiJST0xFX1VTUFSSU8iLCJjcmVhdGVkIjoxN  
jUyNzM0Mjg3ODI2LCJleHAIoJE2NTI4MzQyODZ9.FjUyUYjNsgUfN_DcBAnb3Mqzh0Flckb3KCMGUdozuFG6W-  
7N32g2ROH5ZftrVrtQjLVqCLJjqDS8mLPF8gsW0Q'
```

```
Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (master)  
$ curl -X GET -is http://localhost:8089/api/v1/viagens -H 'Authorization: eyJhbGciOiJIUzUxMiJ9.e  
yJzdWIiOiJc3Vhcm1vQGVtYWlsLmNvbSIsInJvbGUiOiJST0xFX1VTUFSSU8iLCJjcmVhdGVkIjoxNjUyNzM0Mjg3ODI2L  
CJleHAIoJE2NTI4MzQyODZ9.FjUyUYjNsgUfN_DcBAnb3Mqzh0Flckb3KCMGUdozuFG6W-7N32g2ROH5ZftrVrtQjLVqCLJj  
qDS8mLPF8gsW0Q'  
HTTP/1.1 200  
X-Content-Type-Options: nosniff  
X-XSS-Protection: 1; mode=block  
Cache-Control: no-cache, no-store, max-age=0, must-revalidate  
Pragma: no-cache  
Expires: 0  
X-Frame-Options: DENY  
Content-Type: application/json; charset=UTF-8  
Transfer-Encoding: chunked  
Date: Mon, 16 May 2022 20:52:23 GMT  
  
{"data": [{"id": 1, "localDeDestino": "Manaus", "dataPartida": "2021-01-23", "dataRetorno": "2021-02-23",  
"acompanhante": "Isabelle", "regiao": "Norte"}], "errors": []}
```

Enviando um parâmetro via query

Viagens - Região Norte

```
curl -X GET -is http://localhost:8089/api/v1/viagens?regiao=Norte -H 'Authorization:  
eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJc3Vhcm1vQGVtYWlsLmNvbSIsInJvbGUiOiJST0xFX1VTUFSSU8iLCJjcmVhdGVkIjoxN  
jUyNzM0Mjg3ODI2LCJleHAIoJE2NTI4MzQyODZ9.FjUyUYjNsgUfN_DcBAnb3Mqzh0Flckb3KCMGUdozuFG6W-  
7N32g2ROH5ZftrVrtQjLVqCLJjqDS8mLPF8gsW0Q'
```

```
Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (master)  
$ curl -X GET -is http://localhost:8089/api/v1/viagens?regiao=Norte -H 'Authorization: eyJhbGciOi  
IiUzUxMiJ9.eyJzdWIiOiJc3Vhcm1vQGVtYWlsLmNvbSIsInJvbGUiOiJST0xFX1VTUFSSU8iLCJjcmVhdGVkIjoxNjUyNz  
M0Mjg3ODI2LCJleHAIoJE2NTI4MzQyODZ9.FjUyUYjNsgUfN_DcBAnb3Mqzh0Flckb3KCMGUdozuFG6W-7N32g2ROH5ZftrV  
rtQjLVqCLJjqDS8mLPF8gsW0Q'  
HTTP/1.1 200  
X-Content-Type-Options: nosniff  
X-XSS-Protection: 1; mode=block  
Cache-Control: no-cache, no-store, max-age=0, must-revalidate  
Pragma: no-cache  
Expires: 0  
X-Frame-Options: DENY  
Content-Type: application/json; charset=UTF-8  
Transfer-Encoding: chunked  
Date: Mon, 16 May 2022 20:56:52 GMT  
  
{"data": [{"id": 1, "localDeDestino": "Manaus", "dataPartida": "2021-01-23", "dataRetorno": "2021-02-23",  
"acompanhante": "Isabelle", "regiao": "Norte"}], "errors": []}
```

Consultando uma viagem específica

```
curl -X GET -is http://localhost:8089/api/v1/viagens/1 -H 'Authorization: eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ1c3Vhcm1vQGVtYWlsLmNvbSIsInJvbGUiOiJST0xFX1VTVFSSU8iLCJjcmVhdGVkIjoxNjUyNzM0Mjg3ODI2LCJleHAiOjE2NTI4MzQyODZ9.FjUyUYjNsgUFN_DcBAnb3Mqzh0Flckb3KCMGUdozuFG6W-7N32g2ROH5ZftrVrtQjLVqCLJjqDS8mLPF8gsW0Q'
```

```
Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (master)
$ curl -X GET -is http://localhost:8089/api/v1/viagens/1 -H 'Authorization: eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ1c3Vhcm1vQGVtYWlsLmNvbSIsInJvbGUiOiJST0xFX1VTVFSSU8iLCJjcmVhdGVkIjoxNjUyNzM0Mjg3ODI2LCJleHAiOjE2NTI4MzQyODZ9.FjUyUYjNsgUFN_DcBAnb3Mqzh0Flckb3KCMGUdozuFG6W-7N32g2ROH5ZftrVrtQjLVqCLJjqDS8mLPF8gsW0Q'
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Mon, 16 May 2022 21:05:47 GMT

{"data": {"id": 1, "localDeDestino": "Manaus", "dataPartida": "2021-01-23", "dataRetorno": "2021-02-23", "acompanhante": "Isabelle", "regiao": "Norte", "temperatura": 30.0}, "errors": []}
Roberto@DESKTOP-HGDUAQ7 MINGW64 /d/github/gerenciador-viagens (master)
```

Aula 09 - Como usar o Insomnia para Testar APIs Rest

Manipulando APIs Rest com o uso de ferramentas de interface gráfica

O Insomnia é uma ferramenta de código aberto que tem o objetivo de facilitar o design, teste e entrega de APIs Rest.

- Baixe e instale o Insomnia Core for Windows.

<https://insomnia.rest/download>

Requisição para obter um token de acesso

- Clique no botão add (+) e selecione New Request

The screenshot shows the 'New Request' configuration window. It includes fields for 'Name' (set to 'Capturar o token do usuário'), 'Method' (set to 'POST'), and 'Content Type' (set to 'JSON'). A note at the bottom left says '* Tip: paste Curl command into URL afterwards to import it'. A 'Create' button is visible on the right.

- Clique no botão "No Environment" e selecione "Manage Environments"

Base Environment

```
1+ {
2   "servidor": "http://localhost:8089/api"
3 }
```

The screenshot shows the Insomnia application interface after a successful POST request. The top bar shows '200 OK', '4.19 s', '261 B', and 'Just Now'. The main area displays the JSON response body, which includes a 'data' key containing a token and an empty 'errors' array.

```
1+ {
2   "data": {
3     "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c3VhcmlvQGVtYwlsLmNvbS1sInJvbGUiOiJST0xFX1VTUFSSU8iLCJcmVhdGVkIjoxNjUyNzM3OTg4NjY2LC1leHAiOjE2NTI4Mzc500d9.MI-9qth1sL3Kmj4Njm0gzsjxY0LCUadyQ4FKEoc73au4uUrqa2k9PLcuWw2styPov1VrgNz58TE4S2xGt9Dng"
4   },
5   "errors": []
6 }
```

Buscando todas as viagens

New Request

Name (defaults to your request URL if left empty)

Buscar todas as viagens

GET

* Tip: paste Curl command into URL afterwards to import it

Create

Application Edit View Window Tools Help

Insomnia / New Document

DESIGN DEBUG TEST

No Environment Cookies

GET `[_servidor]/v1/viagens`

Send 200 OK 601 ms 154 B Just Now

Body Auth Query Header Docs

Authorization `eyJhbGciOiJIUzIwMiJ9.eyJzdWIiOiJcLjYhZG1pbk8lMkFpbG5jb20iLCJyb2xlijo1Uk9MRV9BRE13TiisInNyzWFew2Q1QjE2NTI4OTYwM250MS1mV4CICIGWYUjKSNjQzN30.cba4UGVtZ6jC06Us8MKu1wSGenKAN7AS3oprnnas5eHxttggtAot:geagEM1LvrYcvT5yZogag3A9c_14ey3mQ`

Preview Header Cookie Timeline

New header New value

```
1+ {
2+   "data": [
3+     {
4+       "id": 1,
5+       "localDestino": "Manaus",
6+       "dataPartida": "2021-01-23",
7+       "dataRetorno": "2021-02-23",
8+       "acompanhante": "Isabelle",
9+       "regiao": "Norte"
10+    },
11+   ],
12+   "errors": []
13+ }
```

Os dados das requisições são guardados em memória, isso significa que se você sair do IntelliJ parando de executar o maven, todos eles serão perdidos.

Para reiniciar o maven no IntelliJ:

`mvn spring-boot:run`

Gerando token de administrador

New Request

Name (defaults to your request URL if left empty)

Capturar o token do administrador

POST No Body

* Tip: paste Curl command into URL afterwards to import it

Create

Application Edit View Window Tools Help

Insomnia / New Document

DESIGN DEBUG TEST

No Environment Cookies

POST `[_servidor]/v1/auth`

Send 200 OK 11.1 s 256 B Just Now

Auth Query Header Docs

JSON

```
1+ {
2+   "data": {
3+     "token": "eyJhbGciOiJIUzIwMiJ9.eyJzdWIiOiJcLjYhZG1pbk8lMkFpbG5jb20iLCJyb2xlijo1Uk9MRV9BRE13TiisInNyzWFew2Q1QjE2NTI4OTYwM250MS1mV4CICIGWYUjKSNjQzN30.cba4UGVtZ6jC06Us8MKu1wSGenKAN7AS3oprnnas5eHxttggtAot:geagEM1LvrYcvT5yZogag3A9c_14ey3mQ"
4+   },
5+   "errors": []
6+ }
```

Cadastrando uma viagem como administrador

New Request X

Name (defaults to your request URL if left empty)
Cadastrar viagem POST ▼ JSON ▼

* Tip: paste Curl command into URL afterwards to import it Create

Application Edit View Window Tools Help

DESIGN DEBUG TEST

No Environment ▼ Cookies

POST ▼ `...servidor /v1/viagens` Send

Filter ▼ + ▼

Viagens

POST Cadastrar viagem

POST Capturar o token do administrador...

POST Capturar o token do usuário

GET Buscar todas as viagens

JSON ▼ Auth ▼ Query Header ▼ Docs

```
1+ {
2   "acompanhante": "Isabelle",
3   "dataPartida": "2021-01-23",
4   "dataRetorno": "2021-02-23",
5   "localDeDestino": "Manaus",
6   "regiao": "Norte"
7 }
```

Edit Tag X

Function to Perform
Response – reference values from other request's responses

Attribute
Body Attribute – value of response body ▼ ▼ ⚙️

Request
[Viagens] POST Capturar o token do administrador ▼ ▼

Filter (JSONPath or XPath)
`$data.token|` ⚙️

Trigger Behavior ?
Never – never resend request ▼ ▼ ⚙️

Live Preview refresh ⟳

```
eyJhbGciOiJIUzUxMiJ9.eyJzdWlOijhZG1pbk8lbWFpbC5jb20iLCJyb2xljoUi9MRV9BRE1JTlsImNyZWFOZWQiOjE2NTI4OTY0MzM5OTMsImV4cCI6MTY1Mjk5NjQzM30.eb4UGVtZ6JcO6Us0WKu1w5GErKAN7A53oprnna5eHxotggtAotzgeaGEMILVrYCvtT5yZdgag3A9c_T40y3mQ
```

Done

Application Edit View Window Tools Help

Insomnia / New Document ▾ DESIGN DEBUG TEST

No Environment ▾ Cookies

POST ▾ `_.servidor/v1/viagens`

Send 201 Created 368 ms 158 B 4 Minutes Ago ▾

Preview ▾ Header 10 Cookie Timeline

JSON ▾ Auth ▾ Query Header ② Docs

```
1+ {
2 "acompanhante": "Isabelle",
3 "dataPartida": "2021-01-23",
4 "dataRetorno": "2021-02-23",
5 "localDeDestino": "São Paulo",
6 "regiao": "Sudeste"
7 }
```

1+ {
2 "data": {
3 "id": 1,
4 "localDeDestino": "São Paulo",
5 "dataPartida": "2021-01-23",
6 "dataRetorno": "2021-02-23",
7 "acompanhante": "Isabelle",
8 "regiao": "Sudeste"
9 },
10 "errors": []
11 }

This screenshot shows the Insomnia API client interface. The left sidebar contains a tree view with a 'Viagens' folder, and two items under it: 'Capturar o token do administr...' and 'Cadastrar viagem'. The main panel shows a POST request to the endpoint `_.servidor/v1/viagens`. The request body is a JSON object with fields: 'acompanhante' (Isabelle), 'dataPartida' (2021-01-23), 'dataRetorno' (2021-02-23), 'localDeDestino' (São Paulo), and 'regiao' (Sudeste). The response is a 201 Created status with a response time of 368 ms and a response size of 158 B. The response body is also a JSON object with a single entry for 'data' and an empty 'errors' array.

Cadastrando novas viagens

Application Edit View Window Tools Help

Insomnia / New Document ▾ DESIGN DEBUG TEST

No Environment ▾ Cookies

POST ▾ `_.servidor/v1/viagens`

Send 201 Created 82.5 ms 157 B Just Now ▾

Preview ▾ Header 10 Cookie Timeline

JSON ▾ Auth ▾ Query Header ② Docs

```
1+ {
2 "acompanhante": "Priscilla",
3 "dataPartida": "2022-01-23",
4 "dataRetorno": "2022-02-23",
5 "localDeDestino": "Porto Alegre",
6 "regiao": "Sul"
7 }
```

```
1+ {
2 "data": {
3 "id": 2,
4 "localDeDestino": "Porto Alegre",
5 "dataPartida": "2022-01-23",
6 "dataRetorno": "2022-02-23",
7 "acompanhante": "Priscilla",
8 "regiao": "Sul"
9 },
10 "errors": []
11 }
```

This screenshot shows the Insomnia API client interface. The left sidebar contains a tree view with a 'Viagens' folder, and two items under it: 'Capturar o token do administr...' and 'Cadastrar viagem'. The main panel shows a POST request to the endpoint `_.servidor/v1/viagens`. The request body is a JSON object with fields: 'acompanhante' (Priscilla), 'dataPartida' (2022-01-23), 'dataRetorno' (2022-02-23), 'localDeDestino' (Porto Alegre), and 'regiao' (Sul). The response is a 201 Created status with a response time of 82.5 ms and a response size of 157 B. The response body is a JSON object with a single entry for 'data' and an empty 'errors' array.

Application Edit View Window Tools Help

Insomnia / New Document ▾ DESIGN DEBUG TEST

No Environment ▾ Cookies

POST ▾ `_.servidor/v1/viagens`

Send 201 Created 36.7 ms 152 B Just Now ▾

Preview ▾ Header 10 Cookie Timeline

JSON ▾ Auth ▾ Query Header ② Docs

```
1+ {
2 "acompanhante": "Isabelle",
3 "dataPartida": "2021-05-23",
4 "dataRetorno": "2021-06-23",
5 "localDeDestino": "Manaus",
6 "regiao": "Norte"
7 }
```

```
1+ {
2 "data": {
3 "id": 3,
4 "localDeDestino": "Manaus",
5 "dataPartida": "2021-05-23",
6 "dataRetorno": "2021-06-23",
7 "acompanhante": "Isabelle",
8 "regiao": "Norte"
9 },
10 "errors": []
11 }
```

This screenshot shows the Insomnia API client interface. The left sidebar contains a tree view with a 'Viagens' folder, and two items under it: 'Capturar o token do administr...' and 'Cadastrar viagem'. The main panel shows a POST request to the endpoint `_.servidor/v1/viagens`. The request body is a JSON object with fields: 'acompanhante' (Isabelle), 'dataPartida' (2021-05-23), 'dataRetorno' (2021-06-23), 'localDeDestino' (Manaus), and 'regiao' (Norte). The response is a 201 Created status with a response time of 36.7 ms and a response size of 152 B. The response body is a JSON object with a single entry for 'data' and an empty 'errors' array.

Gerando token de usuário

Application Edit View Window Tools Help

Insomnia / New Document

DESIGN DEBUG TEST

No Environment Cookies

POST `[_servidor]/v1/auth` Send 200 OK 440 ms 261 B Just Now

JSON Auth Query Header Docs

