

# Teste de API usando Postman

## O que é uma API?

**API** - Application Programming Interface (Interface de programação de aplicação)

**Rest** – arquitetura usada na API, REST significa Transferência Representacional de Estado

**Request** – requisição

**Response** – resposta

**Get**- um tipo de requisição que não modifica nada

**Status code** – é o status do código gerado

## Quais são os principais status que uma API pode retornar?

Considerando a RFC-2616, basicamente, os principais códigos de retornos quando o assunto seria API Rest rodando sobre o protocolo HTTP são 2xx para retornos de sucesso, 4xx para processamento não efetuado por conta da request e 5xx para processamento de erro interno na aplicação ou servidor

## Quais são os status HTTP?

Os códigos de status das respostas HTTP indicam se uma requisição HTTP foi corretamente concluída.

...

As respostas são agrupadas em cinco classes:

Respostas de informação ( 100 - 199 ),

Respostas de sucesso ( 200 - 299 ),

Redirecionamentos ( 300 - 399 )

Erros do cliente ( 400 - 499 )

Erros do servidor ( 500 - 599 ).

**Token** – uma sequencias de caracteres de autenticação

Jasonformatter.org – um site para organizar o resultado de uma requisição, é feita em Jason

O protocolo **HTTP** (Hypertext Transfer Protocol) define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada para um dado recurso;

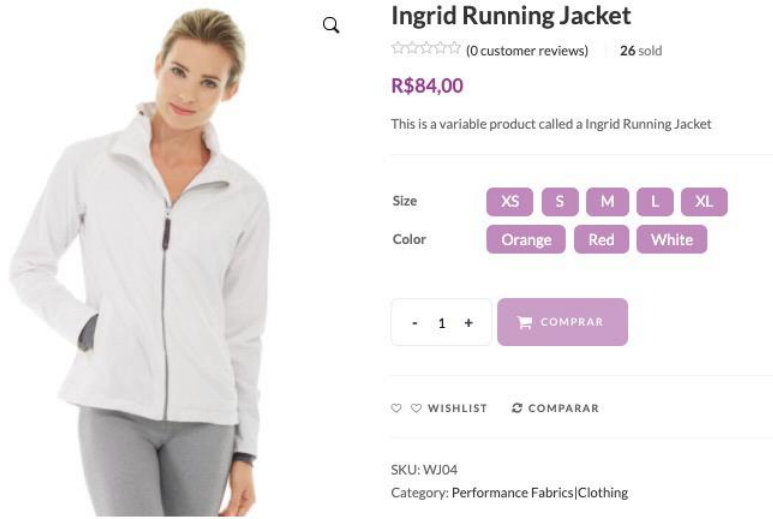
- Também são chamados de **Verbos** HTTP;

- Os principais são:

GET,POST,PUT, DELETE

- Outros métodos HEAD, CONNECT, OPTIONS, TRACE, PATCH.

O método GET solicita a representação de um recurso específico. Requisições utilizando o método GET devem retornar apenas dados, deve retornar apenas dados, exemplo, eu clico num determinado produto no site e ele me mostra mais detalhes do mesmo (nome, tamanho, cor, imagem, preço), ou seja as características do produto, as definições de layout (como css).



O método POST é utilizado para submeter uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso ou efeitos colaterais no servidor. Por exemplo um cadastro com e-mail e senha.

## Cadastro

Email address \*

ebac.qualidade@gmail.com

Password \*

.....

Forte

Seus dados pessoais serão usados para apoiar sua experiência em todo este site, para gerenciar o acesso à sua conta e para outros fins descritos em nossa política de privacidade.

ENVIAR

O método PUT substitui todas as atuais representações do recurso de destino pela carga de dados da requisição, por exemplo mudança de dados

Dados atuais:

## DETALHES DA CONTA




PAINEL	First name *	Last name *
PEDIDOS	Fábio	Araújo
DOWNLOADS	Display name *	
ENDEREÇOS	Fábio Araújo	
DETALHES DA CONTA	This will be how your name will be displayed in the account section and in reviews	
	Email address *	
	email@email.com	

## Dados novos:

### DETALHES DA CONTA

PAINEL	First name *	Last name *
PEDIDOS	Fábio	Araújo
DOWNLOADS	Display name *	
ENDEREÇOS	faraujo	
DETALHES DA CONTA	This will be how your name will be displayed in the account section and in reviews	
	Email address *	
	email_novo2email.com	

O método DELETE remove um recurso específico, por exemplo excluir um item do carrinho de compras (é feita uma requisição no servidor).

