



# Testes de APIs REST

Unidade Curricular de Aplicações Distribuídas / AID

[Versão 1.0] novembro 2021

[Versão 1.1] 8 dezembro 2022



Instituto Politécnico de Castelo Branco  
Escola Superior de Tecnologia



# Sumário

---

- Testes de APIs com POSTMAN
  - Workspace - Funcionalidades Principais
  - Componentes de um Pedido REST
  - Criação de Testes à API
  - Pré-Scripts e Scripts de Testes



# Teste do WebService RESTful (“à mão”)

- Usar um browser (apenas método GET)
- Usar um browser + plugin Postman
- Usar o utilitário curl
  - Disponível em: <http://curl.haxx.se/dlwiz/>

## **Exemplo de Invocação de um método HTTP GET**

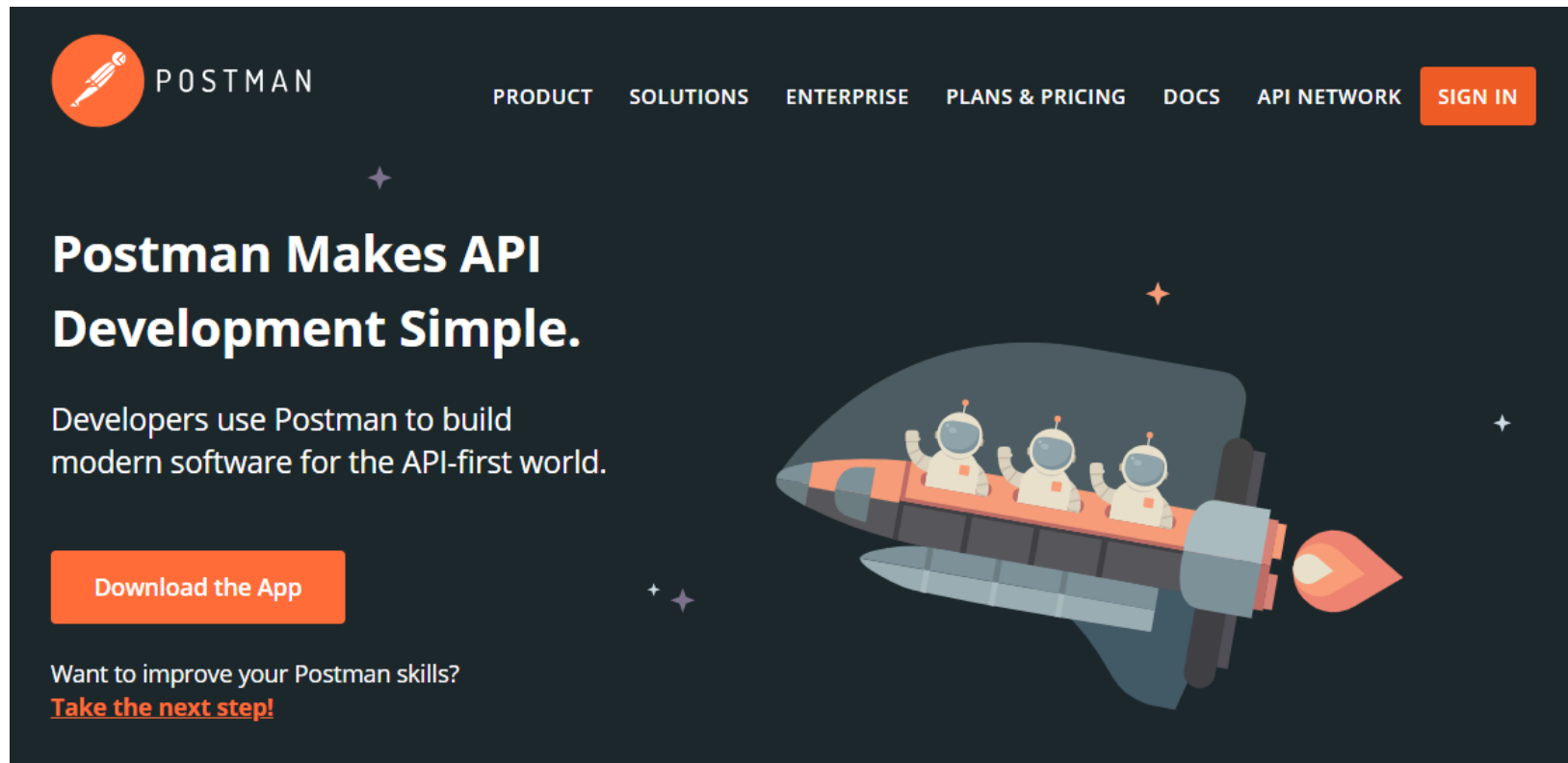
```
curl -XGET http://localhost:8080/ServidorRS/resources/cliente
```

## **Exemplo de Invocação de um método HTTP POST**


```
curl -XPOST -HContent-type:text/xml --data  
"<customer><id>321</id><firstName>Alexandre</firstName><middleName>José</middleName><lastName>da  
Fonte</lastName></customer>"  
http://localhost:8080/raizRS/resources/cliente
```



# Postman



The image shows the hero section of the Postman website. It features a dark blue background with a stylized illustration of a rocket ship with three astronauts. The rocket is orange and grey, with a large orange flame at the bottom. The astronauts are in white suits with orange accents. The background is decorated with small white stars and a larger orange star.

 POSTMAN

PRODUCT SOLUTIONS ENTERPRISE PLANS & PRICING DOCS API NETWORK [SIGN IN](#)