```
1+ {
2+   "email": "usuario@email.com",
3+   "senha": "123456"
4+ }
```

Preview Header Cookie Timeline

```
1+ {
2+   "data": {
3+     "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c3VcmVwb3J0eXVtVUFSUS8iLCJmYm94IjoxMjQyNzAwMDkzLjEwMjE0IjewNTISOTY4OTI9.XMxmcExHwzlwB9cyU7G78kjQg"
4+   },
5+   "errors": []
6+ }
```

Buscar todas as viagens como usuário autenticado

Application Edit View Window Tools Help

Insomnia / New Document

DESIGN DEBUG TEST

No Environment Cookies

GET `[_servidor]/v1/viagens` Send 200 OK 41 ms 429 B Just Now

Body Auth Query Header Docs

```
1+ {
2+   "data": [
3+     {
4+       "id": 1,
5+       "localDeDestino": "São Paulo",
6+       "dataPartida": "2021-01-23",
7+       "dataRetorno": "2021-02-23",
8+       "acompanhante": "Isabelle",
9+       "regiao": "Sudeste"
10+    },
11+    {
12+      "id": 2,
13+      "localDeDestino": "Porto Alegre",
14+      "dataPartida": "2022-01-23",
15+      "dataRetorno": "2022-02-23",
16+      "acompanhante": "Priscilla",
17+      "regiao": "Sul"
18+    },
19+    {
20+      "id": 3,
21+      "localDeDestino": "Manaus",
22+      "dataPartida": "2021-05-23",
23+      "dataRetorno": "2021-06-23",
24+      "acompanhante": "Isabelle",
25+      "regiao": "Norte"
26+    }
27+  ],
28+  "errors": []
29+ }
```

Buscar uma viagem

Application Edit View Window Tools Help

Insomnia / New Document

DESIGN DEBUG TEST

No Environment Cookies

GET `[_servidor]/v1/viagens/2` Send 422 Unprocessable Entity 324 ms 59 B Just Now

Body Auth Query Header Docs

```
1+ {
2+   "data": null,
3+   "errors": [
4+     "A API do Tempo não está online"
5+   ]
6+ }
```

Preview Header Cookie Timeline

Buscar viagem por região

The screenshot shows the Insomnia REST client interface. The top navigation bar includes Application, Edit, View, Window, Tools, Help, DESIGN, DEBUG (which is selected), and TEST. The main area has tabs for No Environment, Cookies, and a sidebar for Body, Auth, Query, Header, and Docs. The URL bar shows `GET /v1/viagens?regiao=Sul`. The response status is 200 OK, with 187 ms and 159 B. The preview tab shows the JSON response:

```
1+ {
2+   "data": [
3+     {
4+       "id": 2,
5+       "localDestino": "Porto Alegre",
6+       "dataPartida": "2022-01-23",
7+       "dataRetorno": "2022-02-23",
8+       "acompanhante": "Priscilla",
9+       "regiao": "Sul"
10+    }
11+ ],
12+ "errors": []
13+ }
```

The sidebar on the left lists several API endpoints under the 'Viagens' category, including 'Capturar o token do administrador...', 'Cadastrar viagem', 'Capturar o token do usuário', 'Buscar todas as viagens', 'Buscar uma viagem', and 'Buscar viagem por região'. The bottom of the interface features a placeholder for entering a URL and sending a response, along with a note to select a body type for requests.

Aula 10 - Como usar o Postman para testar APIs Rest

- Baixe e instale o [Postman](#).

- Clique no menu **Collections** e em **Create a new Collection**. Renomeie a collection para **Gerenciador de Viagens**.

The screenshot shows the Postman interface with a new collection named "Gerenciador de Viagens". The left sidebar has options for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main area shows an overview of the collection with a single GET request to "http://localhost:8000". The "Authorization" tab is selected, showing "No Auth". A note states: "This collection does not use any authorization. Learn more about authorization".

- Adicione uma nova requisição ([Add Request](#)):

The screenshot shows a POST request to "http://localhost:8089/api/v1/auth" with the following body:

```
1 ... "email": "admin@email.com",
2 ... "senha": "654321"
```

The response status is 200 OK, time 933 ms, size 568 B.

- Crie uma nova variável ([new environment](#)):

The screenshot shows the "Globals" section of the workspace. It lists a variable "servidor" with a value of "default" and "http://localhost:8089/api". There is also a "New Environment" button.

- Clique no botão "Save"

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Collections, APIs, Environments, Mock Servers, Monitors, History.
- Top Bar:** File, Edit, View, Help, Workspaces, Reports, Explore, Search Postman, Sign In, Create Account.
- Central Area:**
 - Collection:** Gerenciador de Viagens / Fazer login como administrador
 - Request:** POST {{servidor}}/v1/auth
 - Body:** JSON (Pretty)

```
1   "email": "admin@email.com",
2   ....
3   "senha": "654321"
4 }
```
 - Response:** Status: 200 OK, Time: 509 ms, Size: 568 B
 - Body (Raw):**

```
{"data": {
  "token": "eyJhbGciOiJIUzI1njc3Ij0K.eyJzdWIiOiJhZG1pbkB1bWFnC5jb20iLCJyb2x1IjoiUk9MRV9BRE1JTlIsImNyZWF0ZWQiOjE2NTI4OTk2MTUyMjYsImV4cCI6MTY1Mjk0TYxNH0.aab0a8EVDBLg7RhK-toCPMnNSkhQHJGL8vuyqNjVoIUsf1F0asjbE0pyWzVkJUyzYz3ScS40KqU3GtCuqxCUw"
},
"errors": []}
```

- Salve a request (Save).

Cadastrar viagem

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Collections, APIs, Environments, Mock Servers, Monitors, History.
- Top Bar:** File, Edit, View, Help, Workspaces, Reports, Explore, Search Postman, Sign In, Create Account.
- Central Area:**
 - Collection:** Gerenciador de Viagens / Cadastrar viagem
 - Request:** POST {{servidor}}/v1/viagens
 - Body:** JSON (Pretty)

```
1   "acompanhante": "Priscila",
2   "dataPartida": "2021-02-06",
3   "dataRetorno": "2021-03-06",
4   "localDeDestino": "Paraíba",
5   "regiao": "Nordeste"
6 }
```

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Collections, APIs, Environments, Mock Servers, Monitors, History.
- Top Bar:** File, Edit, View, Help, Workspaces, Reports, Explore, Search Postman, Sign In, Create Account.
- Central Area:**
 - Collection:** Gerenciador de Viagens / Cadastrar viagem
 - Request:** POST {{servidor}}/v1/viagens
 - Headers:** Headers (10)

KEY	VALUE	DESCRIPTION	Bulk Edit	Presets
Authorization	eyJhbGciOiJIUzI1njc3Ij0K.eyJzdWIiOiJhZG1pbkB1bWFnC5jb20iLCJyb2x1IjoiUk9MRV9BRE1JTlIsImNyZWF0ZWQiOjE2NTI4OTk2MTUyMjYsImV4cCI6MTY1Mjk0TYxNH0.aab0a8EVDBLg7RhK-toCPMnNSkhQHJGL8vuyqNjVoIUsf1F0asjbE0pyWzVkJUyzYz3ScS40KqU3GtCuqxCUw			
Content-Type	application/json			
Key	Value	Description		

File Edit View Help

Home Workspaces Reports Explore Search Postman Sign In Create Account

Scratch Pad New Import Overview GET http://127.0.0.1:8080/ Global Gerenciador de Viagens POST Fazer login como administrador POST Cadastrar + ... No Environment

Collections APIs Environments Mock Servers Monitors History

Gerenciador de Viagens / Cadastrar viagem

POST {{servidor}}/v1/viagens

Params Authorization Headers (10) Body ● Pre-request Script Tests Settings Cookies

Headers 8 hidden

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
Authorization	eyJhbGciOiJIUzIwMjQ.eyJzdWIiOiJhZG1pbkB... <input checked="" type="checkbox"/>				
Content-Type	application/json <input checked="" type="checkbox"/>				
Key	Value	Description			

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

Status: 201 Created Time: 1493 ms Size: 524 B Save Response

```
1 "data": {  
2   "id": 4,  
3   "localDeDestino": "Paraíba",  
4   "dataPartida": "2021-02-06",  
5   "dataRetorno": "2021-03-06",  
6   "acompanhante": "Priscila",  
7   "regiao": "Nordeste"  
8 },  
9 }
```

- Salve a request (Save).

- Na requisição **Fazer login como administrador**, crie uma variável global chamada **token**.

The screenshot shows the Postman interface with the 'Scratch Pad' tab selected. In the left sidebar, 'Collections' is expanded, showing 'Globals'. Under 'Globals', there is a table with two rows. The first row has 'servidor' checked, 'default' as the type, and 'http://localhost:8089/api' as the initial value. The second row has 'token' checked, 'default' as the type, and an empty current value. A button 'Add a new variable' is visible at the bottom of the table.

- Clique na aba **Tests** e insira o código:

```
const resposta = pm.response.json();
pm.globals.set("token", resposta.data.token);
```

The screenshot shows the Postman interface with a collection named 'Gerenciador de Viagens'. Inside the collection, there is a POST request for 'Fazer login como administrador'. The 'Tests' tab is selected, containing the following JavaScript code:

```
1 const resposta = pm.response.json();
2 pm.globals.set("token", resposta.data.token);
```

The 'Body' tab shows the response body in JSON format, which includes a 'data' object with a 'token' key and a long string value. The status bar at the bottom indicates a successful 200 OK response.

- Clique no botão **"Save"**.

- Clique no botão **"Send"**.

- Na requisição **Cadastrar Viagem**, substitua o token por **{{token}}**

The screenshot shows the Postman interface with a POST request for '({servidor})/v1/viagens'. The 'Headers' tab is selected, showing the following configuration:

KEY	VALUE	DESCRIPTION	Bulk Edit	Presets
<input checked="" type="checkbox"/> Content-Type	application/json			
<input checked="" type="checkbox"/> Authorization	{{token}}			

- Clique no botão "Send"

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main area displays a collection named 'Gerenciador de Viagens / Cadastrar viagem'. A POST request is selected with the URL `((servidor))/v1/viagens`. The Headers tab is active, showing two entries: 'Content-Type' set to 'application/json' and 'Authorization' set to `((token))`. The Body tab contains a JSON response with the following data:

```
1
2   "data": {
3     "id": 8,
4     "localDeDestino": "Paraíba",
5     "dataPartida": "2021-02-06",
6     "dataRetorno": "2021-03-06",
7     "acompanhante": "Priscila",
8     "regiao": "Nordeste"
9   },
10  "errors": []
```

The status bar at the bottom indicates a 201 Created response with a time of 799 ms and a size of 524 B.

Tests

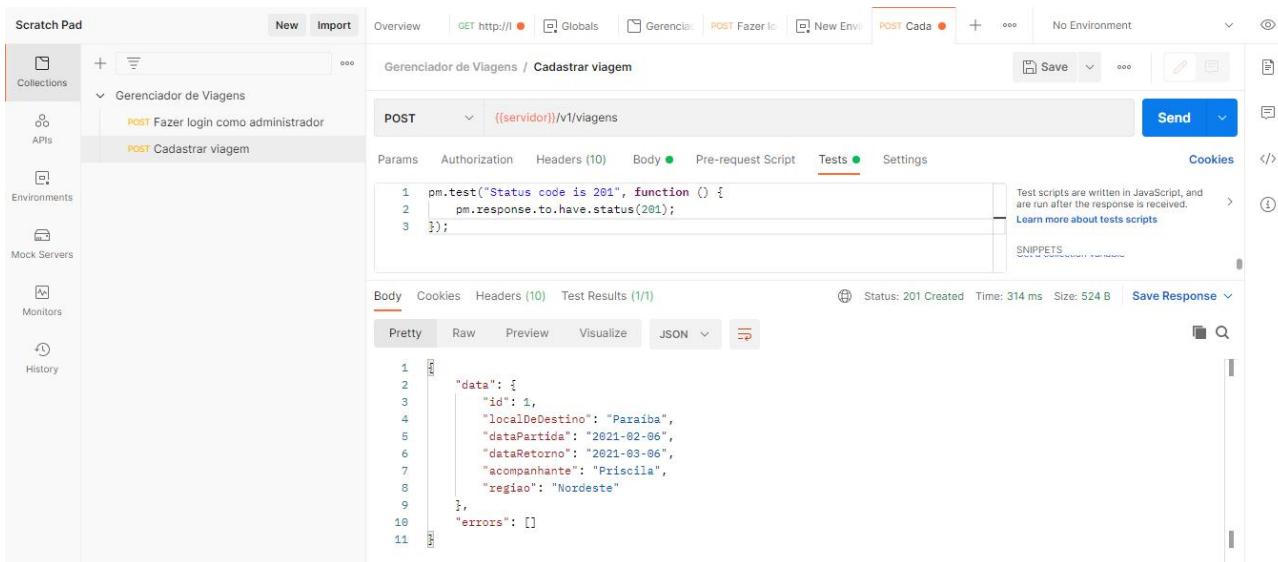
Na requisição "Cadastrar viagem", selecione a aba Tests e selecione o Snippet Status code: Code is 200. Altere para 201.



The screenshot shows the Postman interface with the 'Tests' tab selected for a POST request to `/v1/viagens`. The test script is set to check for a status code of 201:

```
1 pm.test("Status code is 201", function () {
2     pm.response.to.have.status(201);
3 })
```

- Clique no botão "Send":



The screenshot shows the Postman interface after sending the request. The status is `201 Created`, and the response body is displayed in JSON format:

```
1 "data": {
2     "id": 1,
3     "localDeDestino": "Paraíba",
4     "dataPartida": "2021-02-06",
5     "dataRetorno": "2021-03-06",
6     "acompanhante": "Priscila",
7     "regiao": "Nordeste"
8 },
9 "errors": []
```

- Clique na aba Test Results:



The screenshot shows the 'Test Results' section with one result listed:

Body	Cookies	Headers (10)	Test Results (1/1)
All	Passed	Skipped	Failed
PASS	Status code is 201		

Aula 11 - Como automatizar testes de API Rest com RestAssured

Automatizando testes de API Rest usando Java e JUnit

Pela primeira vez vamos criar scripts de teste automatizado capazes de enviar requisições, receber respostas e realizar asserções nelas.

junit maven

<https://mvnrepository.com/artifact/junit/junit/4.12>

Use a versão 4.12

The screenshot shows the Maven Repository interface. At the top, there's a search bar with the placeholder "Search for groups, artifacts, categories" and a "Search" button. Below the search bar, the URL "https://mvnrepository.com/artifact/junit/junit/4.12" is visible. To the left, there's a sidebar titled "Popular Categories" listing various Java libraries like Aspect Oriented, Actor Frameworks, etc. The main content area has a title "JUnit > 4.12" and a sub-section "JUnit is a unit testing framework for Java, created by Erich Gamma and Kent Beck." It displays various metadata: License (EPL 1.0), Categories (Testing Frameworks), Organization (JUnit), HomePage (http://junit.org), Date (Dec 04, 2014), Files (pom (23 KB), jar (307 KB), View All), Repositories (Central, Lutece Paris, Redhat GA, Sonatype, Xceptance), Used By (117,210 artifacts), and Vulnerabilities (Direct vulnerabilities: CVE-2020-15250). A note at the bottom says "Note: There is a new version for this artifact" with links to "New Version" (4.13.2) and "View All". At the bottom, there's a code snippet for Maven dependencies:

```
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

restassured maven

Pegar a versão 3.0.0

<https://mvnrepository.com/artifact/io.rest-assured/rest-assured/3.0.0>



REST Assured > 3.0.0

Java DSL for easy testing of REST services

License	Apache 2.0
Categories	Testing Frameworks
HomePage	http://code.google.com/p/rest-assured
Date	(Jun 03, 2016)
Files	pom (4 KB) jar (575 KB) View All
Repositories	Central Mulesoft Sonatype
Used By	1,988 artifacts
Vulnerabilities from dependencies:	
CVE-2021-20190	
Vulnerabilities	CVE-2020-9548 CVE-2020-9547 View 50 more ...