	Product Name	Unit Price	Stock Status	Add to cart	Remove
	Stellar Solar Jacket	R\$75,00	In Stock	 Ver Opções	 Remove this product
	Augusta Pullover Jacket	R\$57,00	In Stock	 Ver Opções	

## O que testar em uma API?

Segue uma lista de possíveis testes na API:

- Status:** o código de resposta está adequado (2xx, 3xx, 4xx e 5xx);
- Performance:** a resposta retornou dentro do tempo adequado;
- Syntaxe:** o tipo de conteúdo retornado está adequado (Content-Type) / o servidor aceita requisições no formato adequado;
- Tratamento de erro:** o servidor rejeita requisições no formato inadequado / excluir campos obrigatórios deve resultar em erro / requisições com tipos de dados inadequados deve resultar em erro;
- Deteção de erros:** testes negativos para identificar exceções;
- Schema:** o conteúdo da resposta está de acordo com a estrutura ou formato esperado (contrato);
- Funcional:** o servidor retorna o valor previsto de acordo com a requisição / a requisição insere,

atualiza ou exclui um recurso esperado;

**8. Segurança:** Injeções de SQL não impactam na integridade dos dados.

**JSON** (JavaScript Object Notation - Notação de Objetos

JavaScript) é uma formatação leve de troca de dados.

- Para seres humanos, é fácil de ler e escrever. Para máquinas, é fácil de interpretar e gerar.
- JSON é em formato texto e completamente independente de linguagem ideal para troca de dados.

### Objeto:

Um **objeto** é um conjunto desordenado de pares nome/valor.

- Um objeto começa com { chave de abertura e termina com chave de fechamento }.
- Cada nome é seguido por : dois pontos e os pares nome/valor são seguidos por , vírgula.

```
{  
  "id": 123,  
  "nome": "Fábio",  
  "email": "fabio@ebac.com.br"  
}
```

### Array:

Uma **array** é uma coleção de valores ordenados.

- O array começa com [ colchete de abertura e termina com colchete ] de fechamento.
- Os valores são separados por , vírgula.

```
[1,2,3,5,8,13,21]
```

```
["banana", "maçã", "melancia", "uva"]
```

```
[  
  { "  
  usuario": "  
  admin",  
  "senha": "psw!123"  
  },  
  { "  
  "
```

## Valor:

Um **valor** pode ser uma:

- cadeia de caracteres (string)
- um número
- true ou false
- null
- objeto
- array

```
{  
  "id": 123,  
  "nome": "Fábio",  
  "email": "fabio@ebac.com.br",  
  "admin": "true"  
}
```

## Swagger

Swagger é um conjunto de ferramentas para desenvolvedores de API da SmartBear Software e uma especificação anterior na qual a especificação OpenAPI é baseada.

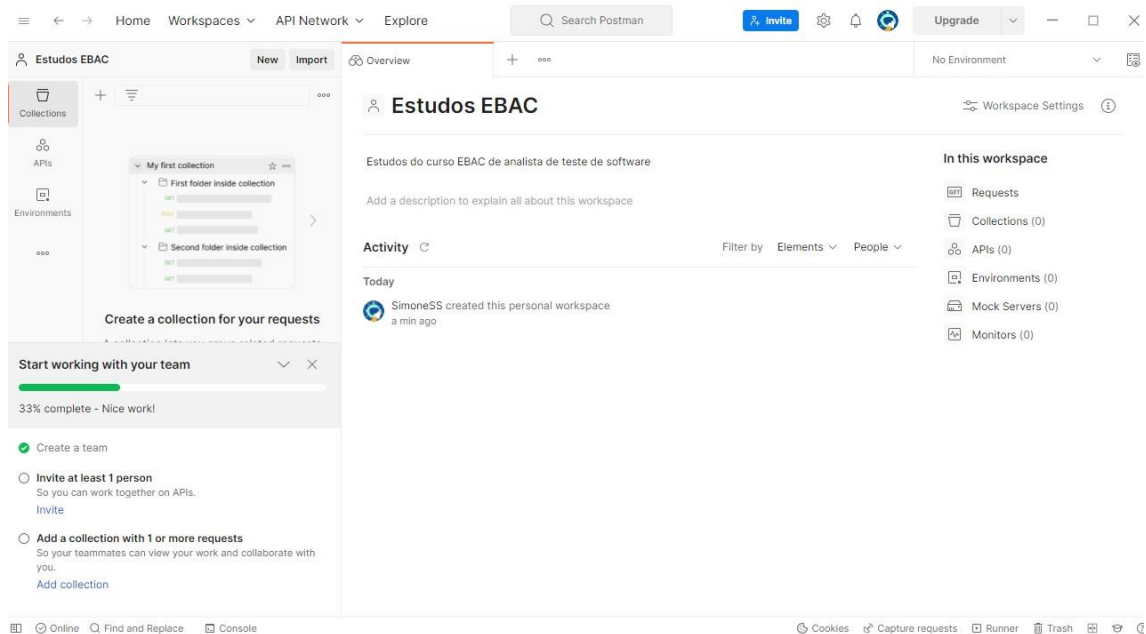
Neste contexto, a principal contribuição do Swagger é garantir a padronização das interfaces de integração. Com isso, sempre que preciso, qualquer desenvolvedor pode ter acesso aos parâmetros necessários para a correta integração com seu sistema, por exemplo