## Postman Makes API Development Simple.

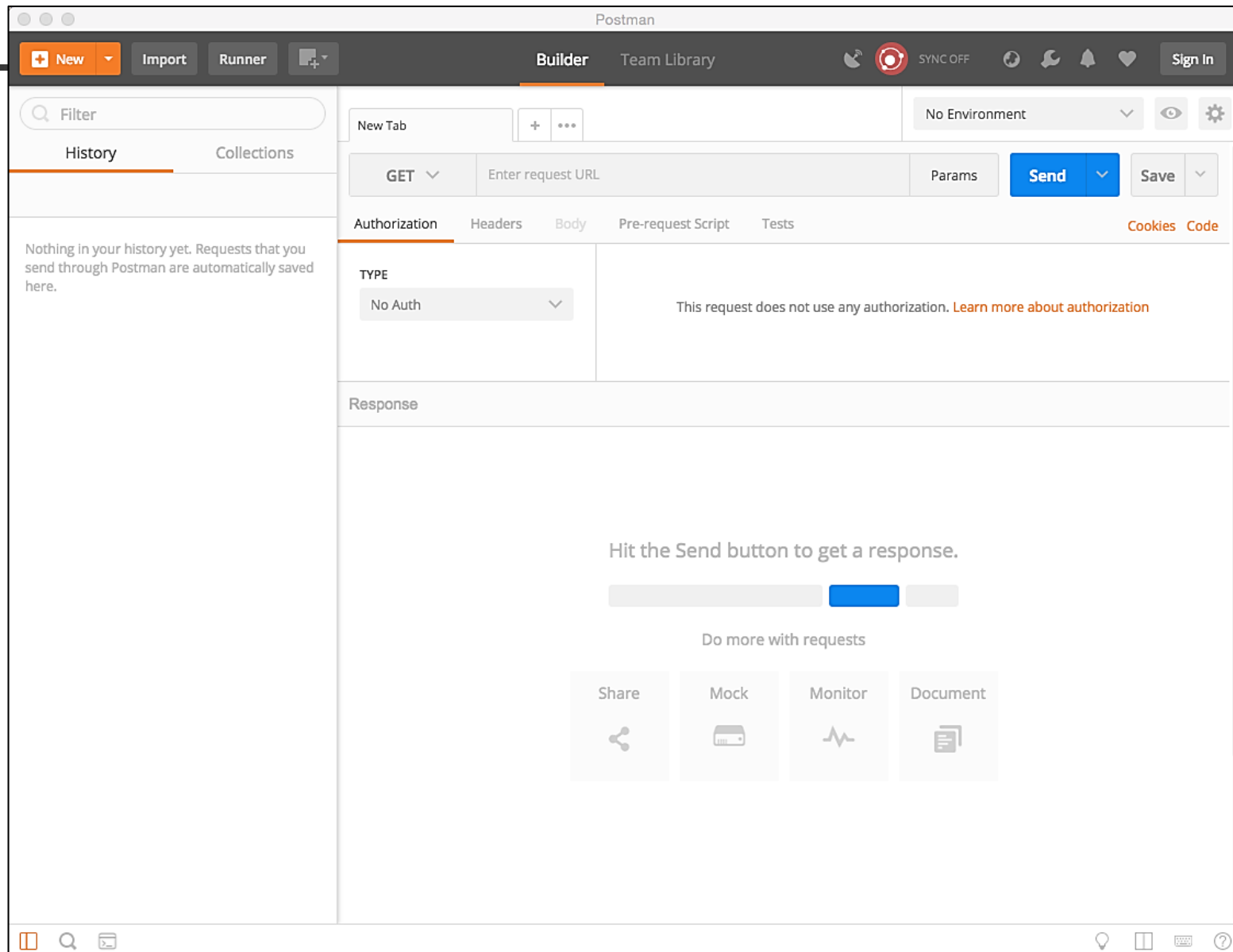
Developers use Postman to build modern software for the API-first world.

[Download the App](#)

Want to improve your Postman skills?  
[Take the next step!](#)



# Postman





# Postman – Workspace

## Funcionalidades Principais

---

- **Novo** - Criar um novo pedido, ou colecção.
- **Importar** - Para importar uma colecção ou ambiente.
- **Runner** - Os testes podem ser executados manualmente ou automatizados através do Collection Runner.
- **My Workspace** - Pode criar um novo espaço de trabalho individualmente ou como uma equipa.
- **Convidar** - Colabore num espaço de trabalho convidando membros da equipa.
- **Colecções** - Organizar a *suite* de testes através da criação de colecções. Cada colecção pode ter subpastas e múltiplos pedidos. Um pedido ou pasta também pode ser duplicado.