Note: There is a new version for this artifact

New Version

5.0.1

[Maven](#) [Gradle](#) [Gradle \(Short\)](#) [Gradle \(Kotlin\)](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#) [Buildr](#)

```
<!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured -->
<dependency>
  <groupId>io.rest-assured</groupId>
  <artifactId>rest-assured</artifactId>
  <version>3.0.0</version>
  <scope>test</scope>
</dependency>
```

Include comment with link to declaration

arquivo pom.xml

A tag **dependencies** possui todas as dependências que o projeto necessita.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.montanha</groupId>
  <artifactId>gerenciador-viagens-montanha</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>GerenciadorViagensMontanha</name>
  <description>Gerenciador de Viagens</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.10.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>com.h2database</groupId>
      <artifactId>h2</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
```

```

<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.7.0</version>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.9.2</version>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.9.2</version>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>3.0.0</version>
    <scope>test</scope>
</dependency>
</dependencies>
```

```

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
</project>
```

Clique com o botão direito do mouse sobre o arquivo **pom.xml** e selecione **Maven --> Reload project**.

- Quando o RestAzurred for baixado os textos em vermelho em dependency ficaram cinza claro.
- Na pasta **test**, na subpasta **Java**, crie um **new package**.

com.montanha.gerenciador.viagens

- Dentro da pasta **viagens** crie uma nova classe chamada **ViagensTest**.

<https://rest-assured.io/>

```
{
  "lotto": {
    "lottoId": 5,
    "winning-numbers": [2, 45, 34, 23, 7, 5, 3],
    "winners": [
      {
        "winnerId": 23,
        "numbers": [2, 45, 34, 23, 3, 5]
      },
      {
        "winnerId": 54,
        "numbers": [52, 3, 12, 11, 18, 22]
      }
    ]
  }
}
```

- Clique em **Docs**.
- E clique no link **GettingStarted**.
- Clique no link **Static Imports**.
- Copie as seguintes linhas:

```
io.restassured.RestAssured.*  
io.restassured.matcher.RestAssuredMatchers.*  
org.hamcrest.Matchers.*
```

ViagensTest.java

```
package com.montanha.gerenciador.viagens;

import io.restassured.http.ContentType;
import org.junit.Test;
import static io.restassured.RestAssured.*;
import static io.restassured.matcher.RestAssuredMatchers.*;
import static org.hamcrest.Matchers.*;

public class ViagensTest {
    @Test
    public void testDadoUmAdministradorQuandoCadastroViagensEntaoObtenhoStatusCode201(){
        // Configurar o caminho comum de acesso a minha API Rest
        baseURI = "http://localhost";
        port = 8089;
        basePath = "/api";

        // Login na API Rest como administrador
        String token = given()
            .body("{\n" +
                "   \"email\": \"admin@email.com\",\n" +
                "   \"senha\": \"654321\"\n" +
                "}")
            .contentType(ContentType.JSON)
            .when()
            .post("/v1/auth")
            .then()
            .log().all()
            .extract()
            .path("data.token");

        System.out.println(token);

        // Cadastrar a viagem
    }
}
```

Rode o teste ([Run Test](#))

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Fri, 20 May 2022 13:14:09 GMT

{
  "data": {
    "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbkBlbWFpbC5jb20iLCJyb2xlIjoiUk9MRV9BRE1JTiIsImNyZWFOZWQi0jE2NTMwNTION
  },
  "errors": [
  ]
}
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbkBlbWFpbC5jb20iLCJyb2xlIjoiUk9MRV9BRE1JTiIsImNyZWFOZWQi0jE2NTMwNTI0NDc1NjIsImV4cCI6MTY
Process finished with exit code 0
```

Aula 12 - Como fazer estratégias de Testes de API Rest

Testar não é apenas sobre interpretação, é sobre planejamento e estratégia

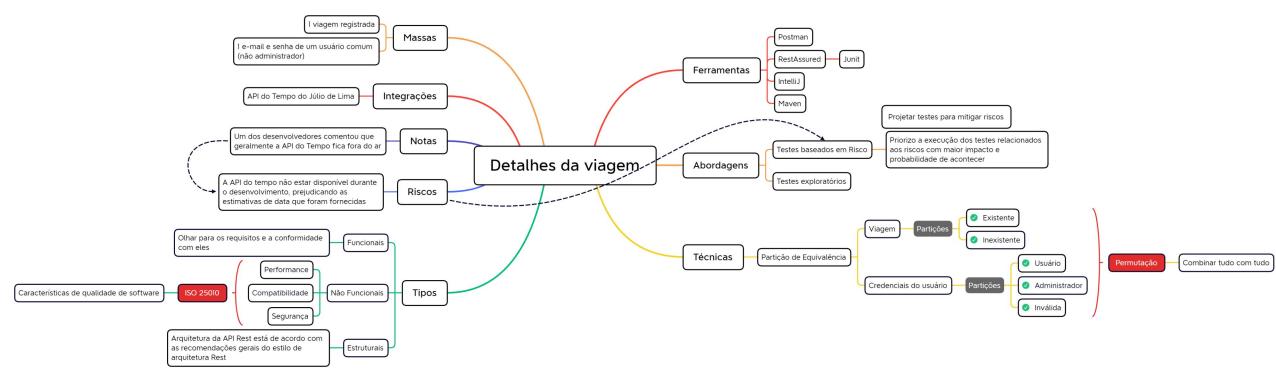
Uma descrição de alto nível descrevendo os requisitos relacionados a necessidade de testes.

Mapa mental (Mind Map)

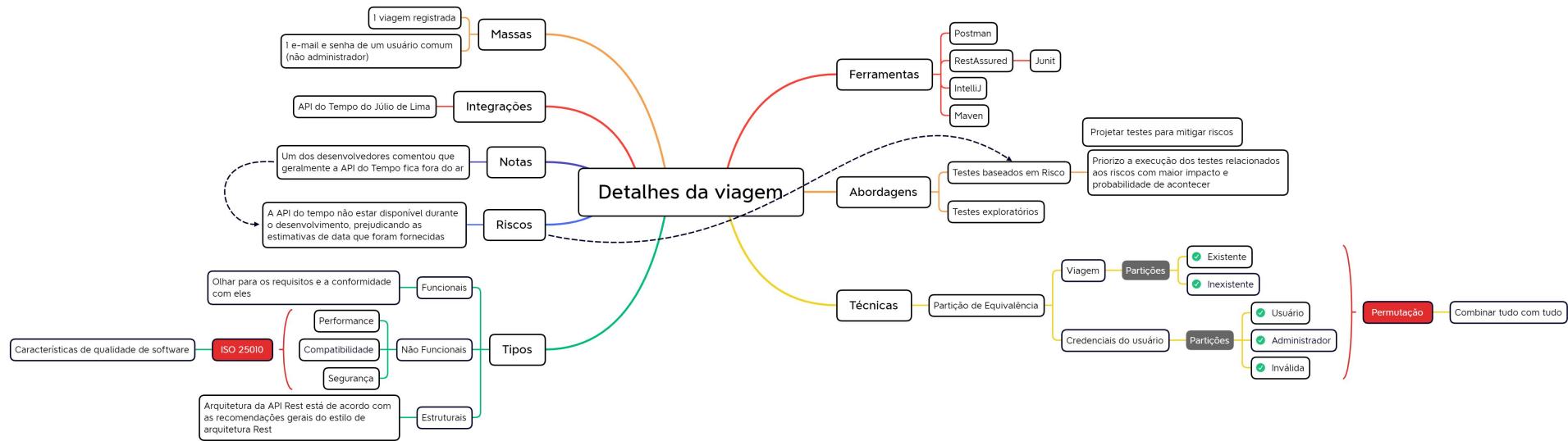
Software XMind

<https://www.xmind.net/download/>

- Baixe e instale o XMind para Windows.



Mapa Mental - Detalhes da viagem



Presented with XMind

Aula 13 - Como medir a Cobertura de Testes de API Rest

Afinal, precisamos medir a extensão dos nossos testes, não é mesmo?

Uma análise de Martin-Lopez et al. sobre como pode ser calculada a cobertura de testes de APIs Rest com base em inputs e outputs.

Abaixo os links mencionados na aula:

Workshop Alberto Martin-Lopez, Sergio Segura, Antonio Ruiz-Cortés

https://personal.us.es/amarlop/wp-content/uploads/2019/09/Test_Coverage_Criteria_for_RESTful_Web_APIS.pdf

Artigo da Nayara Crema (Como verificar a cobertura de testes de APIs REST)

<https://medium.com/revista-dtar/como-verificar-a-cobertura-de-testes-da-api-rest-9e2f745564b>

Há vários tipos de Cobertura de Testes como:

Cobertura de testes com base em requisitos;

Cobertura de testes com base em riscos, etc.

A abordagem de Martin-Lopez trata especificamente sobre cobrir a interface da API Rest.

Swagger é apenas uma das formas de descrever qual é a interface da API Rest. Há outras formas.

Cobertura de entrada e de saída

Algumas vezes, no projeto em que estamos inseridos na empresa, precisamos passar para o time uma clareza sobre o que está sendo implementado nos testes automatizados de API REST ou quanto da API está coberta pelos testes.

Existe uma forma de calcular a cobertura de testes, que se baseia nos critérios de cobertura de entrada (Input Coverage) e cobertura de saída (Output Coverage).

Path Coverage (input)

Verifica a cobertura da suíte de testes de acordo com os endpoints que a API possui. Este critério é importante, pois ao receber uma solicitação, o programa pode executar caminhos diferentes, então precisamos garantir que os endpoints da API REST estão cobertos pelos testes.

A análise é realizada pela quantidade de URI(URL + URN (Resource name)) que a API possui (se for a mesma URI para métodos diferentes, considera-se apenas um).

O ideal é realizar ao menos uma requisição para verificar cada endpoint.

Se verificar a imagem do swagger abaixo, podemos notar 13 endpoints diferentes:

Os testes de cobertura de saída consideram o status code recebido de uma requisição enviada.

pet Everything about your Pets

Find out more: <http://swagger.io>

POST /pet/{petId}/uploadImage	uploads an image	1	
POST /pet	Add a new pet to the store	2	
PUT /pet	Update an existing pet		
GET /pet/findByStatus	Finds Pets by status	3	
<small>/pet/findByTags - Finds Pets by tags</small>			
GET /pet/{petId}	Find pet by ID	4	
PUT /pet/{petId}	Updates a pet in the store with form data		
DELETE /pet/{petId}	Deletes a pet		
store Access to Petstore orders			
POST /store/order	Place an order for a pet	5	
GET /store/order/{orderId}	Find purchase order by ID	6	
DELETE /store/order/{orderId}	Delete purchase order by ID		
GET /store/inventory	Returns pet inventories by status	7	
user Operations about user			
Find out more about our store: http://swagger.io			
POST /user/createWithArray	Creates list of users with given input array	8	
POST /user/createWithList	Creates list of users with given input array	9	
GET /user/{username}	Get user by user name	10	
PUT /user/{username}	Updated user		
DELETE /user/{username}	Delete user		
GET /user/login	Logs user into the system	11	
GET /user/logout	Logs out current logged in user session	12	
POST /user	Create user	13	

Suponha que a automação desta API tenha apenas 6 desses endpoints implementados, mas a API possui 13 endpoints.

Para calcular a cobertura: quantidade de testes automatizados / quantidade de endpoints na API REST.

$$6 / 13 = 0,46$$

Então 46% dos testes de path estão cobertos pela automação.

Aula 14 - Como fazer Testes Funcionais em API Rest

Será que essa função da API Rest corresponde ao que foi requisitado?

Pois é, chegou a hora de analisar como o software está se comportando.

O Swagger é uma documentação da interface existente, mas não aborda regras de negócio.

Testes funcionais validam as funções do software.

Alguns testes funcionais do Gerenciador de Viagens

Regras de negócio

1) Apenas administradores podem registrar novas viagens.

Entradas: Credencial (Administrador = Registrar, Usuário = Não registrar, Inválida = Não registrar e Expirada = Não registrar).

Devem ser consideradas todas as entradas possíveis.

Testes funcionais a serem feitos:

- O software deve permitir o cadastro de uma viagem pelo administrador.

```
{  
  "data": {  
    "id": 3,  
    "localDeDestino": "Manaus",  
    "dataPartida": "2021-05-23",  
    "dataRetorno": "2021-06-23",  
    "acompanhante": "Isabelle",  
    "regiao": "Norte"  
  },  
  "errors": []  
}
```

- O software não deve permitir o cadastro de uma viagem por um usuário.

```
{  
  "timestamp": "2022-05-21T15:06:22.023+0000",  
  "status": 403,  
  "error": "Forbidden",  
  "exception": "org.springframework.security.access.AccessDeniedException",  
  "message": "Acesso negado",  
  "path": "/api/v1/viagens"  
}
```

- O software não deve permitir o cadastro de uma viagem com credenciais inválidas.

The screenshot shows a Postman interface with a POST request to `./servidor/v1/viagens`. The request includes two headers: `Content-Type: application/json` and `Authorization: abc123`. The response status is **401 Unauthorized**, with a timestamp of `2022-05-21T16:01:19.681+0000`, status code `401`, error `"Unauthorized"`, message `"Acesso negado. Você deve estar autenticado no sistema para acessar a URL solicitada."`, and path `"/api/v1/viagens"`.

```

1: {
2:   "timestamp": "2022-05-21T16:01:19.681+0000",
3:   "status": 401,
4:   "error": "Unauthorized",
5:   "message": "Acesso negado. Você deve estar autenticado no sistema para acessar a URL solicitada.",
6:   "path": "/api/v1/viagens"
7: }

```

- O software não deve permitir o cadastro de uma viagem com credenciais expiradas.

Ou seja, utilizando um token expirado o cadastro não deve ser realizado.

2) Apenas usuários podem listar viagens.

Testes funcionais a serem realizados:

- O software deve permitir que apenas usuários listem viagens.
- O software não deve permitir que um administrador liste viagens.
- O software não deve permitir que credenciais inválidas listem viagens.
- O software não deve permitir que credenciais expiradas listem viagens.

Aula 15 - Como fazer Testes de Performance em API Rest

Como é que a API Rest responde a esse número elevado de Requests?

Há ao menos duas formas de descobrir isso: chamando todos os seus amigos do trabalho e pedindo que eles acessem a API Rest, ou sendo prático e usando uma ferramenta.

Testes de performance não são testes funcionais.

Testes de perfomance é um termo muito abrangente, abaixo dele, temos:

- Testes de desempenho
- Testes de carga
- Testes de stress e muitos outros

Vamos utilizar como testes de desempenho onde veremos o tempo de resposta de uma API Rest.