<https://serverest.dev/> - abre o Swagger

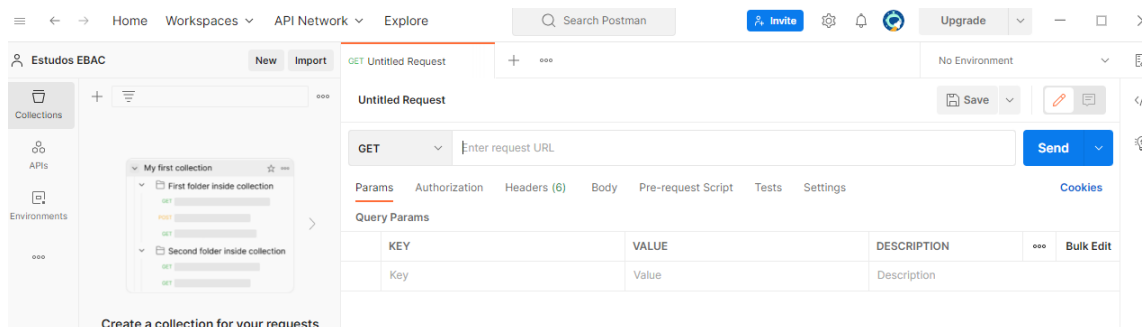
O Postman é uma ferramenta que proporciona criar, compartilhar, testar e documentar APIs. Permite aos usuários criar e salvar solicitações HTTP e HTTPS simples e complexas, bem como ler suas respostas.

<https://www.postman.com/downloads/> (faça o download)

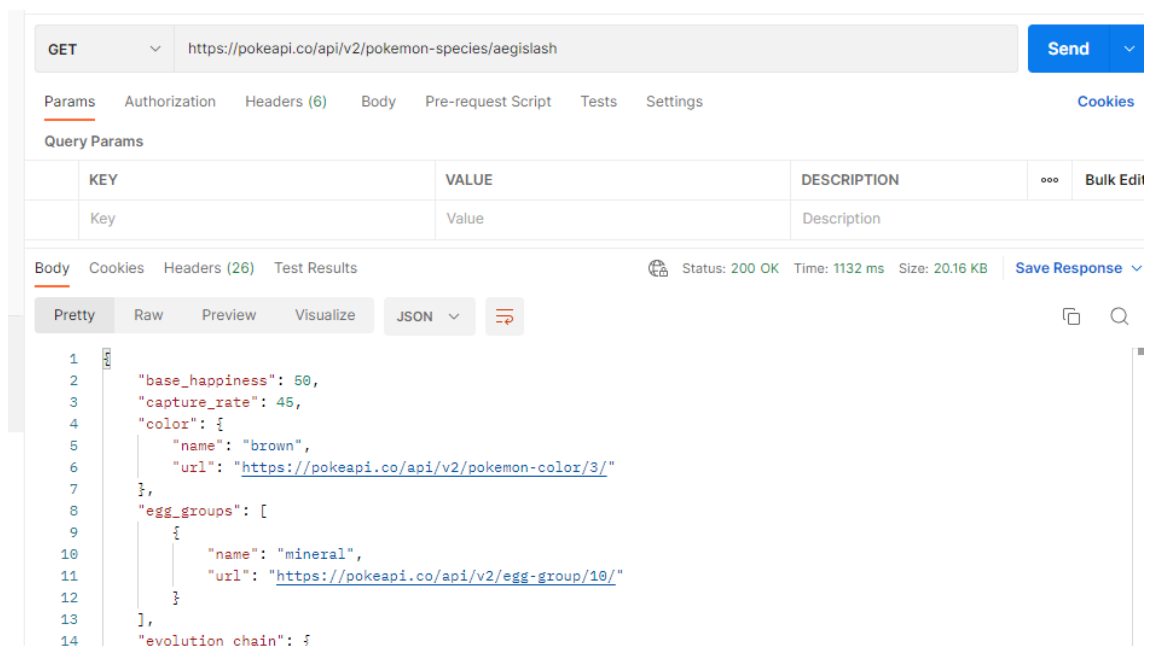
Faça a instalação do Postman e crie uma conta, crie também um workspace (criei um de nome “Estudos EBAC”)

