// Testes

Rest-Assured


REST-assured


CONTE-ME MAIS
SOBRE ISSO

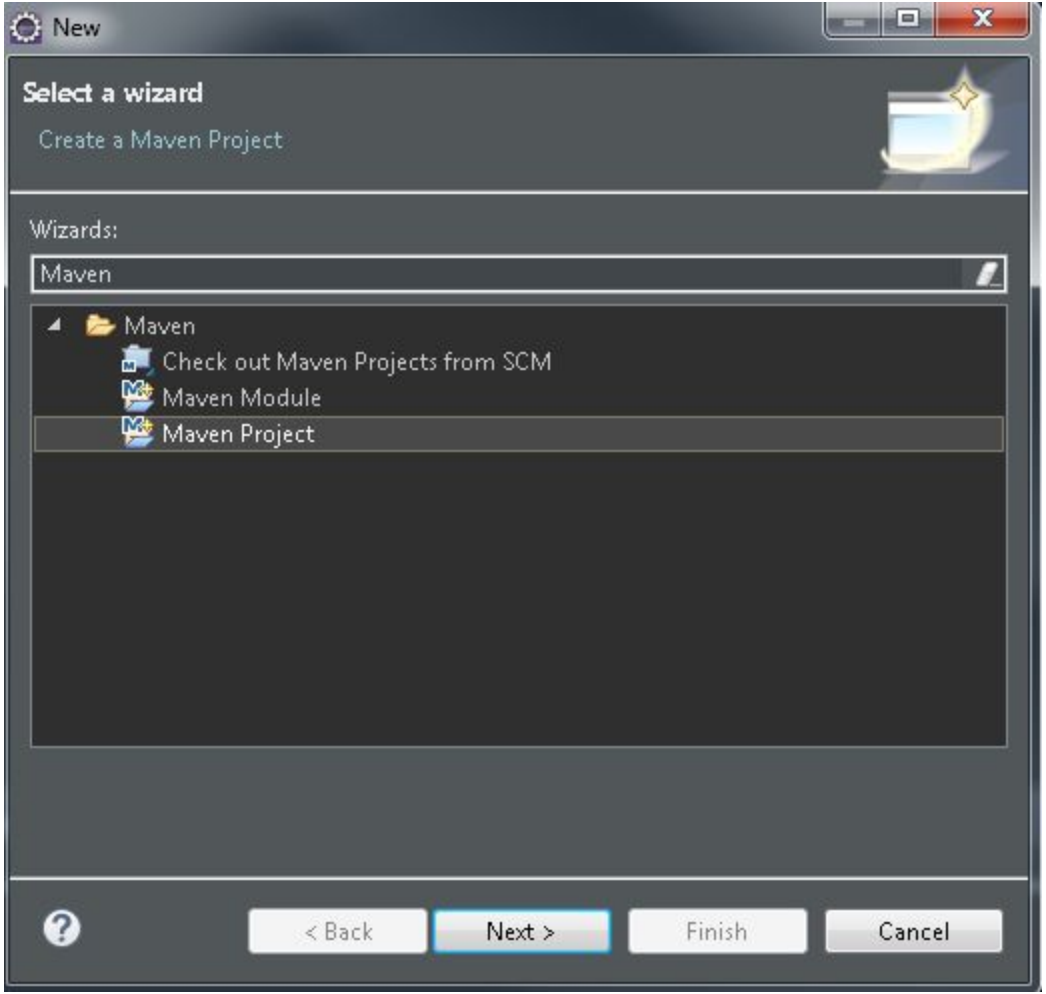
O que é rest-assured?