Baixando e instalando o Apache JMeter

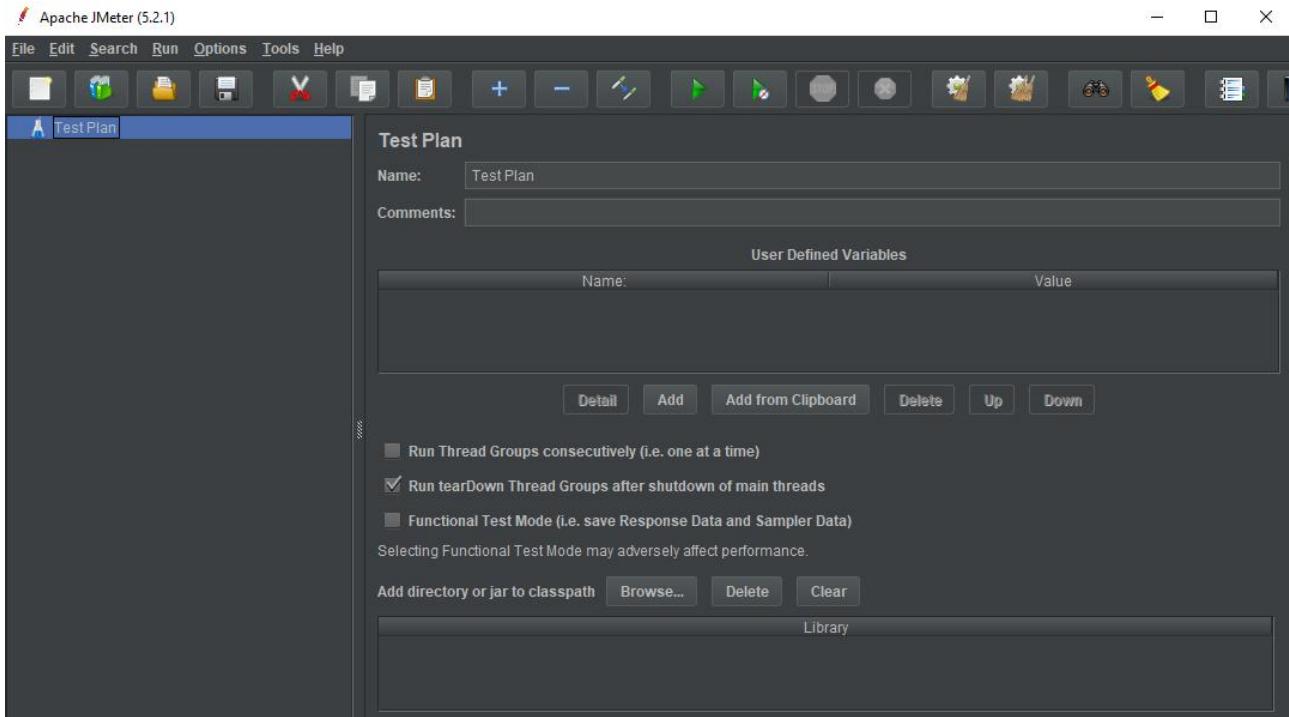
Apache JMeter 5.2.1

<https://archive.apache.org/dist/jmeter/binaries/>

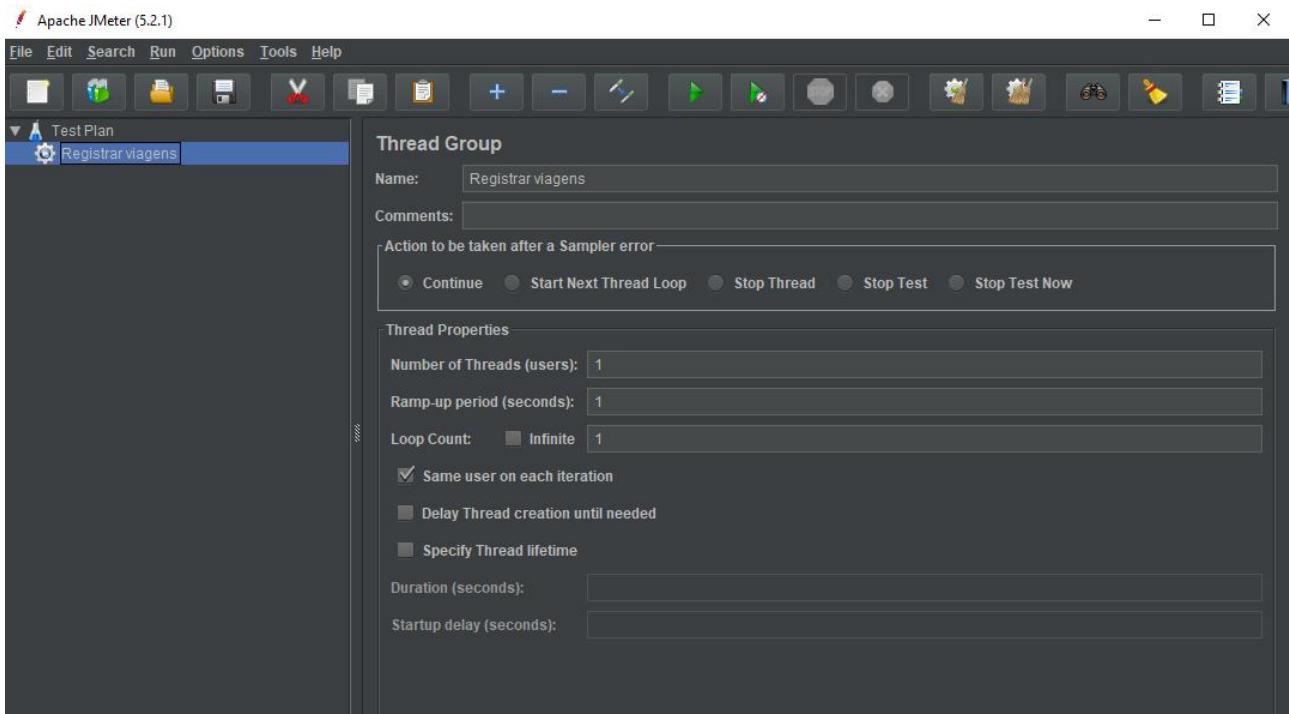
apache-jmeter-5.2.1.zip

Nome	Data de modificaç...	Tipo	Tamanho
bin	01/02/1980 00:00	Pasta de arquivos	
docs	01/02/1980 00:00	Pasta de arquivos	
extras	01/02/1980 00:00	Pasta de arquivos	
lib	01/02/1980 00:00	Pasta de arquivos	
licenses	01/02/1980 00:00	Pasta de arquivos	
printable_docs	01/02/1980 00:00	Pasta de arquivos	
LICENSE	01/02/1980 00:00	Arquivo	15 KB
NOTICE	01/02/1980 00:00	Arquivo	1 KB
README.md	01/02/1980 00:00	Arquivo Fonte Ma...	10 KB

- Dentro da pasta "bin", dê dois cliques sobre o arquivo "jmeter.bat".

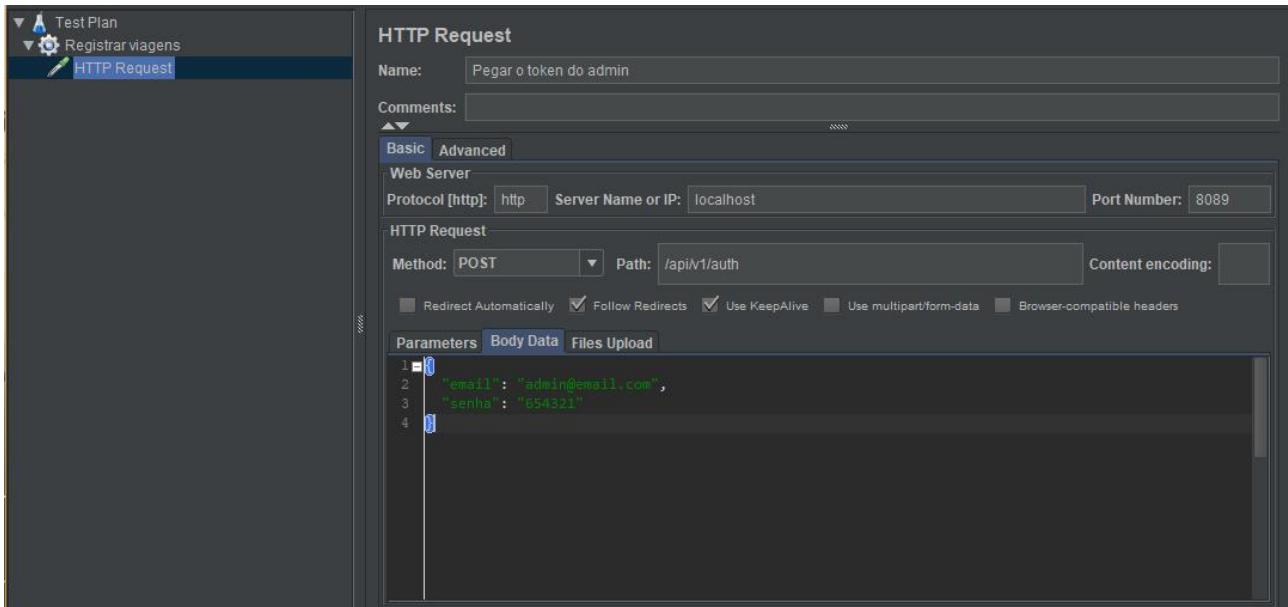


- Clique com o botão direito em **TestPlan** selecione **Add --> Threads (Users) --> Thread Group**.



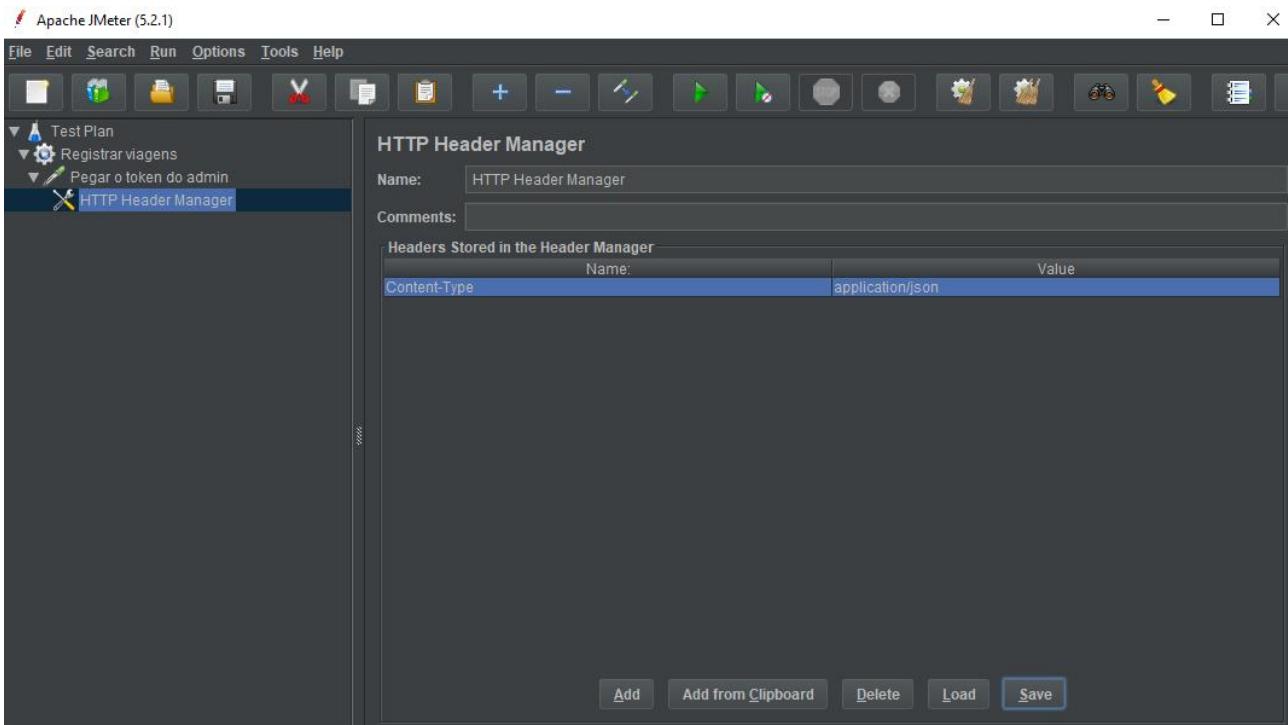
Pegando o token do admin

- Clique com o botão direito do mouse sobre "Registrar viagens" e selecione "Add --> Sampler --> HTTP Request"



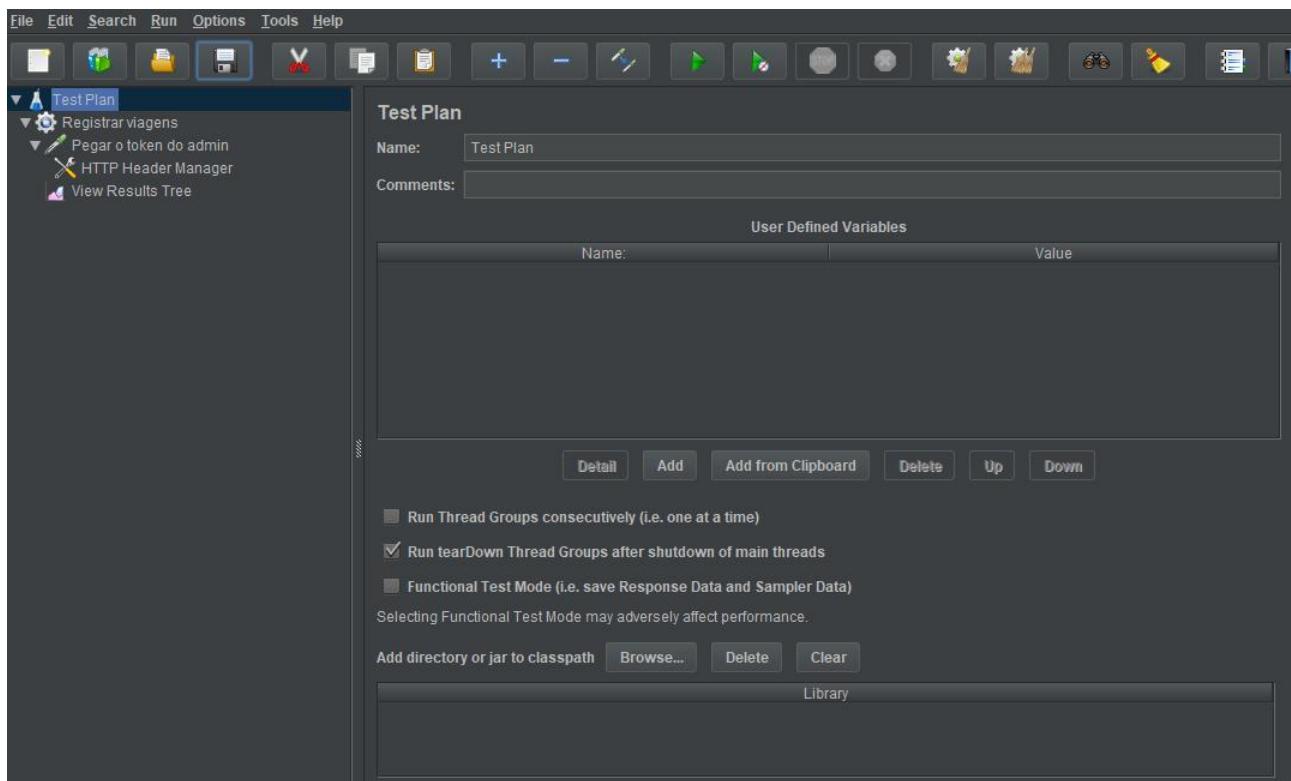
- Clique com o botão direito do mouse sobre "Pegar o token do admin" e selecione "Add --> Config Element --> HTTP Header Manager"

Clique em **Add** e escreva:

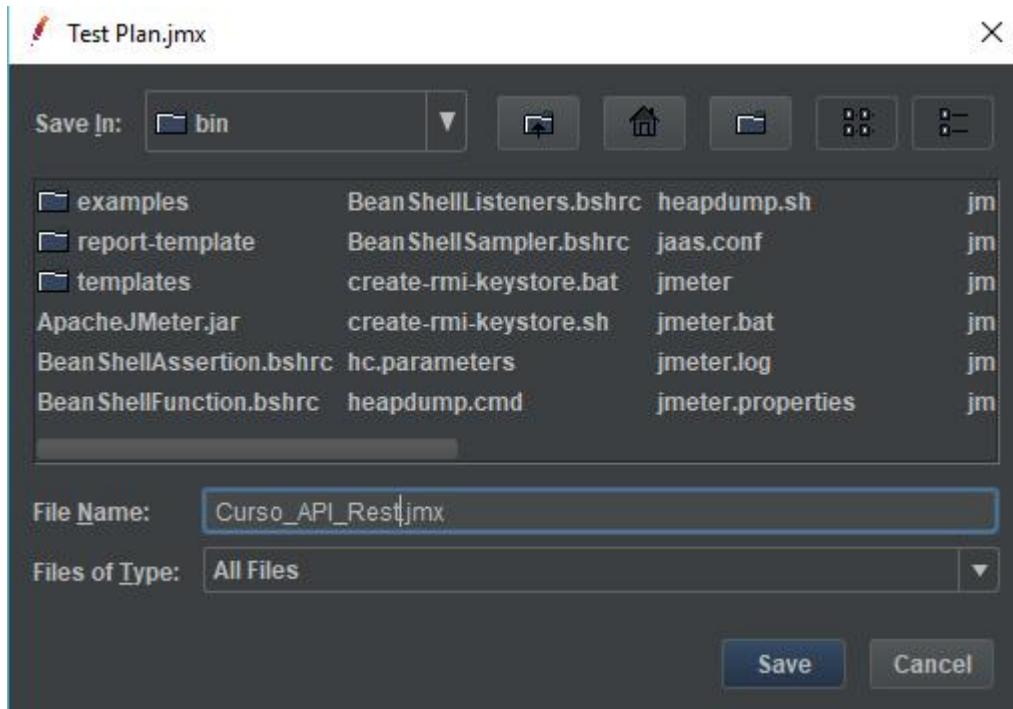


Executando

- Clique com o botão direito do mouse sobre "Registrar viagem" e selecione "Add --> Listener --> View Results Tree"



- Com o botão direito do mouse, clique em **TestPlan** e salve como **Curso_API_Rest.jmx**:



- Clique no botão verde "Start"

- Clique em "View Results Tree":

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: Errors Successes

Search: Case sensitive Regular exp. Search Reset

Text Sampler result Request Response data

Pegar o token do admin

Thread Name: Registrar viagens 1-1
Sample Start: 2022-05-21 14:10:45 BRT
Load time: 22923
Connect Time: 247
Latency: 22816
Size in bytes: 578
Sent bytes: 239
Headers size in bytes: 310
Body size in bytes: 268
Sample Count: 1
Error Count: 0
Data type ("text"|"bin"|""): text
Response code: 200
Response message:

HTTPSampleResult fields:
ContentType: application/json; charset=UTF-8
DataEncoding: UTF-8

Clique na aba **Request**. Será exibido a requisição enviada

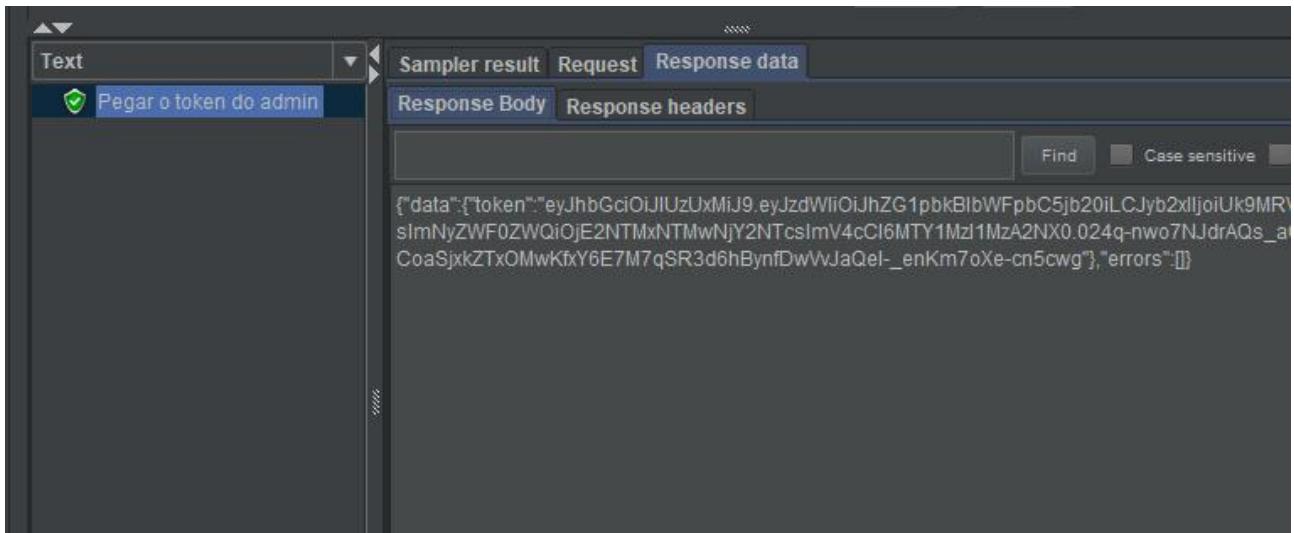
Text Sampler result Request Response data

Pegar o token do admin

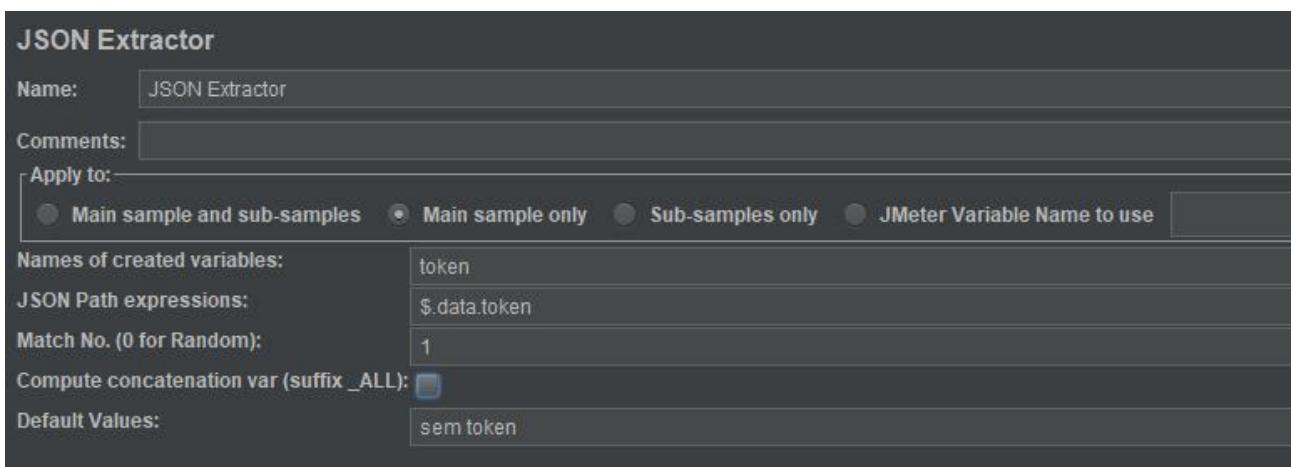
Request Body Request Headers

```
1 POST http://localhost:8089/api/v1/auth
2
3 POST data:
4 [
5   "email": "admin@email.com",
6   "senha": "654321"
7 ]
8
9 [no cookies]
10
```

- Clique na aba **Response data**:

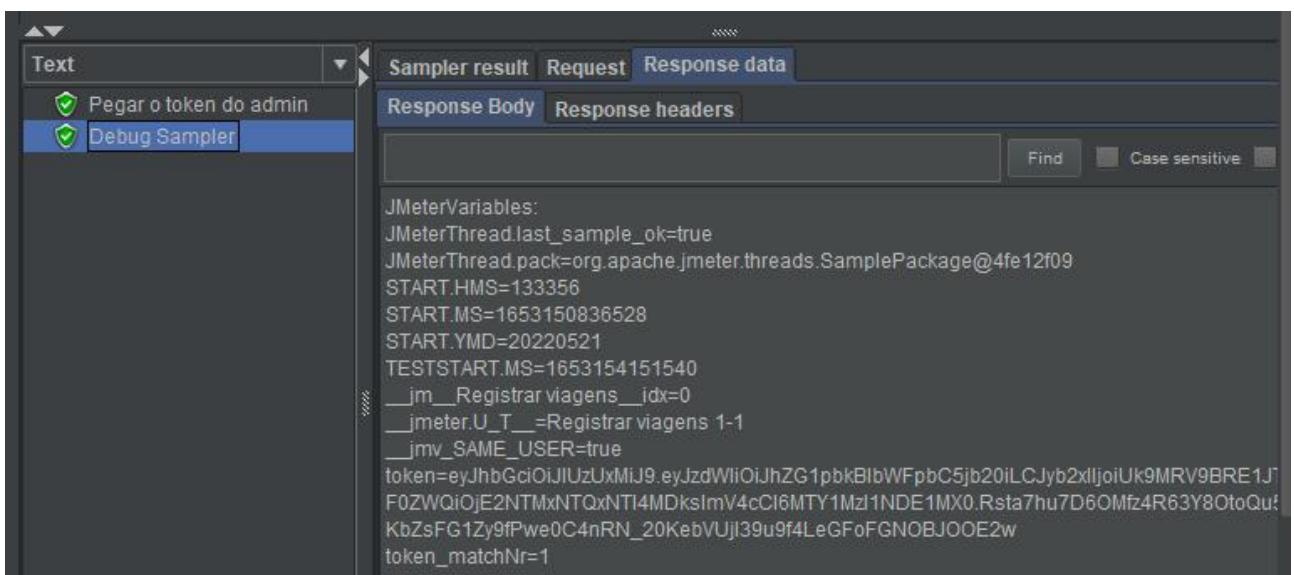


- Com o botão direito do mouse, clique em **Pegar o token do admin** e selecione "**Post Processors --> JSON Extractor**".



- Com o botão direito do mouse, clique em **Pegar o token do admin** e selecione "**Add --> Sampler -> Debug Sampler**"

- Limpe os resultados (**Clear All**) e execute novamente:



- Clique com o botão direito do mouse sobre "Registrar viagens" e selecione "Add --> Sampler --> HTTP Request"

The screenshot shows the "HTTP Request" configuration dialog. The "Name" field is set to "Registrar viagem". Under the "Web Server" tab, the "Protocol [http]: http", "Server Name or IP: localhost", and "Port Number: 8089" fields are filled. In the "HTTP Request" section, the "Method: POST" and "Path: /api/v1/viagens" fields are specified. Below these, several checkboxes are checked: "Follow Redirects", "Use KeepAlive", and "Content encoding:". The "Parameters" tab is selected, showing a JSON payload:

```
1 {"acompanhante": "Isabelle",  
2 "dataPartida": "2021-02-21",  
3 "dataRetorno": "2021-03-21",  
4 "localDeDestino": "Manaus",  
5 "regiao": "Norte"}  
6  
7 }
```

The screenshot shows the "HTTP Header Manager" configuration dialog. The "Name" field is set to "HTTP Header Manager". Under the "Headers Stored in the Header Manager" section, two headers are defined:

	Name:	Value
Content-Type		application/json
Authorization		#{token}

- Salve, limpe os resultados e execute.

The screenshot shows the "Sampler result" tab of the JMeter results viewer. The "Text" column on the left lists samplers: "Pegar o token do admin", "Debug Sampler", and "Registrar viagem". The "Registrar viagem" sampler is currently selected. The "Sampler result" panel displays the following details for the last sample:

Thread Name: Registrar viagens 1-1
Sample Start: 2022-05-21 14:50:37 BRT
Load time: 327
Connect Time: 0
Latency: 324
Size in bytes: 523
Sent bytes: 578
Headers size in bytes: 360
Body size in bytes: 163
Sample Count: 1
Error Count: 0
Data type ("text"|"bin"|""): text
Response code: 201
Response message:

HTTPSampleResult fields:
ContentType: application/json; charset=UTF-8
DataEncoding: UTF-8

Text

Pegar o token do admin
Debug Sampler
Registrar viagem

Sampler result Request Response data

Request Body Request Headers

```
1 POST http://localhost:8089/api/v1/viagens
2
3 POST data:
4 [
5   "acompanhante": "Isabelle",
6   "dataPartida": "2021-02-21",
7   "dataRetorno": "2021-03-21",
8   "localDeDestino": "Manaus",
9   "regiao": "Norte"
10 ]
11
12 [no cookies]
13
```

Find Case sensitive

Text

Pegar o token do admin
Debug Sampler
Registrar viagem

Sampler result Request Response data

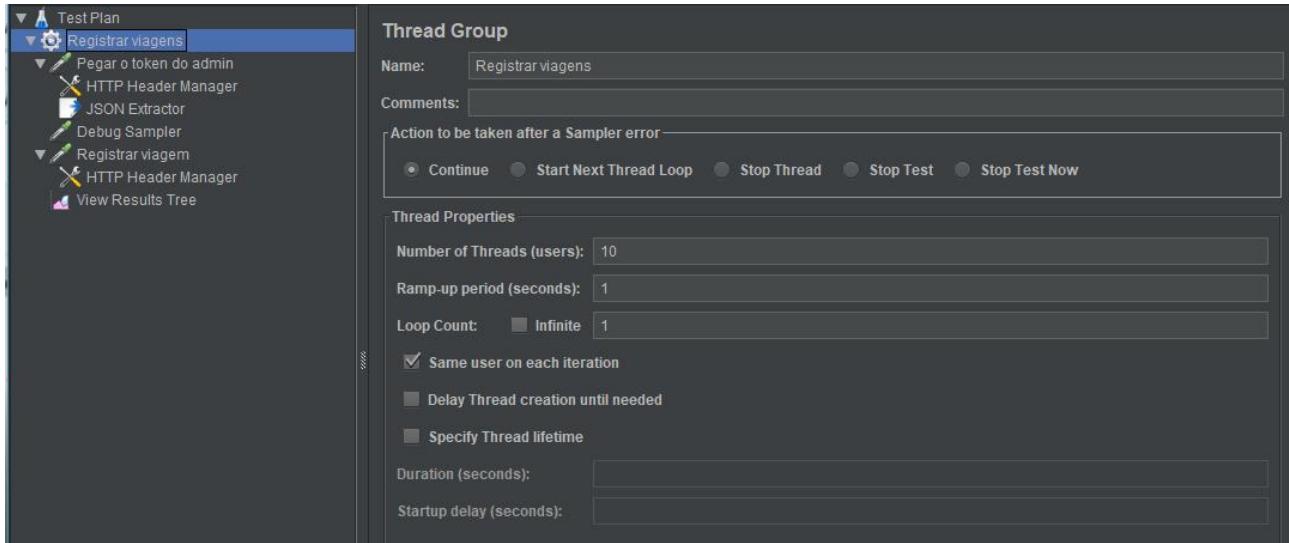
Response Body Response headers

```
{"data": {"id": 6, "localDeDestino": "Manaus", "dataPartida": "2021-02-21", "dataRetorno": "2021-03-21", "acompanhante": "Isabelle", "regiao": "Norte"}, "errors": []}
```

Find Case sensitive

Fazendo o teste com 10 usuários virtuais

- Clique em "Registrar viagens"



- Em **Number of Threads (users)** insira **10**.

- Clique com o botão direito do mouse sobre "**Registrar viagens**" e selecione "**Add --> Listener --> Aggregate Report**"

- Limpe os resultados e execute novamente.

Aggregate Report													
Name:	Aggregate Report												
Comments:													
Write results to file / Read from file													
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Through...	Receiv...	Sent K...	
Pegar o token do admin	10	703	775	872	872	1122	281	1122	0.00%	6.1/sec	3.42	1.41	
Debug Sampler	10	0	0	0	0	0	0	0	0.00%	7.4/sec	3.90	0.00	
Registrar viagem	10	180	128	307	307	386	20	386	0.00%	6.7/sec	3.44	3.79	
TOTAL	30	294	128	827	847	1122	0	1122	0.00%	16.7/sec	8.95	4.45	

Aula 16 - Como fazer testes de compatibilidade em API Rest

Será que o novo funciona em conjunto com o antigo?