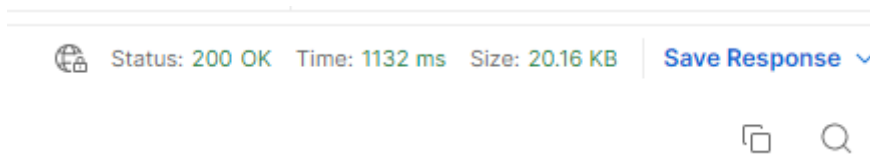


Clicando no sinal de + na parte superior da tela, aparecerá uma tela com o método GET (você pode escolher qualquer método).



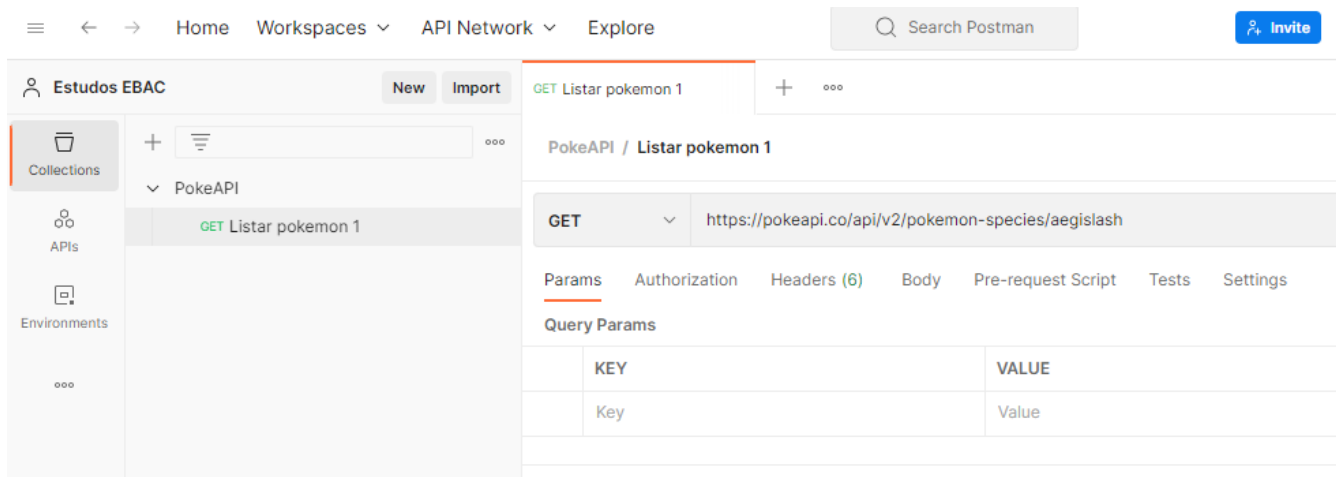
Coloque o endereço <https://pokeapi.co/api/v2/pokemon-species/aegislash> (é uma api do pokemon), veja o resultado:



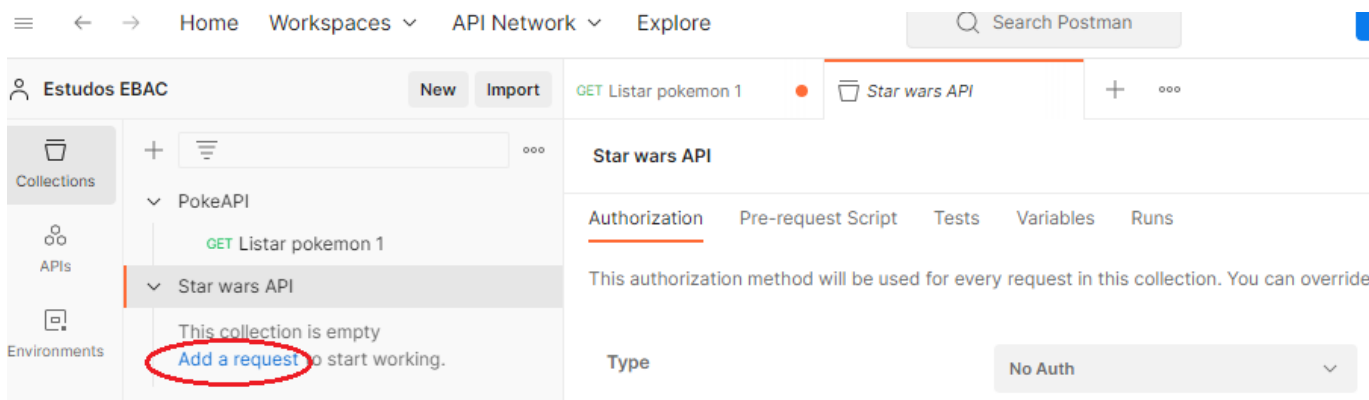


É possível ver o Status, o Tempo e o Tamanho, conforme imagem acima.