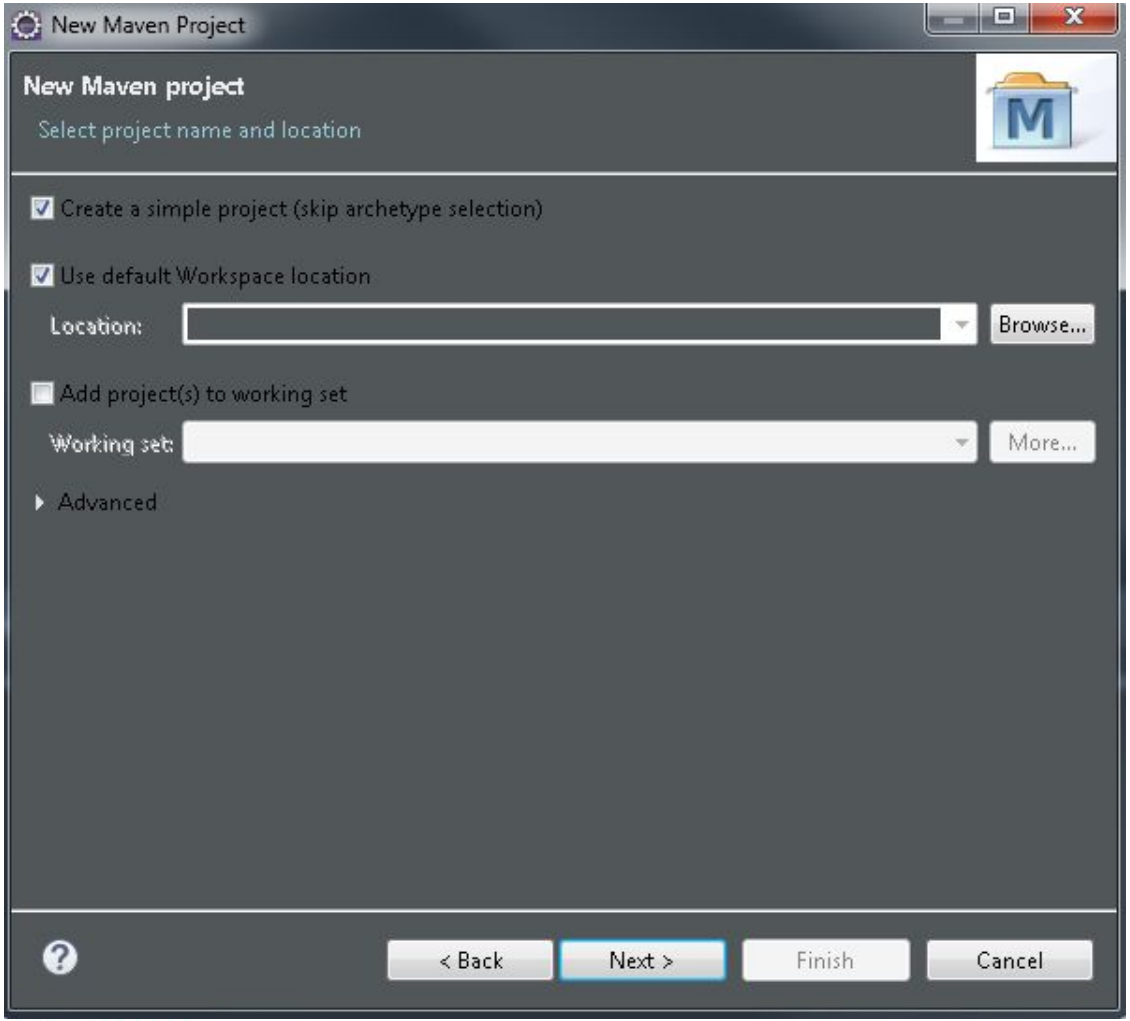
Rest-Assured é um framework open-source que possui suporte para automatizar os métodos