A API Rest tem uma nova versão, mas o App continua achando que ela é sua antiga parceira...

Teste de compatibilidade é um teste baseado uma das oito características de qualidade da ISO 25.010

Teste de retro-compatibilidade é um teste para garantir que a versão nova continue funcionando com seus consumidores antigos.

O teste da retro-compatibilidade deve ser feito quando a API ganha uma nova versão.

É verificado se tudo o que existia na API continua funcionando normalmente.

Parte prática

- Entre no github.com e na parte de busca ([Search GitHub](#)) entre com:

[swagger-diff](#)

- Clique no link:

Sayi/swagger-diff

<https://github.com/Sayi/swagger-diff>

The screenshot shows the GitHub repository page for 'Sayi / swagger-diff'. The repository is public and has 84 commits. It features 5 branches and 3 tags. The master branch is selected, showing a commit from June 2020. The repository contains files like src, .gitignore, .travis.yml, LICENSE, README.md, changelog.png, pom.xml, and swagger-diff.png.

File	Description	Time Ago
src	V1.2.2	2 years ago
.gitignore	Add version numbers to output	4 years ago
.travis.yml	Update .travis.yml	4 years ago
LICENSE	Apache License 2.0	6 years ago
README.md	Update README.md	2 years ago
changelog.png	html png	5 years ago
pom.xml	V1.2.2	2 years ago
swagger-diff.png	doc	6 years ago

- Clique na opção 3 tags.

Sayi / swagger-diff Public

<> Code Issues 11 Pull requests 6 Actions Security Insights

Releases Tags

Tags

v1.2.2 ...

🕒 on 29 Jun 2020 -o e331226 📁 zip 📁 tar.gz 📄 Notes ⏪ Downloads

v1.2.1

🕒 on 9 Nov 2018 -o bda37e6 📁 zip 📁 tar.gz 📄 Notes ⏪ Downloads

v1.2.0 ...

🕒 on 16 May 2018 -o 2f8b853 📁 zip 📁 tar.gz 📄 Notes ⏪ Downloads

- Clique em Downloads da versão v1.2.2

swagger-diff v1.2.2

Latest

7 Sayi released this 29 Jun 2020 · 1 commit to master since this release 🏷 v1.2.2 -o e331226

- Merge pull request #34: Change color for HEAD request type
- Merge pull request #26: JSON string Support
- Merge pull request #30 : Fix model array change detect and added produces/consumes change
- Merge pull request #22: Make ParameterDiff detect changes to fix the Type of a PathParameter

▼ Assets 3

📦 swagger-diff.jar

📁 Source code (zip)

📁 Source code (tar.gz)

- Clique e baixe swagger-diff.jar

- Na página deste vídeo:

Swagger antigo:

<https://www.dropbox.com/s/30mijeugpmnzvfw/swagger-antigo.json?dl=0>

Swagger novo:

<https://www.dropbox.com/s/cz9rh3fgrs43765/swagger-novo.json?dl=0>

github > gerenciador-viagens > swagger			
Nome	Data de modificaç...	Tipo	Tamanho
swagger-antigo.json	21/05/2022 18:18	Arquivo Fonte JSON	8 KB
swagger-diff.jar	21/05/2022 18:05	Executable Jar File	10.329 KB
swagger-novo.json	21/05/2022 18:16	Arquivo Fonte JSON	7 KB

- Abra o gitbash e na pasta que contém os três arquivos, entre com o comando:

```
java -jar swagger-diff.jar -old swagger-antigo.json -new swagger-novo.json
```

```
Roberto@DESKTOP-HGDUAAQT MINGW64 /d/github/gerenciador-viagens/swagger (main)
$ java -jar swagger-diff.jar -old swagger-antigo.json -new swagger-novo.json
## Version 1.0 to 1.0
---
### What's New
---

### What's Deprecated
---
* `POST` /v1/viagens Cadastra uma viagem

### What's Changed
---
```

- Para exportar o resultado em HTML:

```
java -jar swagger-diff.jar -old swagger-antigo.json -new swagger-novo.json -output-mode html > diff.html
```

```
Roberto@DESKTOP-HGDUAAQT MINGW64 /d/github/gerenciador-viagens/swagger (main)
$ java -jar swagger-diff.jar -old swagger-antigo.json -new swagger-novo.json -output-mode html > diff.html
```

BIG-HD (D:) > github > gerenciador-viagens > swagger				
	Nome	Data de modificaç...	Tipo	Tamanho
	diff.html	21/05/2022 18:31	Chrome HTML Do...	2 KB
	swagger-antigo.json	21/05/2022 18:18	Arquivo Fonte JSON	8 KB
	swagger-diff.jar	21/05/2022 18:05	Executable Jar File	10.329 KB
	swagger-novo.json	21/05/2022 18:16	Arquivo Fonte JSON	7 KB

Changelog

Versions

Changes from 1.0 to 1.0.

What's New

What's Deprecated

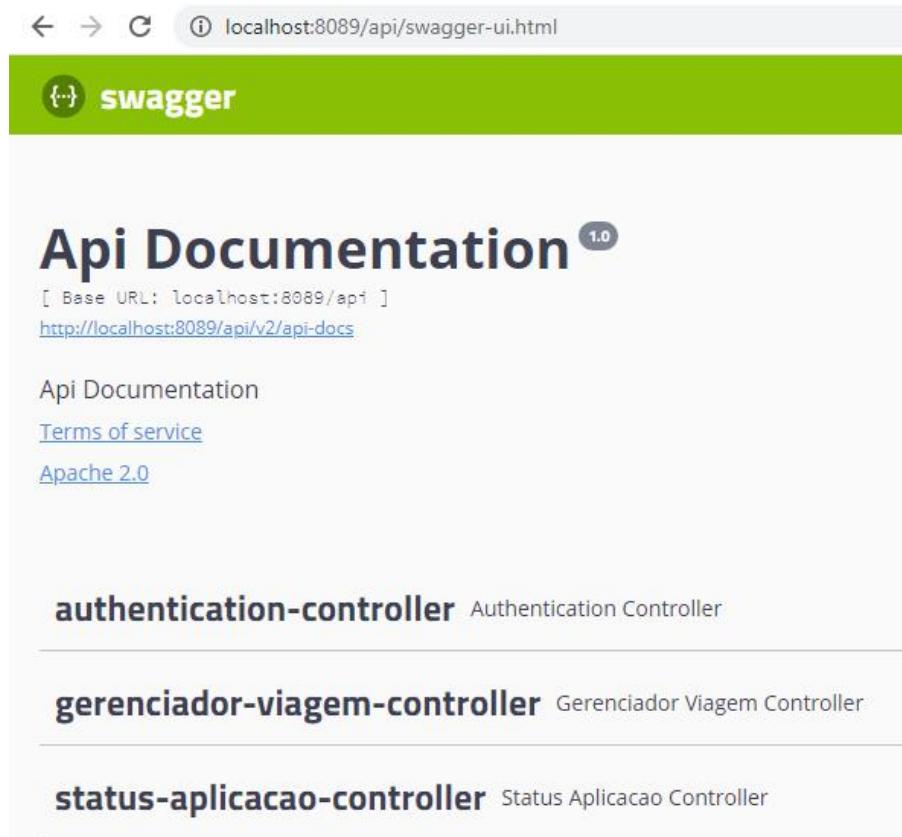
 [/v1/viagens](#) Cadastra uma viagem

What's Changed

Como salvar o arquivo do swagger

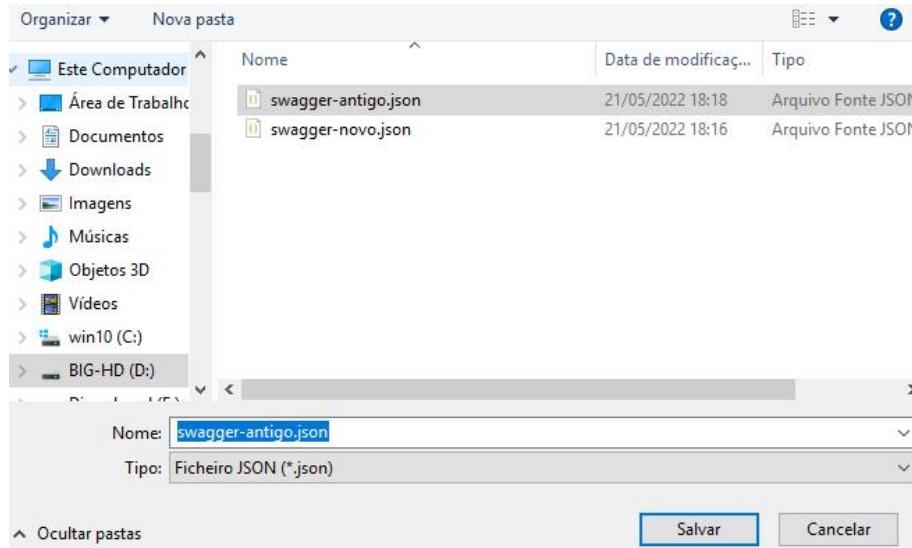
mvn spring-boot:run

localhost:8089/api/swagger-ui.html



The screenshot shows the Swagger UI interface for a Spring Boot application. At the top, there's a navigation bar with back, forward, and refresh buttons, followed by the URL 'localhost:8089/api/swagger-ui.html'. Below this is a green header bar with the 'swagger' logo. The main content area has a title 'Api Documentation' with a '1.0' badge. It includes a note about the base URL: '[Base URL: localhost:8089/api]' and a link to 'http://localhost:8089/api/v2/api-docs'. Below this, there are links for 'Api Documentation', 'Terms of service', and 'Apache 2.0'. The page lists three controller sections: 'authentication-controller' (Authentication Controller), 'gerenciador-viagem-controller' (Gerenciador Viagem Controller), and 'status-aplicacao-controller' (Status Aplicacao Controller).

- Clique com o botão direito do mouse no link (<http://localhost:8089/api/v2/api-docs>) e selecione "Save link as"



Aula 17 - Como criar Mock para testes de API Rest

Criando mocks e reduzindo a dependência de serviços externos

Usando o Wiremock você pode simular serviços HTTP de modo a não depender deles durante uma fase dos seus testes. Isso ajuda muito e acelera o processo de desenvolvimento e testes.

Nessa aula você vai aprender a criar mocks de serviços externos.

Para a API que estamos utilizando para os nossos testes existe um serviço terceiro que é consumido por ela: API do tempo do Júlio de Lima. Às vezes essa API pode não estar disponível com a frequência que desejamos.

ViagemServices

```
public ViagemDtoResponse buscar(Long id) throws IOException, NotFoundException {
    Viagem viagem = viagemRepository.findOne(id);

    if (viagem == null) {
        throw new NotFoundException(format("Viagem com id: [%s] não encontrada", id));
    }

    ViagemDtoResponse viagemDtoResponse = Conversor.converterViagemToViagemDtoResponse(viagem);
    String regiao = viagem.getRegiao();

    if (regiao != null) {
        final String uri = previsaoDoTempoUri + "tempo-api/temperatura?regiao=" + regiao;
        RestTemplate restTemplate = new RestTemplate();

        String previsaoJson = "";

        try {
            previsaoJson = restTemplate.getForObject(uri, String.class);
        } catch (HttpClientErrorException hcee) {
            throw new HttpClientErrorException(HttpStatus.UNPROCESSABLE_ENTITY, "A API do Tempo não está online");
        }

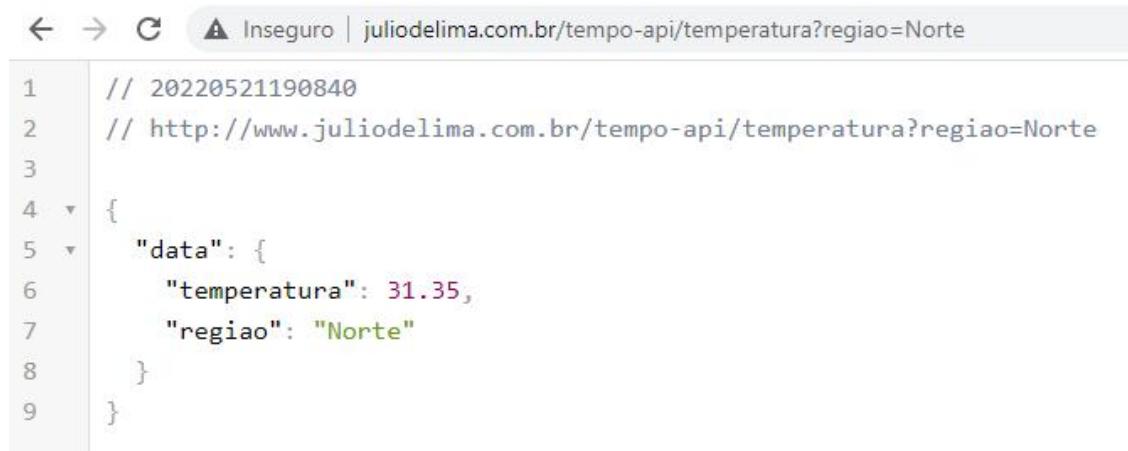
        ObjectNode node = mapper.readValue(previsaoJson, ObjectNode.class);
        viagemDtoResponse.setTemperatura((node.get("data").get("temperatura")).floatValue());

        System.out.println(previsaoDoTempoUri);
    }
}

return viagemDtoResponse;
}
```

Endereço da API do tempo de Julio de Lima

www.juliodelima.com.br/tempo-api/temperatura?regiao=Norte



A screenshot of a browser window showing a JSON response. The URL is <http://www.juliodelima.com.br/tempo-api/temperatura?regiao=Norte>. The response is a single-line JSON object:

```
// 20220521190840
// http://www.juliodelima.com.br/tempo-api/temperatura?regiao=Norte
{
  "data": {
    "temperatura": 31.35,
    "regiao": "Norte"
  }
}
```

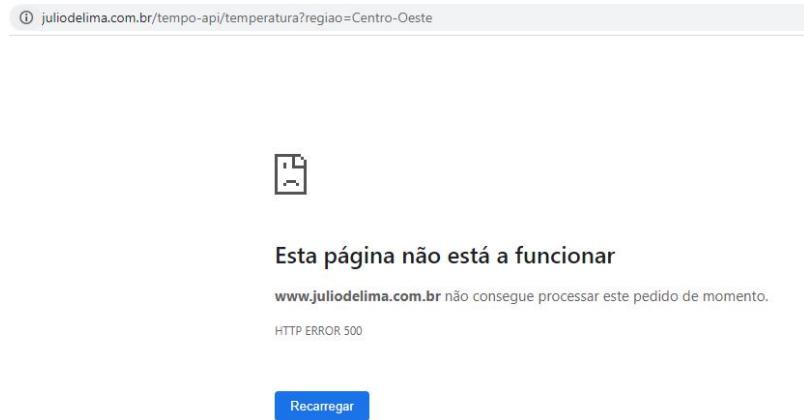
www.juliodelima.com.br/tempo-api/temperatura?regiao=Sul



A screenshot of a browser window showing a JSON response. The URL is <http://www.juliodelima.com.br/tempo-api/temperatura?regiao=Sul>. The response is a single-line JSON object:

```
// 20220521191008
// https://www.juliodelima.com.br/tempo-api/temperatura?regiao=Sul
{
  "data": {
    "temperatura": 27.44,
    "regiao": "Sul"
  }
}
```

<https://www.juliodelima.com.br/tempo-api/temperatura?regiao=Centro-Oeste>



- Se a API não está online ou uma determinada região (Centro-Oeste) não retorna um resultado, pode-se usar um mock que é um simulador para cobrir essa deficiência.

Wiremock

<https://wiremock.org/>

The screenshot shows the official WireMock website at wiremock.org. The page features a large, bold title: "Flexible API mocking for unit, integration and performance testing". Below the title, there's a brief description of what WireMock is: "WireMock is a tool for creating mock APIs. Build predictable development environments, isolate yourself from flakey 3rd party APIs and prototype APIs that don't exist yet." Two main call-to-action boxes are present: "Open Source" (describing the core engine) and "Cloud" (describing MockLab). Both boxes include "Get Started" and "Learn More" buttons.