# Postman – Workspace

## Funcionalidades Principais

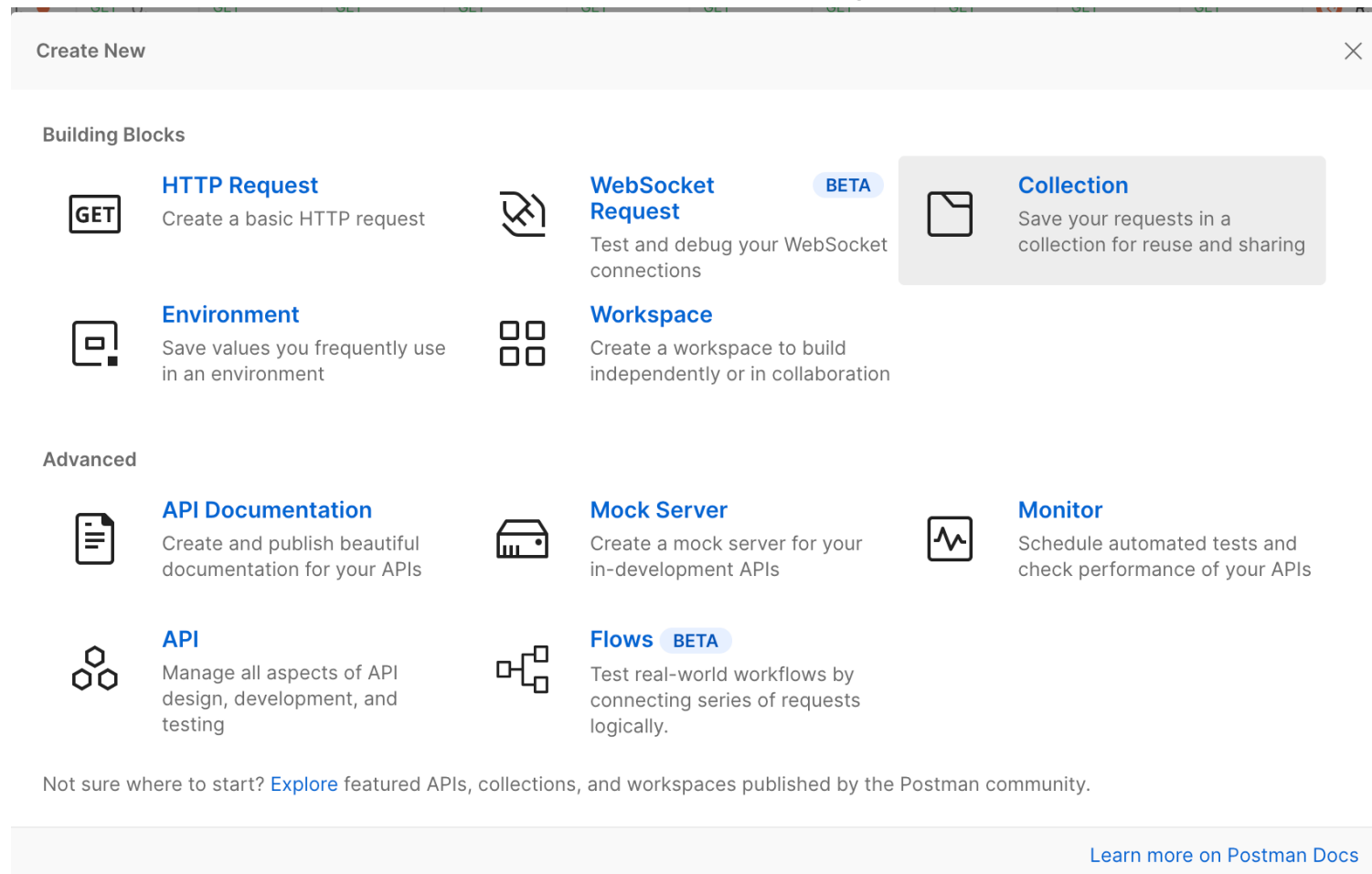
---

- Aba dos Pedidos- Isto mostra o título do pedido em que está a trabalhar. Por defeito, "Pedido sem título" seria exibido para pedidos sem títulos.
- Pedido HTTP - lista de diferentes pedidos, tais como GET, POST, COPY, DELETE, etc. Os pedidos mais utilizados são GET e POST.
- URL - Também conhecido como endpoint, é aqui que identificará o link para onde a API irá comunicar.
- Params – edição dos parâmetros necessários para um pedido, tais como valores de uma chave.
- Cabeçalho Autorização - Para aceder às APIs, é necessária uma autorização adequada. Pode ter a forma de um nome de utilizador e senha, token portador, etc.
- Cabeçalhos - Pode definir cabeçalhos tais como o tipo de conteúdo JSON, dependendo das necessidades da organização.
- Corpo - É aqui que se pode personalizar os detalhes num pedido comumente utilizado em pedidos POST.
- Script Pré-Pedido - Estes são scripts que serão executados antes do pedido.
- Testes - Estes são os scripts executados durante o pedido. É importante ter testes, uma vez que estabelece pontos de verificação para verificar se o estado da resposta é OK, os dados retornados são os esperados e outros testes



# Criar uma Coleção de Pedidos

- New-> Collection
  - Indicar o nome da nova Coleção



Após a criação da Coleção é possível ainda criar Pastas de Testes





# Para criar um pedido REST são precisos 5 componentes:

---

- 1 Seleccionar o Método (GET, POST, PUT, etc)
- 2 Introduzir o URL
- 3 Introduzir os Headers
- 4 Introduzir o corpo (excepto GET ou DELETE)
- 5 Introduzir as credenciais de autenticação (pode ser no header ou no corpo)





# Postman – Workspace

## Funcionalidades Principais

---

- Exercícios Exemplo
  - Criar um pedido GET
  - Criar um pedido POST (Submissão de Objectos JSON/Doc XML)
  - Parametrização de pedidos HTTP (Passagem de parâmetros)
  - Parameterização de Dados no POSTMAN para automatização de pedidos com dados repetidos
  - Criação de Testes Básicos

Consultar o Learning Center Postman em

<https://learning.postman.com/docs/getting-started/introduction/>



# Exemplo Pedido GET

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

The screenshot shows a REST client interface with the following components:

- Request Bar:** Method `GET` (1), URL `https://jsonplaceholder.typicode.com/users` (2), and `Send` button (3).
- Params Table:** A table with columns `KEY`, `VALUE`, and `DESCRIPTION`. It contains one row with `Key`, `Value`, and `Description`.
- Response Bar:** Status `200 OK` (4), Time `784 ms`, and Size `6.27 KB`.
- Response Body:** A JSON array (5) containing two user objects. The first object is: 

```
{  "id": 1,  "name": "Leanne Graham",  "username": "Bret",  "email": "Sincere@april.biz",  "address": {    "street": "Kulas Light",    "suite": "Apt. 556",    "city": "Gwenborough",    "zipcode": "92998-3874",    "geo": {      "lat": "-37.3159",      "lng": "81.1496"    }  },  "phone": "1-770-736-8031 x56442",  "website": "hildegard.org",  "company": {    "name": "Romaguera-Crona",    "catchPhrase": "Multi-layered client server neural-net",    "bs": "harness real-time e-markets"  }}
```

Alexa

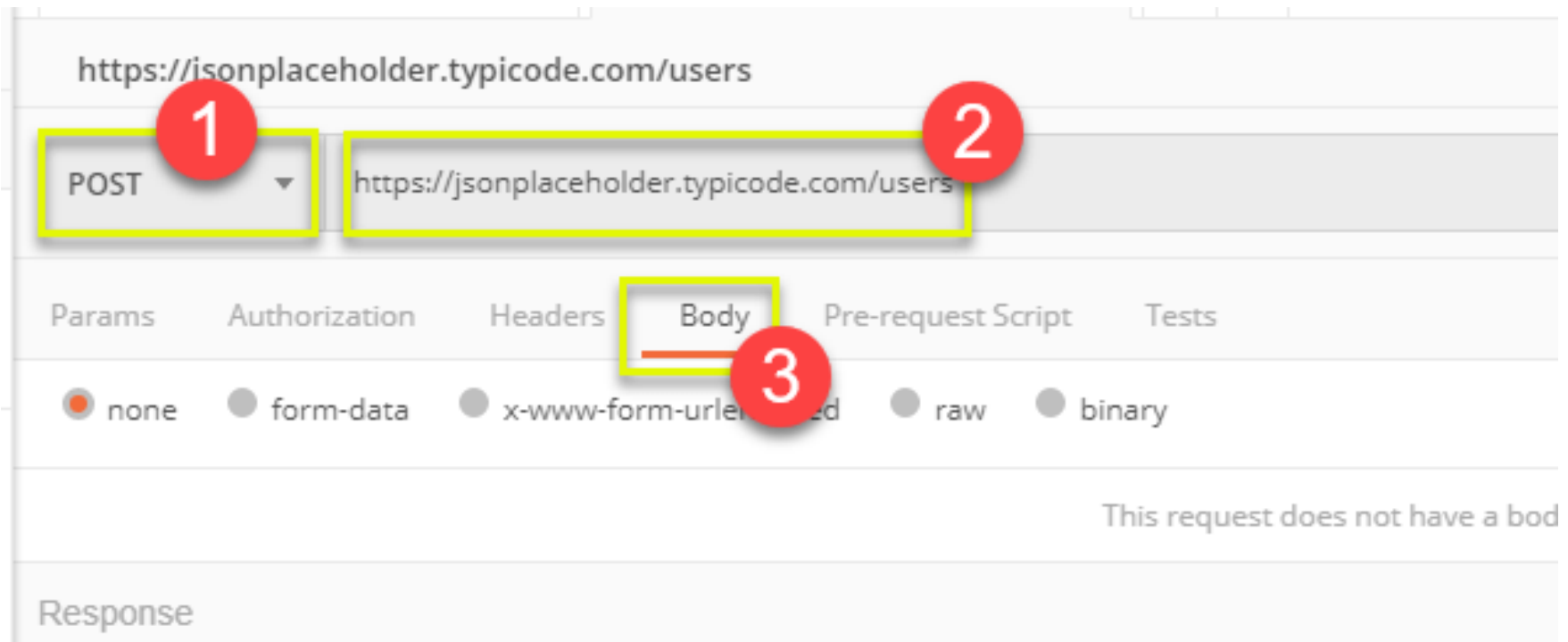
Learn 122



# Exemplo Pedido POST

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

- Criar um User usando JSON



1. Defina o seu pedido HTTP para POST.
2. No campo URL do pedido
3. Seleccionar a Aba Body
4. Seleccionar raw → text → JSON