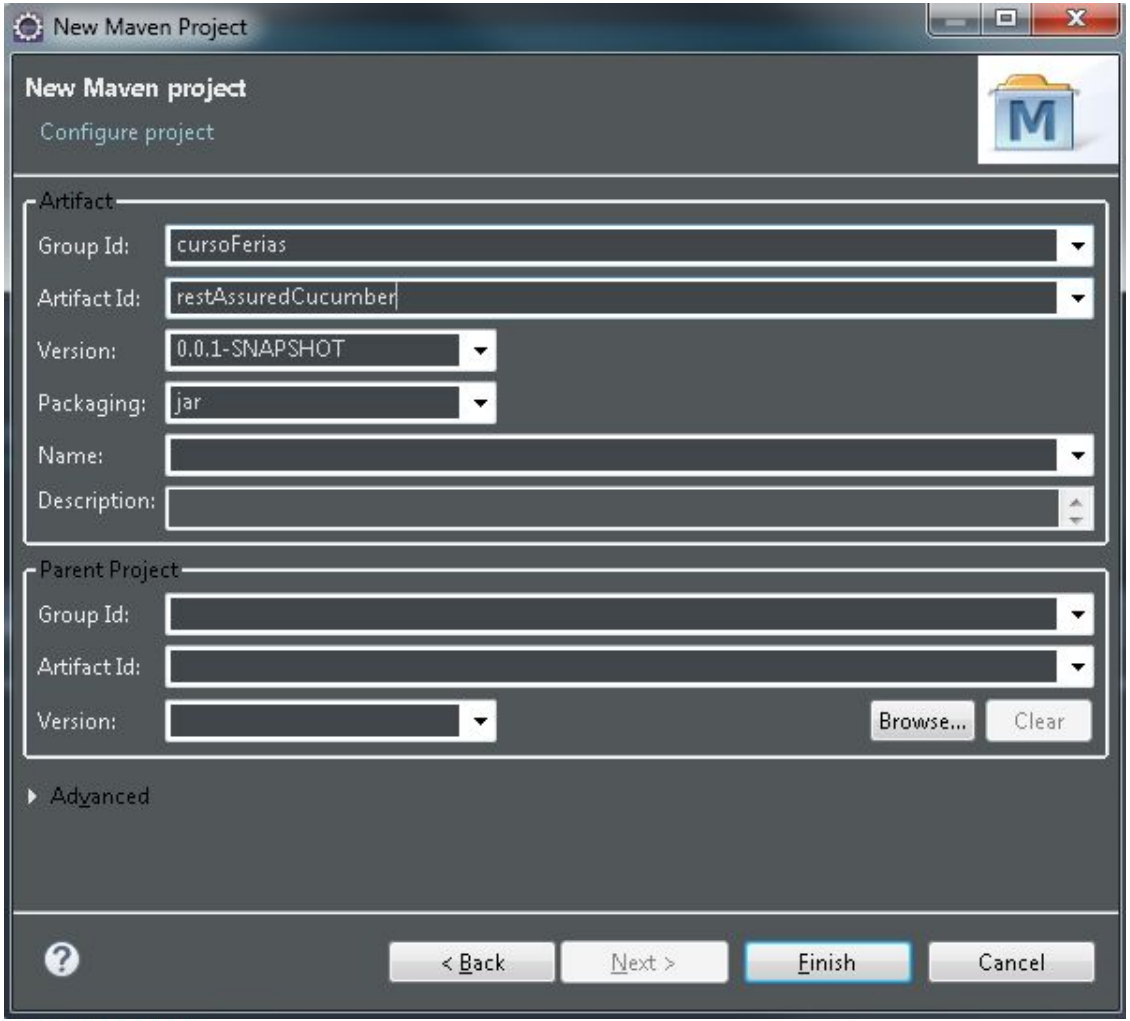
- GET
- POST
- PUT
- DELETE
- HEAD
- OPTIONS
- TRACE

De uma API e permitir fazer validações no header, body e schema da resposta.

Sem mais delongas, vamos criar testes automáticos dos métodos **GET** e **POST**.