Clique em “Save” no canto superior direito, adicione um nome, coloquei “Listar pokemon 1”. Vai dizer que você não tem Collection (coleção) crie uma (criei uma com o nome PokeAPI)



Acima é mostrado o Postman com o teste salvo Listar pokemon 1. Agora crie uma nova coleção com o nome Star wars API, clicando sobre ele é apresentada a tela abaixo:



Clique em “Add a request” para fazer um novo teste. No navegador acesse o endereço da API <https://swapi.dev/> copie a uri <https://swapi.dev/api/> junto com o end point [people/1/](https://swapi.dev/api/people/1/) vai ficar tudo junto assim: <https://swapi.dev/api/people/1/agora> clique em “Send”

Vai ser apresentadas as informações abaixo:

GET <https://swapi.dev/api/people/1/> Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookiec

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk E
Key	Value	Description		

Body Cookies Headers (10) Test Results Status: 200 OK Time: 2.00 s Size: 974 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "name": "Luke Skywalker",
3   "height": "172",
4   "mass": "77",
5   "hair_color": "blond",
6   "skin_color": "fair",
7   "eye_color": "blue",
8   "birth_year": "19BBY",
9   "gender": "male",

```

Essa API traz informações do personagem Luke Skywalker (suas características).

Renomeie o nome do teste (nome atual é New request) clicando nos três pontinhos na frente do nome, escolha a opção “Rename” e coloque o nome como “Listar personagem 1”, na mesma opção (três pontinhos) escolha a opção “Duplicate” para duplicar o teste, renomeia com o nome “Listar personagem 2”. Na URI acione o número 2 na frente do endereço assim: <https://swapi.dev/api/people/2/>, clique em “Send”

Desta vez vai ser mostrado outro personagem (C-3PO) junto com as características, assim:

APIs Star wars API

- GET Listar personagem 1
- GET Listar personagem 2

Environments

Start working with your team

33% complete - Nice work!

- ☒ Create a team
- ☐ Invite at least 1 person
  - So you can work together on APIs.
  - [Invite](#)
- ☐ Add a collection with 1 or more requests
  - So your teammates can view your work and collaborate with

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

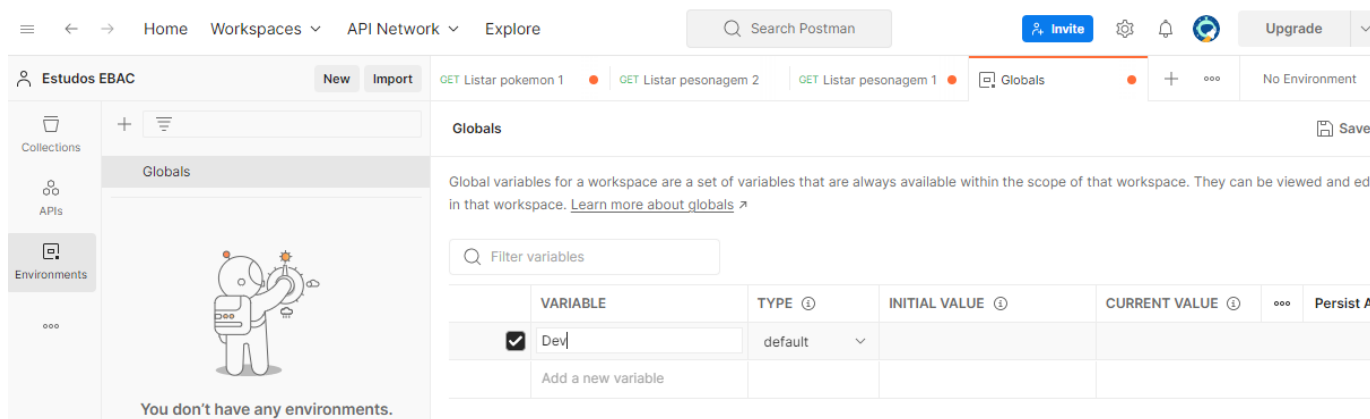
```