- Clique em DOCS

The screenshot shows the WireMock documentation homepage at wiremock.org/docs/. The top navigation bar includes links for "Docs", "Support", "Community", and "MockLab". The left sidebar contains a navigation menu with sections like "Setup", "Configuration", "Java Usage", "Stubbing & Verifying", "Record & Playback", "Extensibility", and "Reference". The main content area is titled "Home - WireMock User Documentation" and describes WireMock as an HTTP mock server used for stubbing and verification. It also mentions record/playback features and Java API access via REST and JSON. A note encourages users to post questions on the mailing list if unanswered.

- Clique em **Download and Installation**.

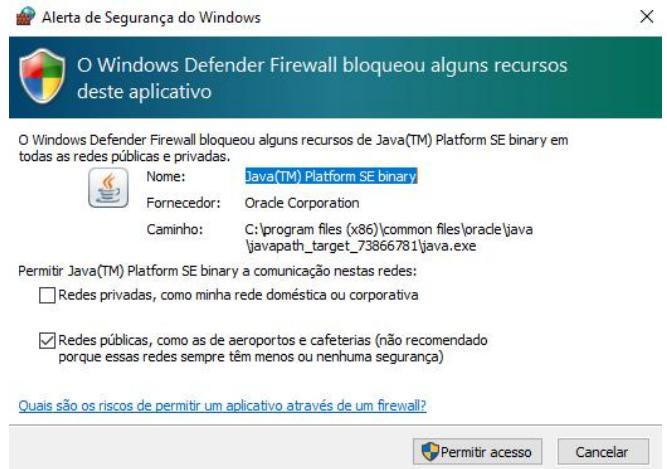
Direct download

If you want to run WireMock as a standalone process you can [download the standalone JAR from here](#)

- Faça o download do arquivo JAR que executa o WireMock.

Rodando o WireMock

```
java -jar wiremock-jre8-standalone-2.33.2.jar
```

A screenshot of a terminal window titled "MINGW64:/d/github/gerenciador-viagens/wiremock". The command \$ java -jar wiremock-jre8-standalone-2.33.2.jar is run, followed by several SLF4J error messages. Then, a large ASCII art logo consisting of dollar signs (\$) is displayed. Finally, configuration options are listed: port: 8080, enable-browser-proxying: false, disable-banner: false, no-request-journal: false, verbose: false.

Clique em [Running as a Standalone Process](#).

<https://wiremock.org/docs/running-standalone/>

Criando um mock

Configuring via JSON over HTTP

You can create a stub mapping by posting to WireMock's HTTP API:

```
$ curl -X POST \
--data '{ "request": { "url": "/get/this", "method": "GET" }, "response": { "status": 200,
http://localhost:8080/_admin/mappings/new'}
```

And then fetch it back:

```
$ curl http://localhost:8080/get/this
Here it is!
```

The full stubbing API syntax is described in [Stubbing](#).

http://localhost:8080/_admin/mappings/new

{ "request": { "url": "/get/this", "method": "GET" }, "response": { "status": 200, "body": "Here it is!\n" }}

- Abra o Insomnia e crie um **new folder** chamado **WireMock**.

- E dentro dessa pasta crie uma nova requisição chamada "**Criar Mock**"

New Request ×

Name (defaults to your request URL if left empty)

Criar Mock POST JSON

* Tip: paste Curl command into URL afterwards to import it Create

Body que se espera retornar:

```
{
  "data": {
    "temperatura": 22.14,
    "regiao": "Centro-Oeste"
  }
}
```

json formatter

json formatter

Aproximadamente 10.100.000 resultados (0,35 segundos)

Dica: Pesquisar apenas resultados em **português (Brasil)**. Especifique seu idioma de pesquisa em Preferências

<https://jsonformatter.curiousconcept.com> Traduzir esta página

JSON Formatter & Validator
The **JSON Formatter & Validator** beautifies and debugs JSON data with advanced formatting and validation algorithms.
Mutate: Data Converter · ZUQ: Alvin McDonald's Initials... · XPath Expression Tester

<https://jsonformatter.org> Traduzir esta página

Best JSON Formatter and JSON Validator: Online JSON ...
Online **JSON Formatter** / Beautifier and JSON Validator will format JSON data, and helps to validate, convert JSON to XML, JSON to CSV. Save and Share JSON.
XML Formatter · JSON Parser · JSON Viewer · JSON Editor

<https://elmah.io/tools/json-form...> Traduzir esta página

<https://jsonformatter.org>

https://jsonformatter.org

JSON BEAUTIFIER JSON PARSE XML FORMATTER JSBEAUTIFIER SAVE RECENT LINKS LOGIN

Upload Data

Validate

2 Tab Space

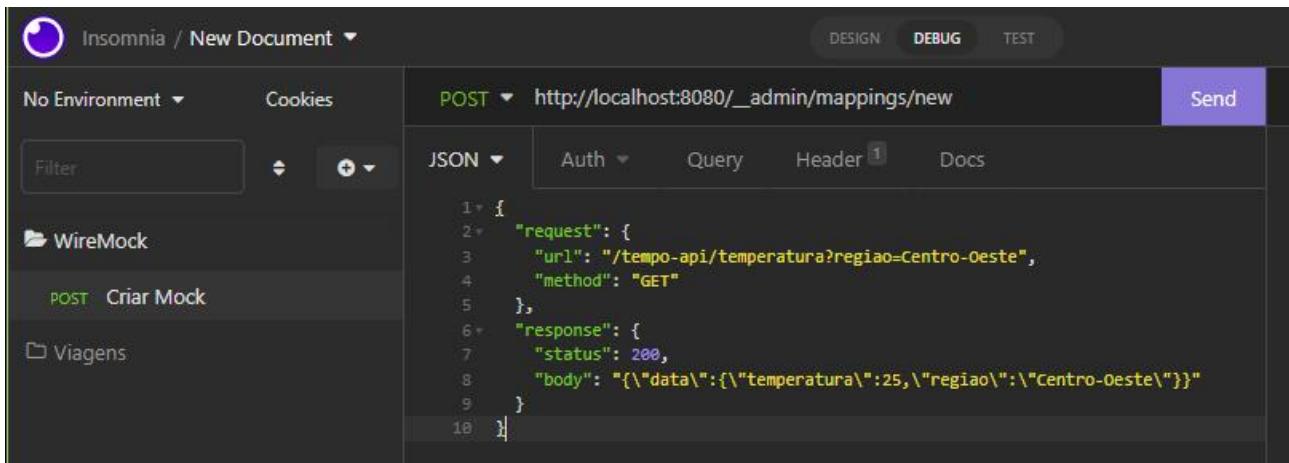
Format / Beautify

Minify / Compact

Veja as experiências únicas que os apps estão criando para celulares dobráveis

- Clique no botão "Minify / Compact" e copie o resultado

No Insomnia:

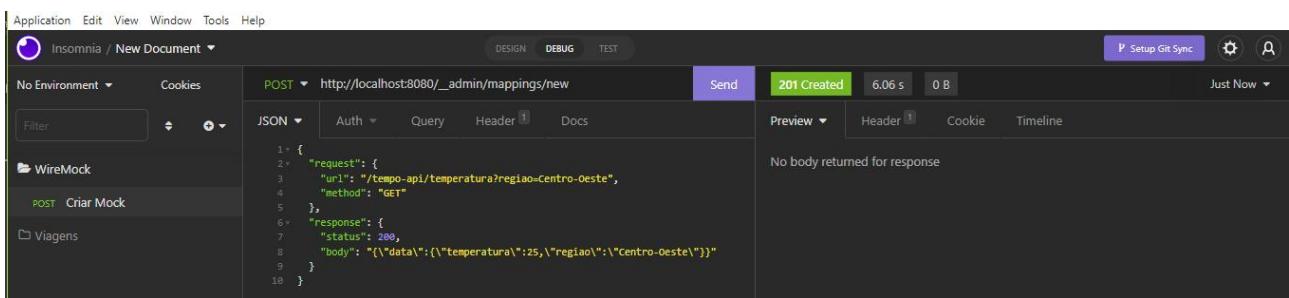


The screenshot shows the Insomnia REST Client interface. On the left, there's a sidebar with 'No Environment' selected and a 'WireMock' section containing a 'Criar Mock' item. The main area shows a POST request to `http://localhost:8080/_admin/mappings/new`. The 'JSON' tab is active, displaying the following code:

```
1+ {
2+   "request": {
3+     "url": "/tempo-api/temperatura?regiao=Centro-Oeste",
4+     "method": "GET"
5+   },
6+   "response": {
7+     "status": 200,
8+     "body": "{\"data\":{\"temperatura\":25,\"regiao\":\"Centro-Oeste\"}}"
9+   }
10 }
```

A purple 'Send' button is at the top right. The status bar at the bottom indicates 'DESIGN'.

- Clique no botão "Send"



The screenshot shows the Insomnia REST Client after sending the POST request. The status bar at the top now shows '201 Created', '6.06 s', and '0 B'. The status bar at the bottom indicates 'DEBUG'. The response pane on the right shows 'No body returned for response'.

Status: 201 Created

<http://localhost:8080/tempo-api/temperatura?regiao=Centro-Oeste>



The screenshot shows a browser window with the URL `localhost:8080/tempo-api/temperatura?regiao=Centro-Oeste`. The page content displays the JSON response:

```
1 // 20220522085846
2 // http://localhost:8080/tempo-api/temperatura?regiao=Centro-Oeste
3
4 {
5   "data": {
6     "temperatura": 25,
7     "regiao": "Centro-Oeste"
8   }
9 }
```

Informando o uso do Mock na API Rest

- Dentro da API Rest faça a seguinte alteração no arquivo abaixo:

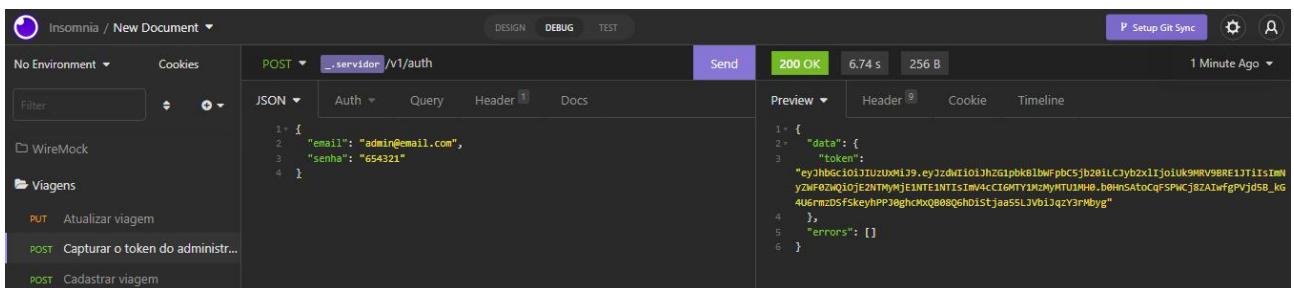
src\resources\application.properties

```
server.port=8089
spring.jackson.serialization.WRITE_DATES_AS_TIMESTAMPS = false
jwt.secret=istoehapenasumteste
jwt.expiration=99999
previsaoDoTempoUri=http://localhost:8080/
server.servlet-path=/api
```

mvn spring-boot:run

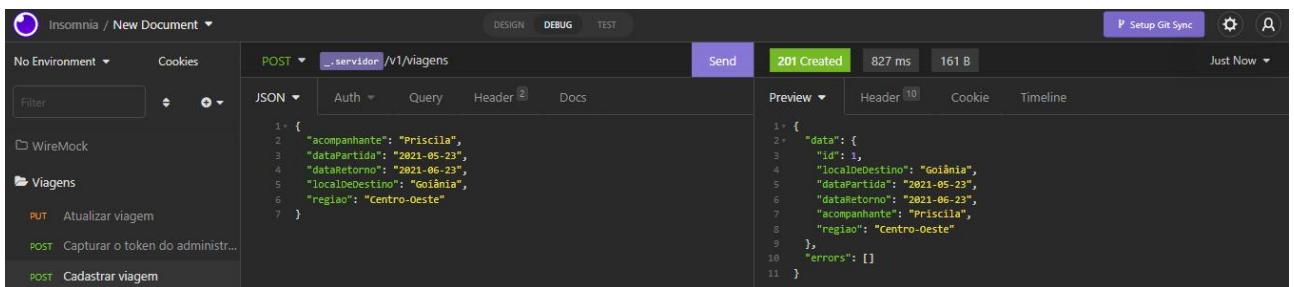
Capturando o token do administrador

No Insomnia, capture o token do administrador:



The screenshot shows the Insomnia REST Client interface. A POST request is made to `...servidor /v1/auth`. The request body is a JSON object with fields `"email": "admin@email.com"` and `"senha": "654321"`. The response status is **200 OK**, with a response time of 6.74 s and a response size of 256 B. The response body contains a large JWT token.

Cadastrando uma viagem para a região Centro-Oeste



The screenshot shows the Insomnia REST Client interface. A POST request is made to `...servidor /v1/viagens`. The request body is a JSON object with fields `"acompanhante": "Priscila"`, `"dataPartida": "2021-05-23"`, `"dataRetorno": "2021-06-23"`, `"localDeDestino": "Goiânia"`, and `"regiao": "Centro-Oeste"`. The response status is **201 Created**, with a response time of 827 ms and a response size of 161 B. The response body indicates a new resource was created with ID 1.

Capturando o token do usuário

No Insomnia, capture o token do usuário:

The screenshot shows the Insomnia REST Client interface. The top bar indicates a successful response (200 OK) with 335 ms duration and 261 B size. The main area shows a JSON response body:

```
1+ {
2+   "data": {
3+     "token": "eyJhbGciOiJIUzI1NiJ9eyJzdWIiOiJ1c3Vhcm1vQGVtYWIslmNvbSisInJVbGUjOjISt0xFX1VTUFSSU8iLCJmcWhdckVkJjoxNjUzMjIyMTQ3MzAwLClhAiOjeZNTMzMjixNDZ9-YiaLEWT_R1miQUNDakgBZQmijMn3hrrowViRoe-CTe6ILTaMjkWkW2Cb0Z_3ZuB5zcBUM9oXi6Zd8zXqVjw"
4+   },
5+   "errors": []
6+ }
```

Buscar dados de uma viagem específica

The screenshot shows the Insomnia REST Client interface. The top bar indicates a successful response (200 OK) with 4.41 s duration and 180 B size. The main area shows a JSON response body:

```
1+ {
2+   "data": {
3+     "id": 1,
4+     "localDeDestino": "Goiânia",
5+     "dataPartida": "2021-05-23",
6+     "dataRetorno": "2021-06-23",
7+     "acompanhante": "Priscila",
8+     "regiao": "Centro-Oeste",
9+     "temperatura": 25.0
10+   },
11+   "errors": []
12+ }
```

Aula 18 - Como validar a estrutura do corpo da requisição com JsonSchema

Validando a estrutura do corpo da requisição

Já pensou que há muita coisa a ser validado no corpo da resposta? Pois é, pensando nisso eu te apresento o JSON Schema, uma forma muito mais simples de validar a estrutura do corpo da resposta.