New
Select a wizard
Create a Maven Project
Wizards:
Maven
Maven
Check out Maven Projects from SCM
Maven Module
Maven Project
< Back
Next >
Finish
Cancel


New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
More...
Advanced
< Back
Next >
Finish
Cancel


New Maven Project
New Maven project
Configure project
Artifact
Group Id:
cursoFerias
Artifact Id:
restAssuredCucumber
Version:
0.0.1-SNAPSHOT
Packaging:
jar
Name:
Description:
Parent Project
Group Id:
Artifact Id:
Version:
Browse...
Clear
Advanced
< Back
Next >
Finish
Cancel

No arquivo **pom.xml** adicionar as seguintes dependências:

```
<dependency>
    <groupId>org.hamcrest</groupId>
    <artifactId>hamcrest-all</artifactId>
    <version>1.3</version>
    <scope>test</scope>
</dependency>
```

- hamcrest-all: Possibilita a criação de regras de verificação (matchers) de forma declarativa;

Fonte: https://mvnrepository.com/

No arquivo **pom.xml** adicionar as seguintes dependências:

```
<dependency>
      <groupId>io.rest-assured</groupId>
      <artifactId>rest-assured</artifactId>
      <version>4.3.3</version>
</dependency>
```

- rest-assured: Biblioteca do framework rest-assured;

Fonte: https://mvnrepository.com/

No arquivo **pom.xml** adicionar as seguintes dependências:

```
<dependency>
      <groupId>io.rest-assured</groupId>
      <artifactId>json-path</artifactId>
      <version>4.3.3</version>
</dependency>
```

- json-path: Biblioteca responsável por ler as respostas das mensagens retornadas pela API;