1 {
2   "name": "C-3PO",
3   "height": "167",
4   "mass": "75",
5   "hair_color": "n/a",
6   "skin_color": "gold",
7   "eye_color": "yellow",
8   "birth_year": "112BBY",
9   "gender": "n/a",
10  "homeworld": "https://swapi.dev/api/planets/1/",
11  "films": [

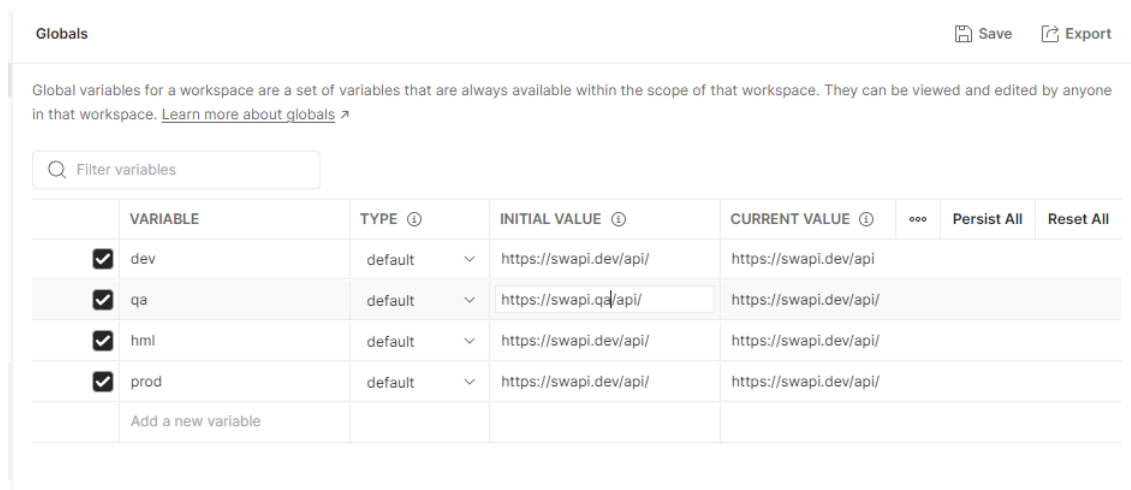
```

O que podemos fazer agora, para um reaproveitamento é criar uma variável de ambiente, com a URL base <https://swapi.dev/api/people/2/> para tal, clique na opção “Environments”(Ambientes) do lado esquerdo da tela, e depois clique em “Globals” veja a tela abaixo:





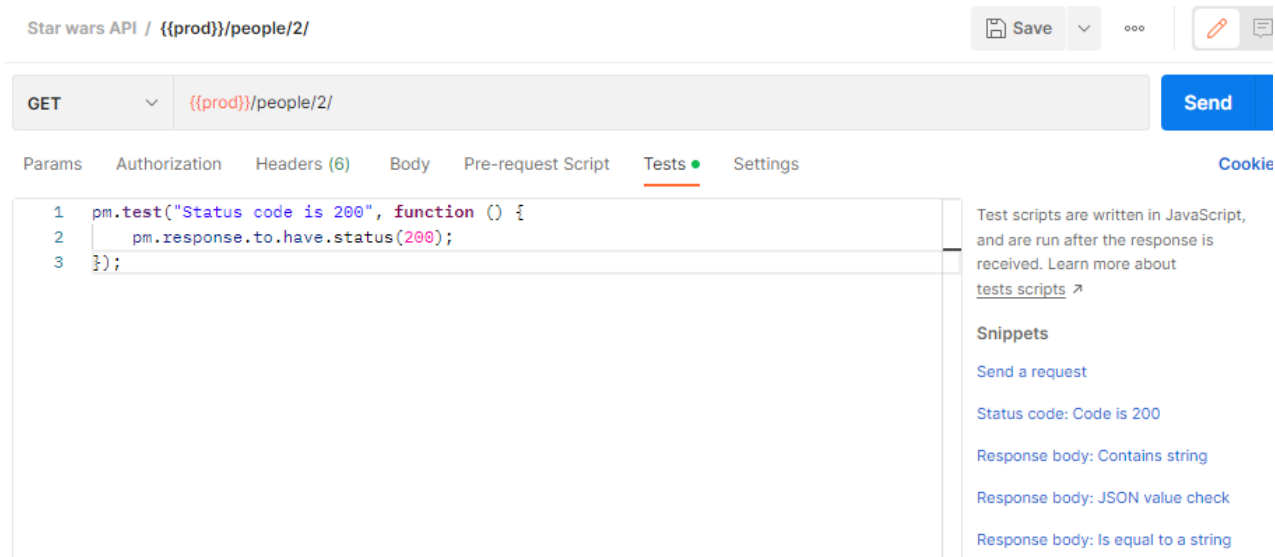
Embaixo da opção “Variable” coloque um nome (coloquei Dev de desenvolvimento), mas pode ser criadas outras com outros nomes como qa, hml de homologação, prod de produção:



Depois dos nomes basta adicionar a URL na opção “Initial Value” e em “Current Value” assim: <https://swapi.dev/api/people/2/> (tem que mudar o .dev para .qa para a opção qa e assim sucessivamente). Salve, e de volta ao teste Listar personagem 2, na URL digite **{{dev}}** a variável dentro de duas chaves e clique em “Send” o teste deve continuar funcionando. Esse procedimento é muito útil quando se quer comparar um ambiente com outro.

Agora crie uma nova requisição clicando no botão + no canto superior direito, na URI digite **{{prod}}** junto com o end point **/people/2/** vai ficar assim: **{{prod}}/people/2/** clique em “Send”, o teste tem que continuar funcionando (trazendo os dados do personagem C-3PO).

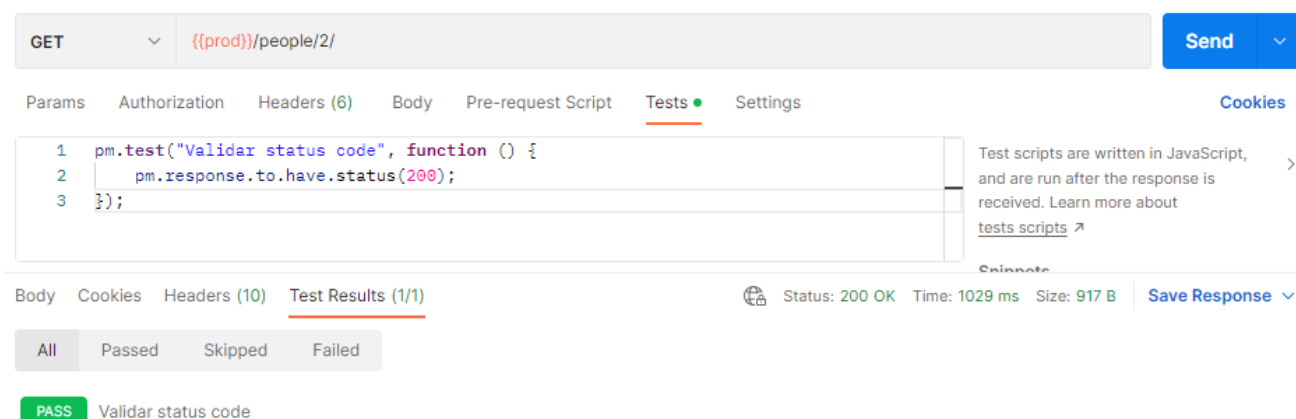
Ainda nesse teste, clique na opção “Test results” vai reparar que está vazio, sem nenhum resultado, clique na opção “Test” que aparece logo acima da tela.



Escolha um script pronto (Status code is 200) do lado direito da tela, como mostra a imagem acima. Você pode mudar o nome do script conforme foi feito abaixo:

```
pm.test("Validar status code", function () {
  pm.response.to.have.status(200);
});
```

Agora clique em “Send” o teste deve passar dando o resultado como 200, conforme abaixo:



Agora escolha o script “Response body: contain string” e coloque um nome para o script, assim:

```
pm.test("Validar nome do personagem", function () {
  pm.expect(pm.response.text()).to.include("string_you_want_to_search");
});
```

Copie o nome do personagem C-3PO e adicione no lugar de ("string\_you\_want\_to\_search"); clique em “Send”. Agora tem que aparecer dois testes passados conforme imagem abaixo:

Star wars API / {{prod}}/people/2/

GET {{prod}}/people/2/

Params Authorization Headers (6) Body Pre-request Script Tests Settings

```
1 pm.test("Validar status code", function () {
2   |   pm.response.to.have.status(200);
3   | });
4
5 pm.test("Validar nome do personagem", function () {
6   |   pm.expect(pm.response.text()).to.include("C-3PO");
7   | });
```

Body Cookies Headers (10) Test Results (2/2)

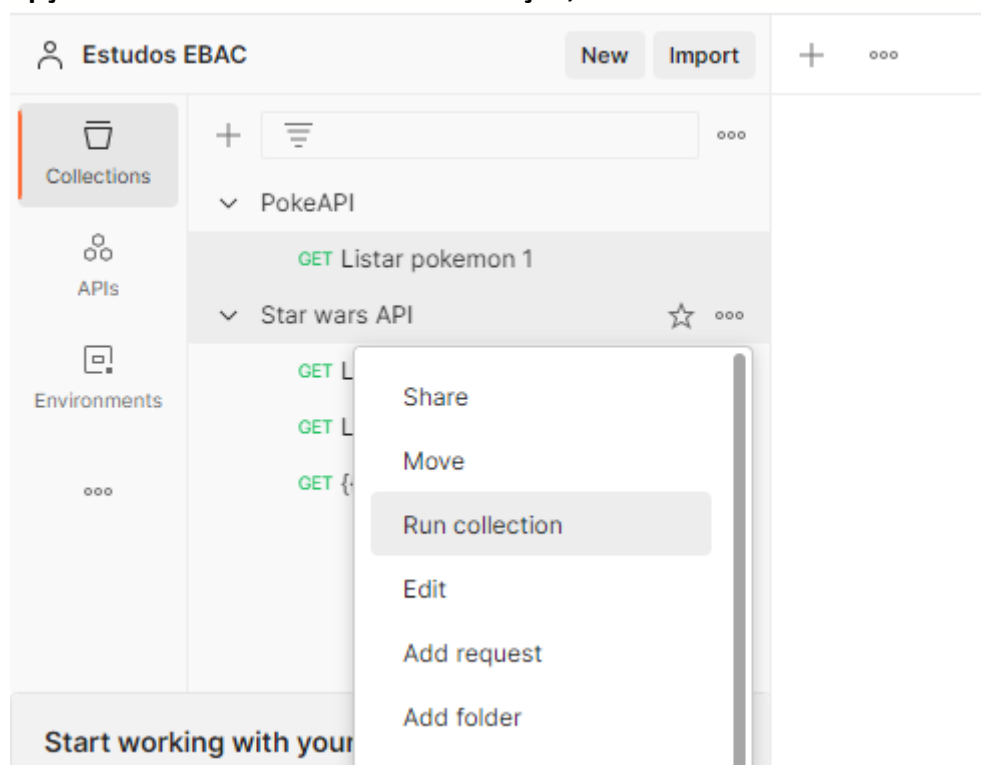
Status: 200 OK Time: :

All Passed Skipped Failed

PASS Validar status code

PASS Validar nome do personagem

Clicando nos três pontinhos ... na frente de uma collection é possível escolher a opção de rodar todos os testes da coleção, assim:



Basta clicar em “Run collection”, será apresentada a seguinte tela:

Runner

+ ...

No Enviro

RUN ORDER

Deselect All | Select All | Reset

☒ GET Listar pesonagem 1

☒ GET Listar pesonagem 2

☒ GET {{prod}}/people/2/

Choose how to run your collection

☒ Run manually  
Run this collection in the Collection Runner.

☐ Schedule runs  
Periodically run collection at a specified time on the

☐ Automate runs via CLI  
Configure CLI command to run on your build pipelin

Run configuration

Iterations

1

Delay

0

ms

Data

Select File

Onde você pode selecionar os testes que quer rodar, clicando e arrastando é possível mudar as ordens de execução.

Em “Run configuration” é possível colocar interações no campo “Iterations” (que simula usuários acessando a API ao mesmo tempo), vou colocar 10, em “Delay” é possível colocar o tempo entre uma requisição e outra, vou colocar 500 milissegundos (que é a metade de um segundo).

RUN ORDER

Deselect All | Select All | Reset

☒ GET Listar pesonagem 1

☒ GET Listar pesonagem 2

☒ GET {{prod}}/people/2/

Choose how to run your collection

☒ Run manually  
Run this collection in the Collection Runner.

☐ Schedule runs  
Periodically run collection at a specified time on the

☐ Automate runs via CLI  
Configure CLI command to run on your build pipelin

Run configuration

Iterations

10

Delay


500

ms

Agora basta rodar os testes. Ele irá fazer 10 requisições, e todas devem passar.

## Star wars API - Run results

[Run Again](#)[A](#)

 Run on Today, 15:08:55 · [View all runs](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	10	32s 151ms	20	538 ms

[All Tests](#)   Passed (20)   Failed (0)   Skipped (0)

**GET** `{{prod}}/people/2/` `https://swapi.dev/api/people/2/` `/ {{prod}}/people/2/`

Pass   Validar status code

Pass   Validar nome do personagem

É possível exportar o resultado dos testes, clicando na opção “Export results”.