- Dentro da API Rest faça a seguinte alteração no arquivo abaixo:

src\resources\application.properties

```
server.port=8089
spring.jackson.serialization.WRITE_DATES_AS_TIMESTAMPS = false
jwt.secret=istoehapenasumteste
jwt.expiration=99999
previsaoDoTempoUri=http://juliodelima.com.br/
server.servlet-path=/api
```

mvn spring-boot:run

- Abra o Postman

Fazer login como administrador

The screenshot shows the Postman interface with the following details:

- Collection:** Gerenciador de Viagens
- Request:** POST Fazer login como administrador
- URL:** {{servidor}}/v1/auth
- Method:** POST
- Body:** (selected)
JSON:
```

```
1 ... "email": "admin@email.com",
2 ... "senha": "654321"
```

```
- Headers:** (8 items)
- Tests:** (selected)
- Responses:** (selected)
Pretty
Raw
Preview
Visualize
JSON
```

```
1 ...
2 "data": {
3 "token": "eyJhbGciOiJIUzUxMiJ9.
4 eyJzdWIiOiIhZG1pbkBlbwFpbG5jb28iLCJyb2xliIjoiUk9MRV9BRE13TiIsImNyZWFOZwQiojE2NTMyMjM2MDc2OTEsiMv4cCI6MTY
5 1MzMzMzYwNn0.6Fr2hJ70uneid42Sg3ZrqXnFYy-G8hCs2IZ54KkojtSoWShq2YDkIgqtzww_I9hbIxj21n6MX-uqw_JKN716Zg"
6 },
7 "errors": []
```

```

The screenshot shows the Postman interface with the 'Scratch Pad' tab selected. In the left sidebar, 'Collections' is expanded, showing 'Globals'. The main area displays a table of global variables:

VARIABLE	TYPE	INITIAL VALUE	CURRENT VALUE	Persist All	Reset All
servidor	default	http://localhost:8089/api	http://localhost:8089/api		
token	default		eyJhbGciOiJIUzUxMiJ9.eyJzdWlrcHJZG1pbkBlbWFpbC5j...		

Cadastrar viagem

The screenshot shows the Postman interface with a workspace named 'Gerenciador de Viagens'. A POST request is selected with the URL {{(servidor)}}/v1/viagens. The 'Body' tab is active, showing the following JSON payload:

```

1  "data": {
2     "id": 2,
3     "localDeDestino": "Manaus",
4     "dataPartida": "2021-02-06",
5     "dataRetorno": "2021-03-06",
6     "acompanhante": "Isabelle",
7     "regiao": "Norte"
8   },
9   "errors": []
10
11

```

The response status is 201 Created, with a time of 99 ms and a size of 519 B.

No Google, procure por:

`jsonschema`

The screenshot shows a Google search results page for 'jsonschema'. The search bar contains 'jsonschema'. Below the search bar, there are filters: 'Todas', 'Videos', 'Imagens', 'Notícias', 'Shopping', 'Mais', and 'Ferramentas'. The search results section shows the following information:

- Aproximadamente 4.010.000 resultados (0,34 segundos)
- <https://json-schema.org> Traduzir esta página
- JSON Schema | The home of JSON Schema**
JSON Schema is a vocabulary that allows you to annotate and validate JSON documents.
Benefits. Describes your existing data format(s). Provides clear human- and ...
[Getting Started Step-By-Step](#) · [Understanding JSON](#) · [Specification](#) · [Blog](#)
- <https://www.jsonschema.net> Traduzir esta página
- JSON Schema Tool**
jsn is a modern CLI for generating JSON Schema from JSON. Available for Mac OS, Windows, and Linux. ... The same API that powers jsonschema.net is available to ...

<https://www.jsonschema.net/>

<https://www.jsonschema.net/login>

The screenshot shows the login interface of jsonschema.net. It features a top navigation bar with links for 'New', 'Em pausa', 'Github', and a 'Try it now' button. Below the header is a form with two input fields: 'Email *' and 'Password *'. A large blue 'Login' button is centered below the password field. To the right of the login form, there are links for 'Forgot your password? [Recover it](#)' and 'Don't have an account? [Sign up](#)'. Below these, the word 'or' is followed by three social login buttons: 'Log in with Google' (with a Google logo), 'Log in with Github' (with a Github logo), and 'Log in with Facebook' (with a Facebook logo). At the bottom, a link reads 'You can also [continue as a Guest](#)'.

Cadastre-se, clicando em "[Sign up](#)"

The screenshot shows the jsonschema.net schema editor interface. On the left, the 'JSON' tab is active, displaying a 'Valid JSON' area with a code editor containing the following JSON:

```
1 v {  
2 v     "checked": false,  
3 v     "dimensions": {  
4 v         "width": 5,  
5 v         "height": 10  
6 v     },  
7 v     "id": 1,  
8 v     "name": "A green door",  
9 v     "price": 12.5,  
10 v    "tags": [  
11 v        "home",  
12 v        "green"  
13 v    ]  
14 v }
```

Below the code editor are buttons for 'Reset' and 'Submit'. The character count is shown as '205/20,000 characters'. On the right, the 'Schema' tab is active, displaying a code editor with the following JSON schema:

```
1 v {  
2 v     "$schema": "https://json-schema.org/draft/2019-09/schema",  
3 v     "$id": "http://example.com/example.json",  
4 v     "type": "object",  
5 v     "default": {},  
6 v     "title": "Root Schema",  
7 v     "required": [  
8 v         "checked",  
9 v         "dimensions",  
10 v        "id",  
11 v        "name",  
12 v        "price",  
13 v        "tags"  
14 v    ],  
15 v    "properties": {  
16 v        "checked": {  
17 v            "type": "boolean",  
18 v            "default": false,  
19 v        }  
20 v    }  
21 v }
```

At the bottom of the schema editor, the text 'Created: May 22, 2022 10:03 AM Updated: May 22, 2022 10:03 AM' is displayed. The interface includes tabs for 'Syntax' and 'Edit' at the top right.

- Copie e cole o resultado do cadastro de viagem obtido no Postman, dentro do campo **JSON** e clique no botão **Submit**.

The screenshot shows a JSON editor interface. On the left, under 'JSON', there is a code editor containing valid JSON data for a trip. On the right, under 'Schema', there is a code editor containing a generated JSON Schema. Below the editors, there are buttons for 'Reset' and 'Submit'. At the bottom, it shows character count (230/20,000), creation date (May 22, 2022 10:10 AM), and update date (May 22, 2022 10:10 AM). A 'Try it now' button and a GitHub link are also visible at the top.

```
1 v
2 v
3 v
4 v
5 v
6 v
7 v
8 v
9 v
10 v
11 v
12 v
13 v
14 v
15 v
16 v
17 v
18 v
```

```
1 v
2 v
3 v
4 v
5 v
6 v
7 v
8 v
9 v
10 v
11 v
12 v
13 v
14 v
15 v
16 v
17 v
18 v
```

- Copie o resultado apresentado no campo **Schema**.

- No Postman clique na aba **Tests** e cole esse resultado dentro de uma variável chamada **schema**.

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});

const schema = {
    "$id": "http://example.com/example.json",
    "type": "object",
    "default": {},
    "title": "Root Schema",
    "required": [
        "data",
        "errors"
    ],
    "properties": {
        "data": {
            "type": "object",
            "default": {},
            "title": "The data Schema",
            "required": [
                "id",
                "localDeDestino",
                "dataPartida",
                "dataRetorno",
                "acompanhante",
                "regiao"
            ],
            "properties": {
                "id": {
                    "type": "integer",
                    "default": 0,
                    "title": "The id Schema",

```

```
        "examples": [
          10
        ],
      },
      "localDeDestino": {
        "type": "string",
        "default": "",
        "title": "The localDeDestino Schema",
        "examples": [
          "Manaus"
        ]
      },
      "dataPartida": {
        "type": "string",
        "default": "",
        "title": "The dataPartida Schema",
        "examples": [
          "2021-02-06"
        ]
      },
      "dataRetorno": {
        "type": "string",
        "default": "",
        "title": "The dataRetorno Schema",
        "examples": [
          "2021-03-06"
        ]
      },
      "acompanhante": {
        "type": "string",
        "default": "",
        "title": "The acompanhante Schema",
        "examples": [
          "Isabelle"
        ]
      },
      "regiao": {
        "type": "string",
        "default": "",
        "title": "The regiao Schema",
        "examples": [
          "Norte"
        ]
      }
    },
    "examples": [
      {
        "id": 10,
        "localDeDestino": "Manaus",
        "dataPartida": "2021-02-06",
        "dataRetorno": "2021-03-06",
        "acompanhante": "Isabelle",
        "regiao": "Norte"
      }
    ],
    "errors": {
      "type": "array",
      "default": [],
      "title": "The errors Schema",
      "items": {}
    }
  }
```

```
"examples": [
    []
]
},
"examples": [
    "data": {
        "id": 10,
        "localDeDestino": "Manaus",
        "dataPartida": "2021-02-06",
        "dataRetorno": "2021-03-06",
        "acompanhante": "Isabelle",
        "regiao": "Norte"
    },
    "errors": []
]
}
```

- A seguir insira um novo teste, abaixo:

```
pm.test("Validar a estrutura do corpo da resposta", function () {
    pm.response.to.have.jsonSchema(schema);
});
```

- Clique no botão "Send"

- Clique na aba "Tests Results"

The screenshot shows the 'Test Results' tab in Postman, which is currently selected. It displays two test results: one for 'Status code is 201' and another for 'Validar a estrutura do corpo da resposta'. Both tests are marked as 'PASS'.

Test	Status	Description
Status code is 201	PASS	
Validar a estrutura do corpo da resposta	PASS	

Aula 19 - Como ler logs de APIs Rest

Dicas de como ler melhor a identificação de causa raiz com o uso de logs

Nessa aula você vai aprender como ler logs é uma atividade importante, inclusive para aquelas pessoas que apenas testam aplicações.

Logs são registros de eventos ou acontecimentos dentro da plataforma que você está testando.

Leitura de Logs

Há vários níveis de log, dentre eles estão:

DEBUG: Usado para depurar, tempo de desenvolvimento

TRACE: Usado para depurar, tempo de produção

INFO: Informações importantes sobre o progresso do uso

WARN: Situação potencialmente perigosa

ERROR: Erros que não impedem o uso da aplicação

FATAL: Erros que impedem o uso da aplicação

O DEBUG é usado para ambiente local e o TRACE é utilizado em produção.

Leitura de Logs

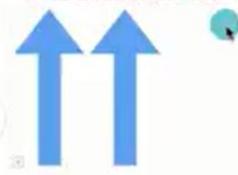
```
1  @RestController
2  public class LoggingController {
3
4      Logger logger = LoggerFactory.getLogger(LoggingController.class);
5
6      @RequestMapping("/")
7      public String index() {
8          logger.trace("A TRACE Message");
9          logger.debug("A DEBUG Message");
10         logger.info("An INFO Message");
11         logger.warn("A WARN Message");
12         logger.error("An ERROR Message");
13
14         return "Howdy! Check out the Logs to see the output...";
15     }
16 }
```



Quando um erro acontece com a aplicação em execução

Leitura de Logs

```
06-18 15:24:30.271 3817-3831/com.codepath.flixster E/Surface: getSlotFromBufferLocked: unknown buffer: 0x7f4b73f99ec0  
06-18 15:24:32.928 3817-3817/com.codepath.flixster E/AndroidRuntime: FATAL EXCEPTION: main
```



```
    Process: com.codepath.flixster, PID: 3817  
    java.lang.RuntimeException: Unable to instantiate activity ComponentInfo{com.codepath.flixster/.MainActivity}  
        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:315)  
        at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:304)  
        at android.app.ActivityThread.handleMessage(ActivityThread.java:172)  
        at android.os.Handler.dispatchMessage(Handler.java:102)  
        at android.os.Looper.loop(Looper.java:148)  
        at android.app.ActivityThread.main(ActivityThread.java:5417) <1 internal frames>  
        at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:941)  
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:860)
```



Aula 20 - Dicas para entrevistas

Preparando-se para entrevistas e nosso encerramento

Essa é a hora de conhecer mais sobre como arrebentar nas entrevistas que exigem conhecimento em Testes de API Rest.

Dicas para entrevistas

- 1) Lembre-se, testar APIs não é apenas sobre enviar requisições e receber respostas, logo, use mindmaps para estratégia de testes.
- 2) Prepare-se para explicar os principais conceitos relacionados a interface de APIs Rest.
- 3) Encontre espaço para demonstrar o conhecimento que você aprendeu sobre as camadas que estão por detrás da API Rest.
- 4) Ao ser indagado sobre o que testar em uma API Rest, explique com detalhes as técnicas e estratégias que aprendemos aqui no curso.
- 5) Deixe claro ao entrevistador as formas de automatizar testes que você aprendeu aqui. Capriche na criação dos scripts antes de enviá-los ao entrevistador.

Considerações e Perguntas

Testar APIs não é apenas sobre enviar requisições e receber respostas
Não se limite a apenas enviar requisições e receber respostas.

1) Temos aqui esse swagger para uma determinada API Rest. O que você testaria aqui?

Há formas distintas de testar. Você pode testar com base em regras de negócio, aplicar técnicas, testar de maneira não funcional, testar performance, testar a compatibilidade, analisar logs, aplicar testes para validar a estrutura da resposta (através do Json Schema)

Quando você está testando uma aplicação de interface gráfica (um site ou um aplicativo em um celular) você já tem um senso natural de como as coisas devem funcionar. Você visualmente consegue identificar o que é um botão, o que é um campo, o que é uma barra de rolagem, porque você já conhece a interface da aplicação.

No caso de uma API (Interface de Programação de Aplicação), a interface tem outras características, outras formas de operação.

Existem:

- Cabeçalho (Header)
 - Corpo (Body)
 - Parâmetros
 - Métodos
 - Autenticação
-

2) Quais são formas de envio de parâmetros para API Rest?

- Envio de parâmetro **via Query** que é enviado na URI após o símbolo de interrogação
- Envio de parâmetros **via Path** por exemplo, atributo id (api/v1/viagens/1, api/v1/viagens/2, etc.).
- Envio de parâmetros **via Header**, por exemplo, o Content Type e Authentication (com o token)
- Envio de parâmetros **via Body**, por exemplo, envio de email e senha para se autenticar

No Swagger está descrito a interface da API Rest

3) Quais são as camadas que estão por detrás da API Rest?

Por detrás de uma API Rest há:

- **Controller**: recebe as informações do mundo exterior (por exemplo, através do cURL, Postman, Insomnia, Rest Assured, ou através de outras informações que estão consumindo aquela API).
- **Service**: é uma camada da aplicação que geralmente armazena as regras de negócios
- **Repository**: camada da aplicação responsável por trafegar as informações entre a regra de negócio e o banco de dados. É a interface com o banco de dados.

A requisição é enviada. O Postman chama o controller. O controller determina o que vem na requisição e qual o método que ele deve chamar. E aí chama os services. O Service chama o repository, o repository interage com o banco de dados que devolve a informação para o repository, que devolve a informação para o service, que devolve a informação para o controller. E o controller devolve a informação para a interface gráfica que renderiza esses dados.

4) O que testar em uma API Rest?

Os testes em uma API Rest podem ser feitos de duas formas:

- 1) Pensando de maneira aleatória no que deveria ser testado
- 2) Demonstrando suas habilidades e explicando com detalhes as técnicas e estratégias (com o MindMap).

Para cada teste que for fazer em uma API Rest demonstre o fundamento utilizado na escolha do teste.

Compatibilidade é uma das oito características de qualidade da ISO 25010.

Testar Retro Compatibilidade através do software Swagger Diff pode revelar algumas inconsistências.

5) Quais são as formas de automatizar testes em APIs Rest

- Princípios da automação de testes com Postman (scripts na aba Tests)
 - Automatização de testes com o Rest Assured
 - Automatização de testes de performance com o JMeter
 - Automatização de testes de compatibilidade com o Swagger Diff
-

Ferramentas e Testes

1. **Rest Assured** (Automatização de Testes de APIs Rests)
2. **JMeter** (Testes de Performance)
3. **Swagger Diff** (Testes de Compatibilidade)
4. **WireMock** é uma ferramenta que possibilita a simulação de serviços de API baseadas em HTTP. É útil quando temos a necessidade de realizar testes e nem todos os serviços estão integrados ou disponíveis.
5. **JSON Schema** permite validar a estrutura do corpo da resposta.

Cobertura de Testes

Existe uma forma de calcular a cobertura de testes, que se baseia nos critérios de cobertura de entrada (Input Coverage) e cobertura de saída (Output Coverage).

Estratégias de testes

MindMap (Mapa Mental)