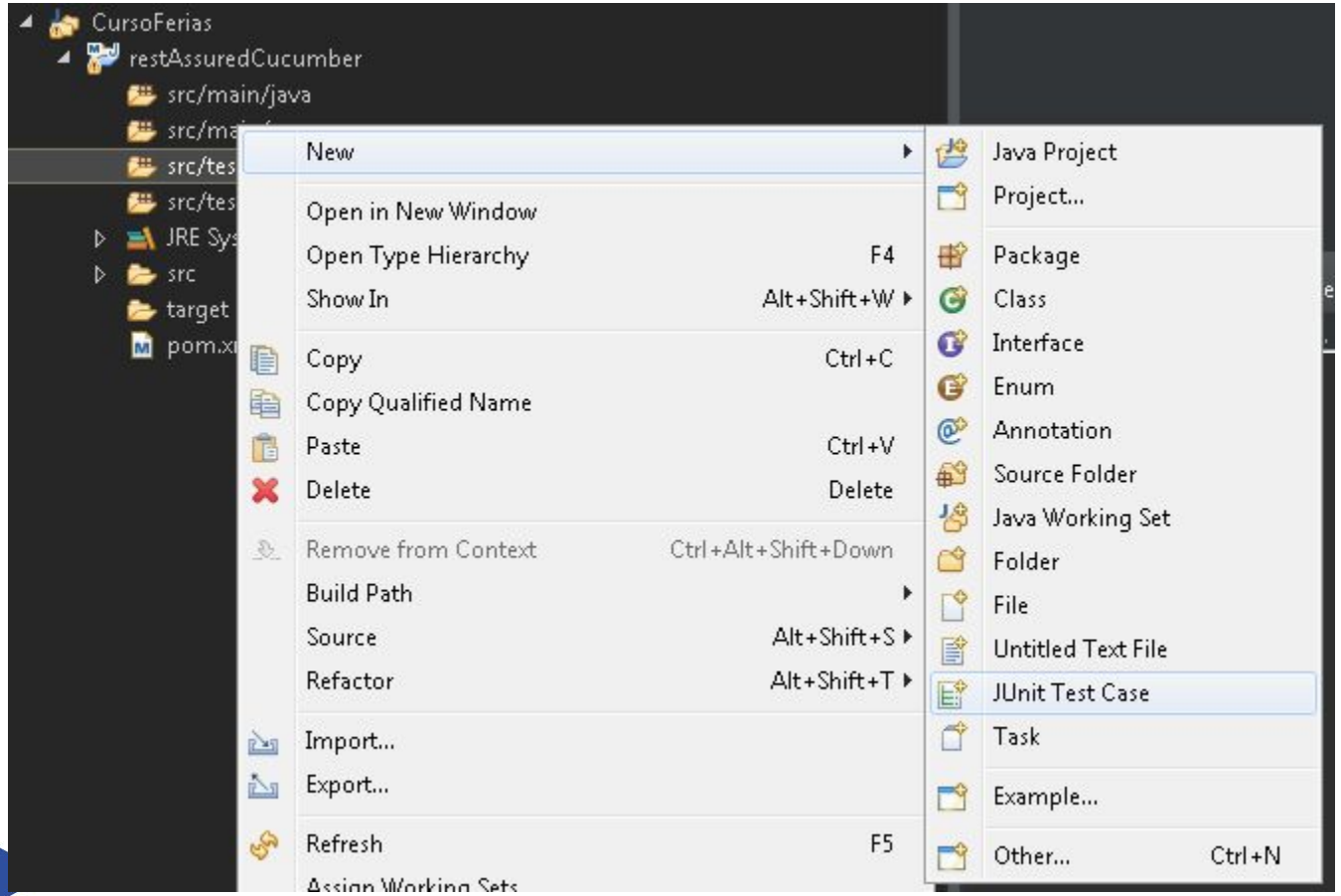
Fonte: https://mvnrepository.com/

No arquivo **pom.xml** adicionar as seguintes dependências:

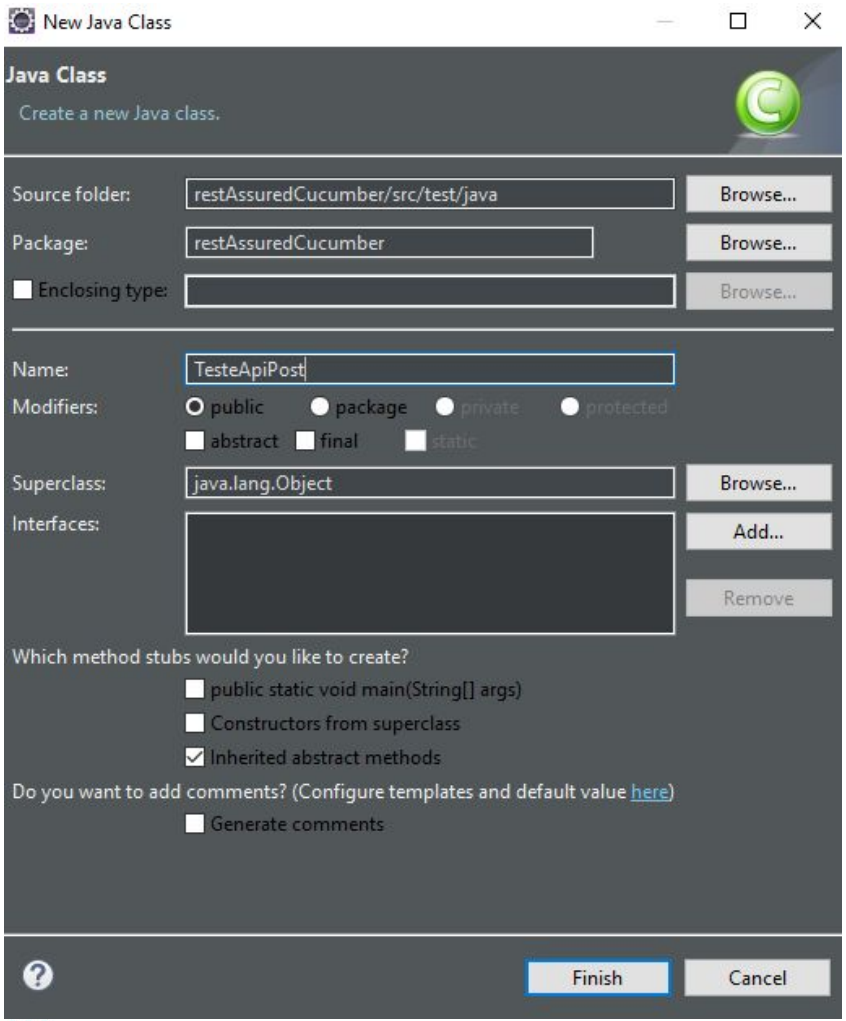
```
<dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
</dependency>
```

- junit: Biblioteca que auxilia na escrita de testes automatizados.