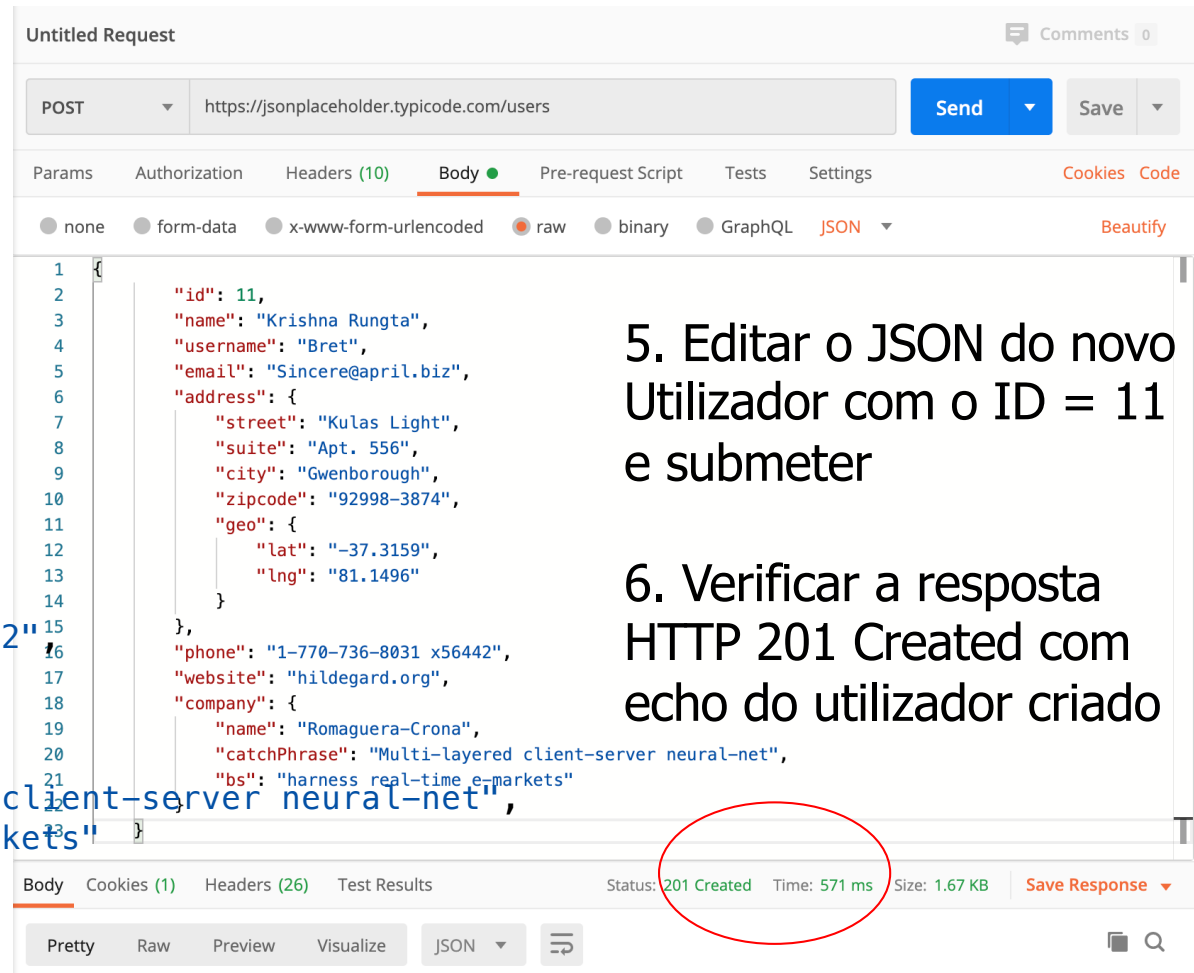
# Exemplo Pedido POST

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

## ■ Criar um User usando JSON (Cont.)

```
{
  "id": 11,
  "name": "Krishna Rungta",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
}
```

Alexandre Fonte



5. Editar o JSON do novo Utilizador com o ID = 11 e submeter

6. Verificar a resposta HTTP 201 Created com echo do utilizador criado



## Exemplo Pedido GET com passagem de PATHPARAM

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

- Parameterização PATHPARAM de um pedido GET
  - Exemplo consulta os dados do utilizador com id =10

GET <https://jsonplaceholder.typicode.com/users/10> Send Save

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies (1) Headers (26) Test Results Status: 200 OK Time: 66 ms Size: 1.59 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 10,
3   "name": "Clementina DuBuque",
4   "username": "Moriah.Stanton",
5   "email": "Rey.Padberg@karina.biz",
6   "address": {
7     "street": "Kattie Turnpike",
8     "suite": "Suite 198",
9     "city": "Lebsackbury",
10    "zipcode": "31428-2261".
```



# Exemplo Pedido POST

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

- Parameterização de Dados POSTMAN usando variáveis

The screenshot shows the Postman interface for a POST request to `https://jsonplaceholder.typicode.com/users`. The URL bar contains `{{url}}/users`. The request method is set to POST, and the body type is JSON (application/json). The body content is a JSON object with the following fields:

```
{
  "id": "{{id}}",
  "name": "{{name}}",
  "username": "{{username}}",
  "email": "{{email}}",
  "address": {
    "street": "Live Oak Street",
    "suite": "Apt. 556",
    "city": "Cavite",
  }
}
```

Annotations in the image highlight the use of double curly braces `{{variable}}` for parameterization.

1. Referenciar a variável com chavetas duplas.

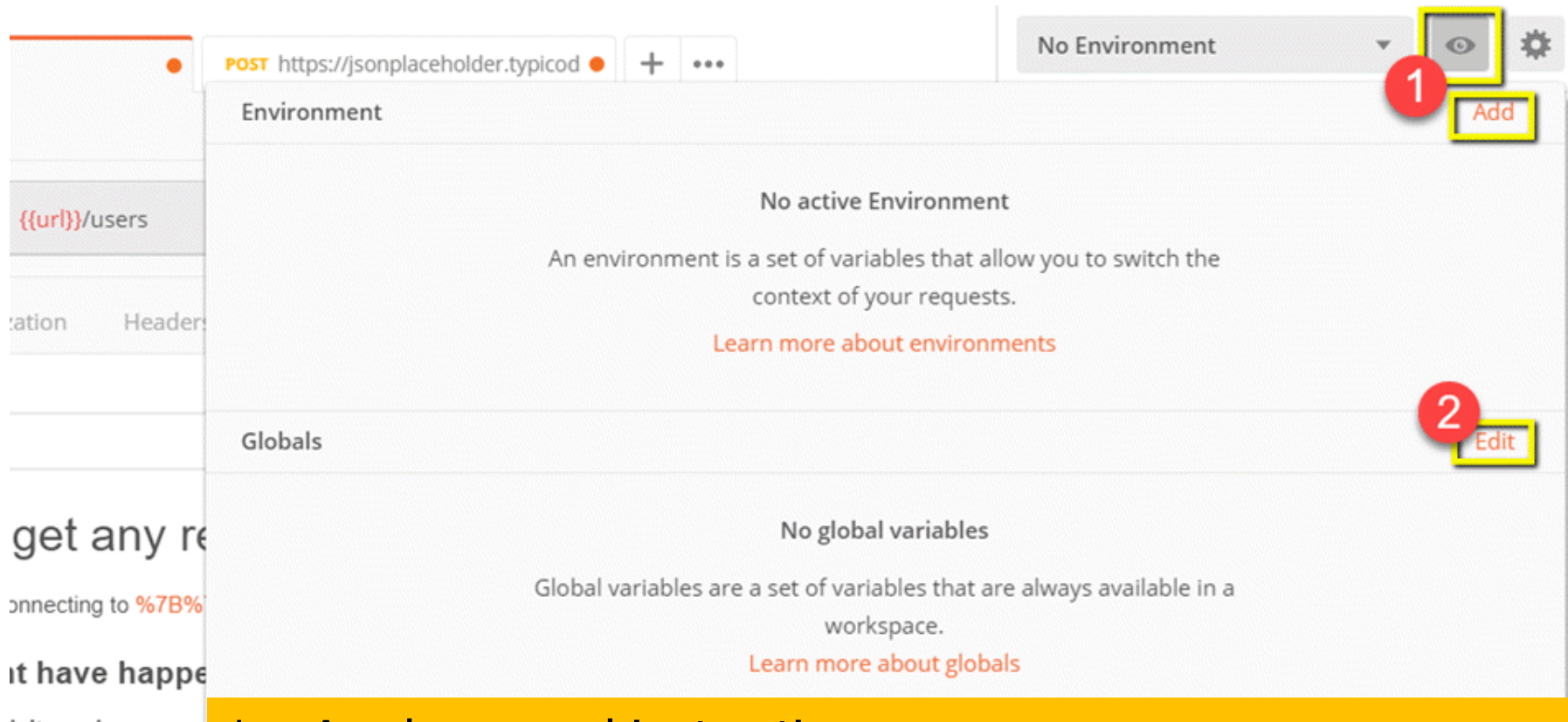
As chavetas duplas em qualquer local simbolizam a criação de um parâmetro.



# Exemplo Pedido POST

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

## ■ Parameterização de Dados POSTMAN (Cont.)



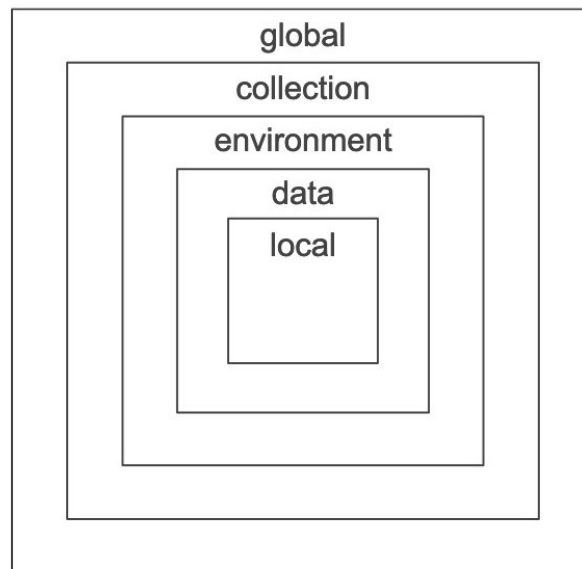
1. Aceder ao ambiente ativo
2. Definir as variáveis como globais no ambiente POSTMAN ou como parte de um ambiente





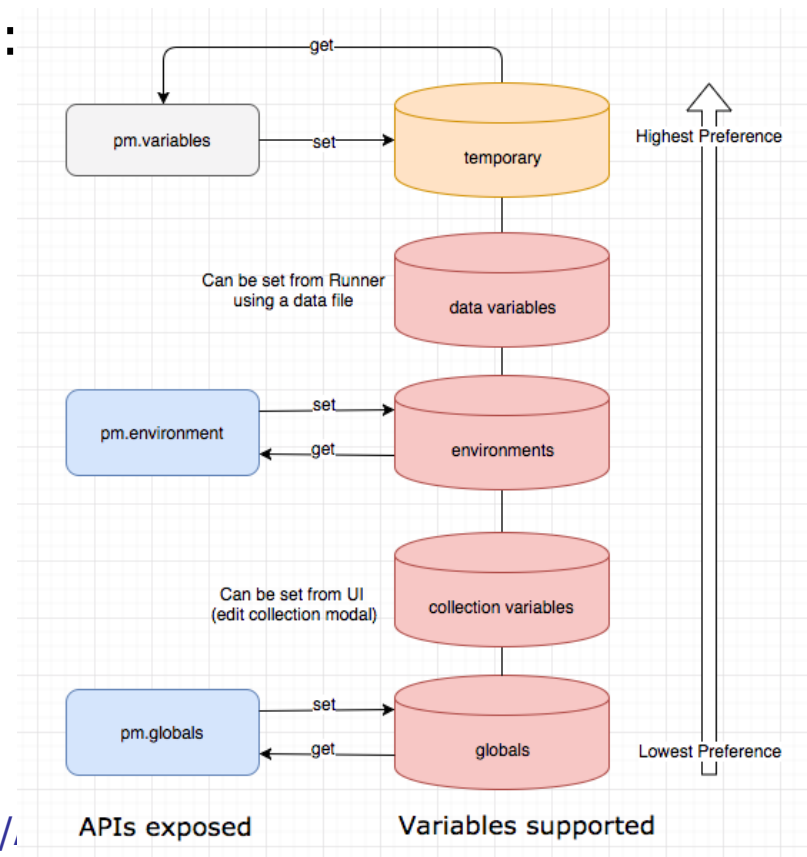
# + Sobre Variáveis no Postman

- Uma variável é uma representação simbólica dos dados sem ter que os introduzir novamente, o que é muito útil quando os mesmos valores aparecem em múltiplos lugares ou pedidos.
- As variáveis tornam os pedidos mais flexíveis e legíveis abstraindo os detalhes.
- As variáveis podem ter vários âmbitos (scopes):
  - Globais: acessíveis entre diferentes coleções
  - Coleção: acessíveis à coleção e independentes do ambiente
  - Ambiente: acessíveis ao ambiente (ex. Local ou em produção).
  - Locais: ao script / data: que vem de fontes externas de dados



Alexandre Foi

: Apoio a AD/





# + Sobre Variáveis no Postman

- Definição e uso de Variáveis nos Scripts

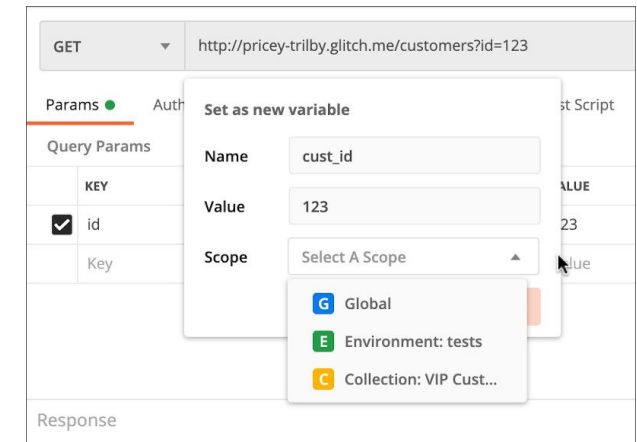
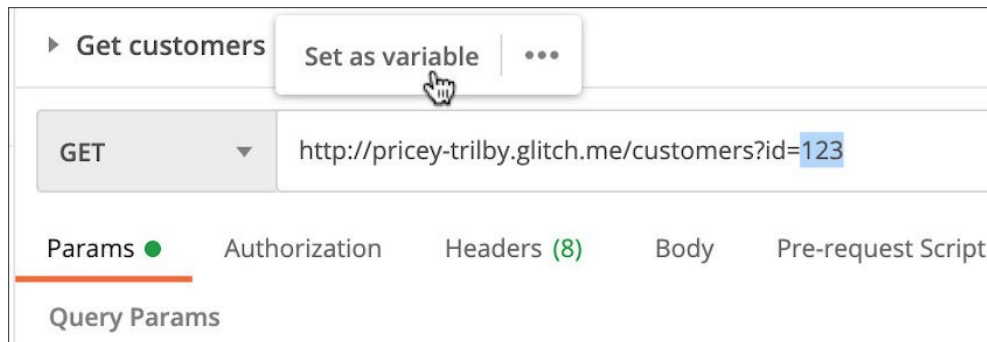
```
pm.globals.set("variable_key", "variable_value");  
pm.collectionVariables.set("variable_key", "variable_value");  
pm.environment.set("variable_key", "variable_value");  
pm.variables.set("variable_key", "variable_value")
```