Fonte: https://mvnrepository.com/


CursoFerias
restAssuredCucumber
src/main/java
src/tes
src/tes
JRE Sys
src
target
pom.xr
New
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Assign Working Sets
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Task
Example...
Other... Ctrl+N

// Testes


New Java Class
Java Class
Create a new Java class.
Source folder:
restAssuredCucumber/src/test/java
Browse...
Package:
restAssuredCucumber
Browse...
Enclosing type:
Browse...
Name:
TesteApiPost
Modifiers:
public
package
private
protected
abstract
final
static
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel

Depois de criada a classe, faça o importe das seguintes bibliotecas:

- import static io.restassured.RestAssured.given;
- import io.restassured.http.ContentType;
- import static org.hamcrest.Matchers.*;

No nosso exemplo sera utilizado o site https://reqres.in/ que simula apis fake, utilizaremos a api https://reqres.in/api/login.

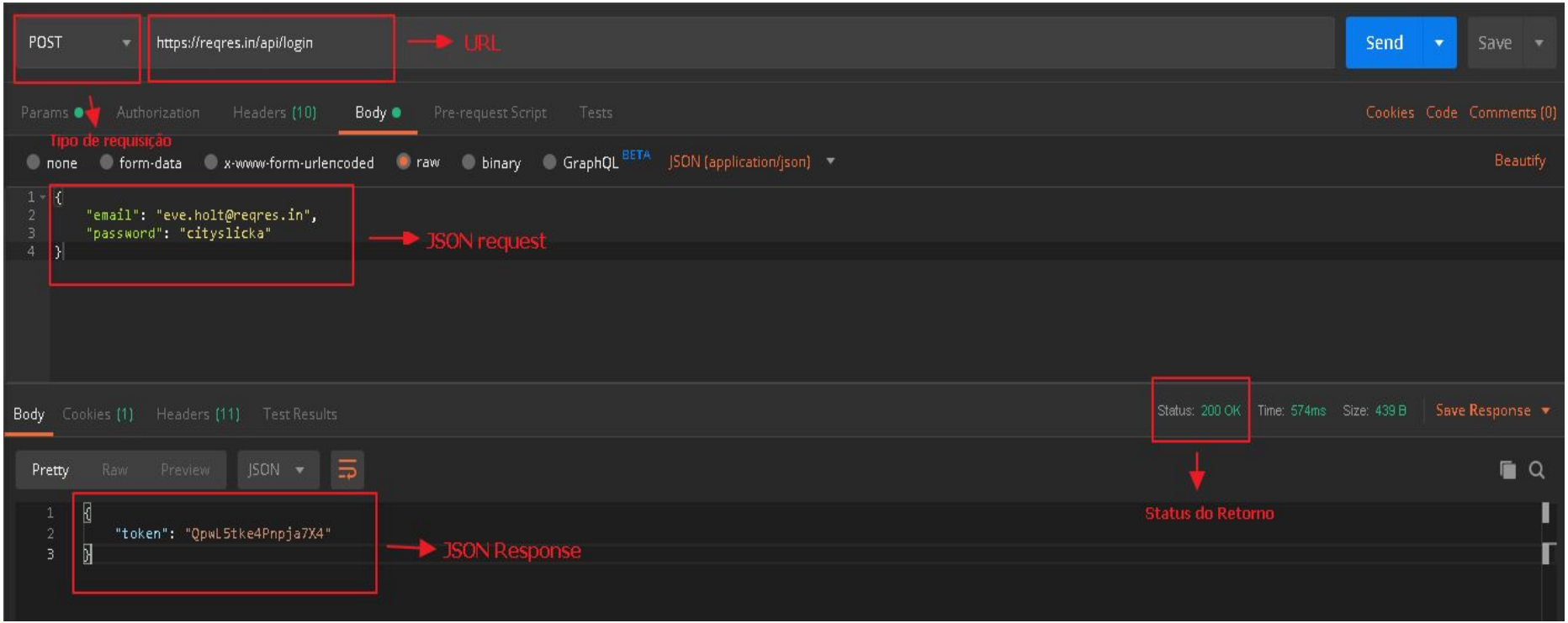
```
public class TesteAPIPost {

	@Test
	public void TestesPost(){
		String baseURI = "https://reqres.in/api/login";
	}
```