```
//access a variable at any scope including local  
pm.variables.get("variable_key");  
//access a global variable  
pm.globals.get("variable_key");  
//access a collection variable  
pm.collectionVariables.get("variable_key");  
//access an environment variable  
pm.environment.get("variable_key");
```

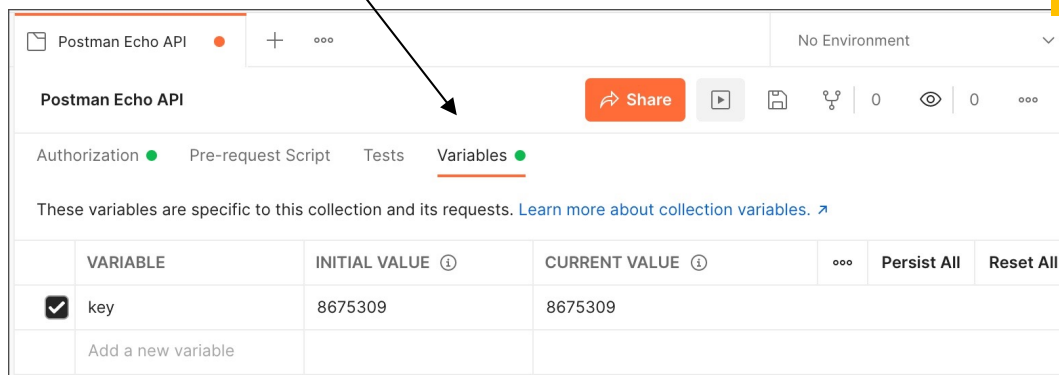


# + Sobre Variáveis no Postman

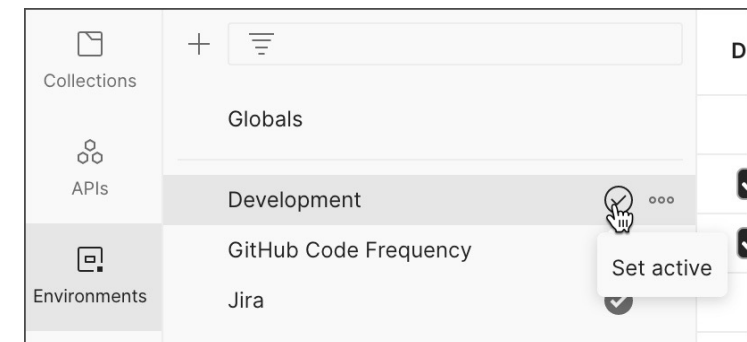
- Definição de Variáveis Visualmente no Request Builder, na Coleção ou no ambiente



Na coleção



No ambiente ativando o ambiente





# Exemplo Pedido POST

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

- Parameterização de Dados POSTMAN (Cont.)
  - Defina uma nova variável de âmbito da coleção

EnsaioTP Watch 0

Authorization Pre-request Script Tests **Variables**

These variables are specific to this collection and its requests. [Learn more about collection variables.](#)

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ
<input checked="" type="checkbox"/>	urlcoll	https://jsonplaceholder.typicode.com/users	https://jsonplaceholder.typicode.com/users
	Add a new variable		

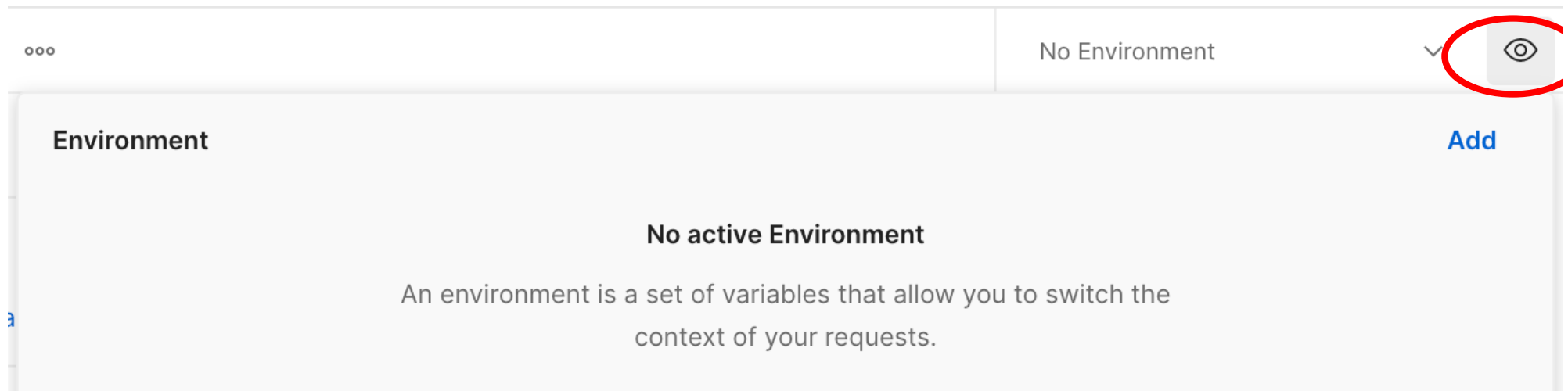
1. Aceder à coleção
2. Definir uma nova variável **urlcoll** como parte do âmbito da coleção em "Variables" e Gravar para esta ficar ativa
3. Ensaiar alterando no pedido POST a variável para **{{urlcoll}}**



# Exemplo Pedido POST

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

- Parameterização de Dados POSTMAN (Cont.)
  - Defina uma nova variável num novo ambiente **testes**



1. Adicionar um novo ambiente **testes**
2. Definir uma nova variável **urltestes** e Gravar para esta ficar ativa
3. Ensaiar alterando no pedido POST a variável para **{{urltestes}}**



# Scripts no POSTMAN

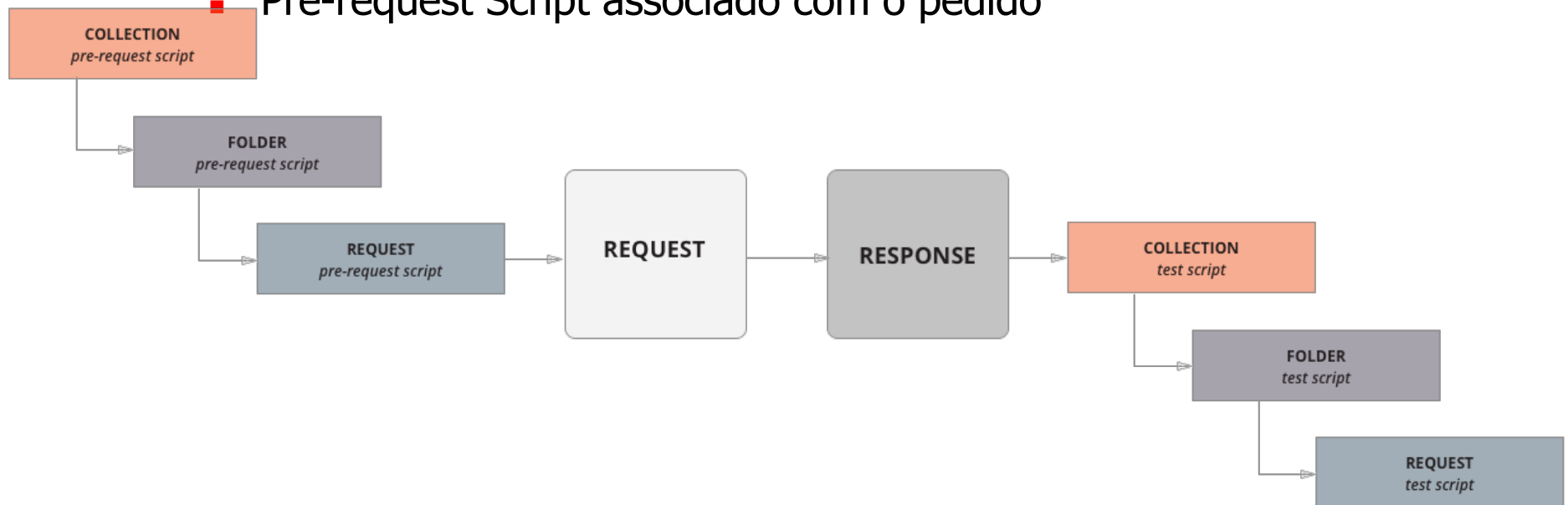
- O Postman contém um ambiente run-time baseado em Node.js que permite adicionar comportamento dinâmico aos pedidos e às coleções de testes
  - Permite criar pedidos com parâmetros dinâmicos
  - Permite passar dados entre pedidos
- É possível adicionar código JavaScript para ser executado durante dois eventos
  - Antes do pedido ser enviado, como um pre-request script no tabulador **Pre-request Script**
  - Após a receção da resposta, como um test-script no tabulador **Tests**

KEY	VALUE	DESCRIPTION
Key	Value	Description



# Scripts no POSTMAN

- É possível adicionar Scripts à Coleção, às Pastas e aos Pedidos
- A ordem de execução antes do envio de um pedido é:
  - Pre-request Script associado com a coleção
  - Pre-request Script associado com uma pasta
  - Pre-request Script associado com o pedido

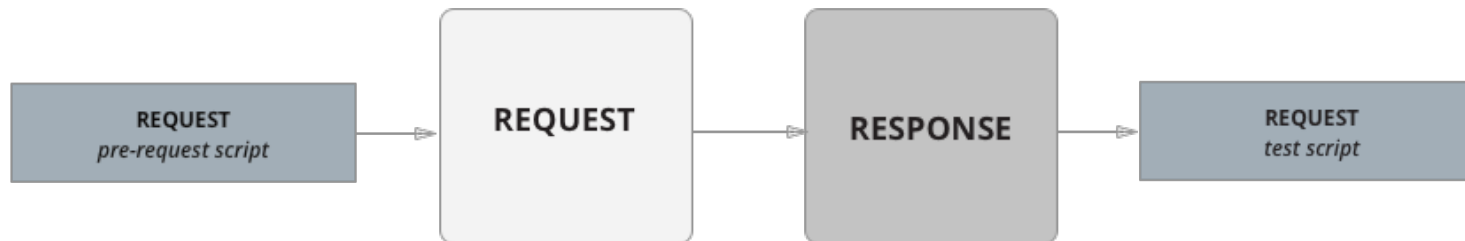




# Exemplo Pedido POST

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

- Criação de Testes à API
  - Obedece à Ordem na Figura
    - 1.º Escreve-se um script de pré-teste ou pre-request script (opcional)
    - 2.º Escreve-se um script de teste



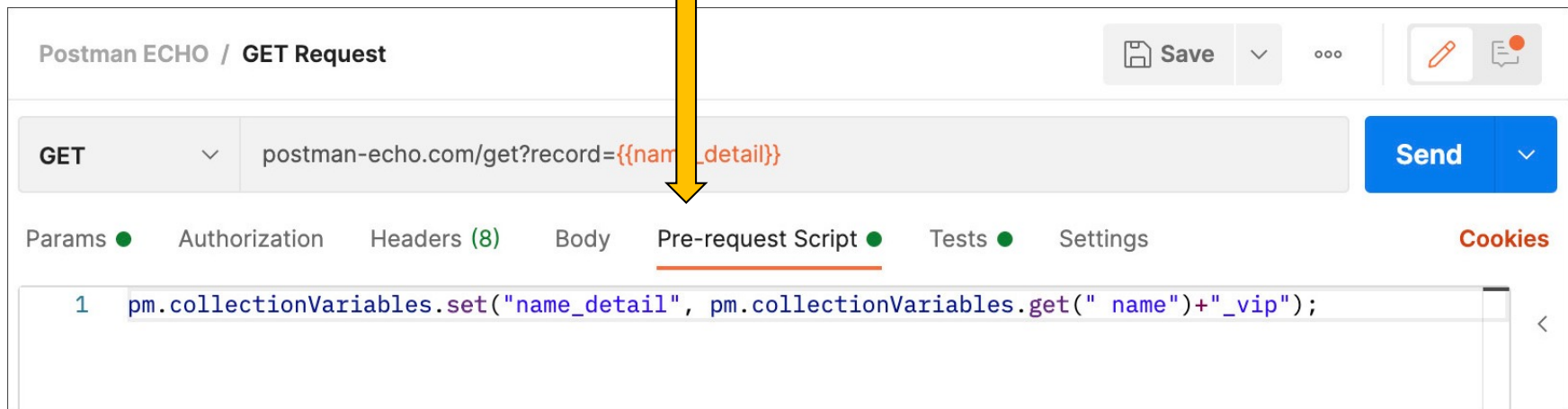




# Exemplo Pedido POST

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

- Criação de Testes à API
  - Obedece à Ordem na Figura
    - 1.º Escreve-se um script de pré-teste (opcional)





# Exemplo Pedido POST

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

## ■ Criação de Testes à API

- Obedece à Ordem na Figura
  - 2.º Escreve-se um script de teste

Watching APIs / GET Auto-Complete

GET postman-echo.com/get

Params Authorization Headers (8) Body Pre-request Script **Tests** Settings Cookies