Agora vamos criar uma variável que recebe o nosso JSON com os atributos email e password.

```
public class TesteAPIPost {

    @Test
    public void TestesPost(){
        String baseURI = "https://reqres.in/api/login";
        String myJson = "{\"email\":\"eve.holt@reqres.in\",\"password\": \"cityslicka\"}";
    }
```

O próximo passo é criar um código que irá validar o envio de uma requisição do tipo POST. O serviço que vamos usar como exemplo, é um serviço que ao receber um JSON com os atributos email e password, retornará um token.

Vamos utilizar no nosso método Java a estrutura Given(Dado), When(Quando) e Then(Então) que basicamente é a estrutura de um BDD.


POST
https://reqres.in/api/login
URL
Send
Save
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Cookies
Code
Comments (0)
Tipo de requisição
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
BETA
JSON (application/json)
Beautify
{
"email": "eve.holt@reqres.in",
"password": "cityslicka"
}
JSON request
Body
Cookies (1)
Headers (11)
Test Results
Status: 200 OK
Time: 574ms
Size: 439 B
Save Response
Pretty
Raw
Preview
JSON
{
"token": "QpwL5tke4Pnpja7X4"
}
JSON Response
Status do Retorno

// Testes

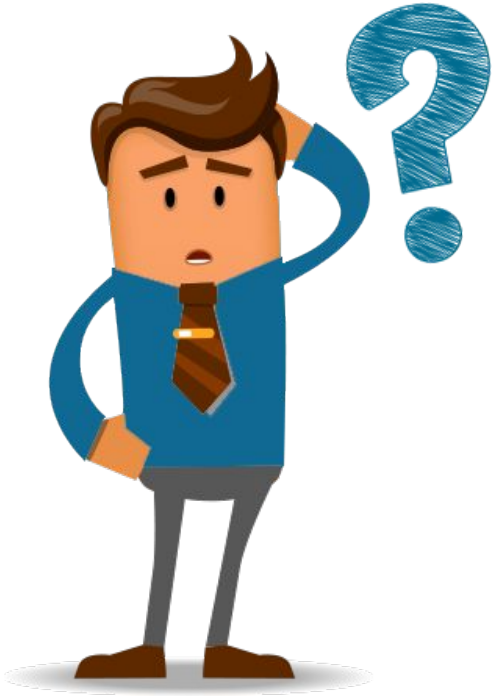
```
@Test
public void TestePost() {
        String baseURI = "https://reqres.in/api/login";
        String myJson = "{\"email\":\"eve.holt@reqres.in\",\"password\":
\"cityslicka\"}";

        given().
                contentType(ContentType.JSON).
                body(myJson).
        when().
                post(baseURI)..
        then().
                statusCode(200).
                body("token",
                containsString("QpwL5tke4Pnpja7X4"));
        }
```

Então o que esse código faz?

É muito simples, ele valida se a resposta recebida ao passar determinada informação para um serviço REST é a resposta esperada.

Em outras palavras:
Dado (given) que eu passe a URL https://reqres.in/api/login
E no body um JSON com os atributos email e password
E por meio do método POST
Quando (when) eu fizer a requisição
Então (then) o resultado esperado é que o status de retorno seja igual a 200
E o atributo token enviado no retorno contenha o código "QpwL5tke4Pnpja7X4".

Dúvidas?



matera

www.matera.com