```
1 pm.test('Success',function() {
2   pm.response
3 })
4
5
```

Test scripts are written in JavaScript, and are run after the response is received.  
[Learn more about tests scripts](#)

SNIPPETS

- Get an environment variable
- Get a global variable
- Get a variable
- Set an environment variable
- Set a global variable
- Clear an environment variable
- Clear a global variable
- Send a request
- Status code: Code is 200
- Response body: Contains string



# Exemplo Pedido POST

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

- Criação de um 1.º Pré-Teste Básico
  - Defina uma variável global **url** com o valor do url do nosso exemplo e uma variável de coleção **id** = 1

The screenshot shows a REST client interface. At the top, the method is set to 'GET' and the URL is a placeholder: `{{url}}/users/{{id}}`. A 'Send' button is on the right. Below the URL bar, tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings' are visible. The 'Pre-request Script' tab is active, showing a JavaScript script:

```
1 pm.globals.set("url", "https://jsonplaceholder.typicode.com");
2 pm.collectionVariables.set("id", "1");
```

To the right of the script editor, there is explanatory text: 'Pre-request scripts are written in JavaScript, and are run before the request is sent. Learn more about [pre-request scripts](#)'. Below this, a 'Snippets' section lists several reusable code blocks: 'Get a global variable', 'Get a variable', 'Get a collection variable', 'Set an environment variable', and 'Set a global variable'.



# Exemplo Pedido POST

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

## ■ Criação de um 1.º Teste Básico

The screenshot shows the Postman interface for a GET request to `https://jsonplaceholder.typicode.com/users/{{id}}`. The **Tests** tab is active, displaying a JavaScript test script:

```
1 pm.test("Status code is 200", function () {  
2   pm.response.to.have.status(200);  
3 });
```

Below the script, a list of test snippets is visible:

- Clear a global variable
- Send a request
- Status code: Code is 200
- Response body: Contains string
- Response body: JSON value check
- Response body: Is equal to a string
- Response headers: Content-Type header check
- Response time is less than 200ms

The **Test Results (1/1)** tab at the bottom shows a single test result:

Test Results (1/1)
<b>PASS</b> Status code is 200

Additional details at the bottom of the interface include: Status: 200 OK, Time: 224 ms, Size: 1.58 KB, and a **Save Response** button.



# Exemplo Pedido POST

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

- Criação de um 2.º Teste Básico verificação do Body JSON da Resposta

GET `{{url}}/users/{{id}}` Send Save

Params Authorization Headers (7) Body Pre-request Script **Tests** Settings Cookies Code

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Verificar se os dados são do user com id = 1", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.id).to.eql(1);
7 });
```

Test scripts are written in JavaScript, and are run after the response is received.  
[Learn more about tests scripts](#)

SNIPPETS  
[Clear a global variable](#)

[Send a request](#)

Status code: Code is 200

Response body: Contains string

Response body: JSON value check

Response body: Is equal to a string

Response headers: Content-Type header check

Response time is less than 200ms

Body Cookies (1) Headers (26) **Test Results (2/2)** Status: 200 OK Time: 64 ms Size: 1.58 KB Save Response

All Passed Skipped Failed

**PASS** Status code is 200

**PASS** Verificar se os dados são do user com id = 1



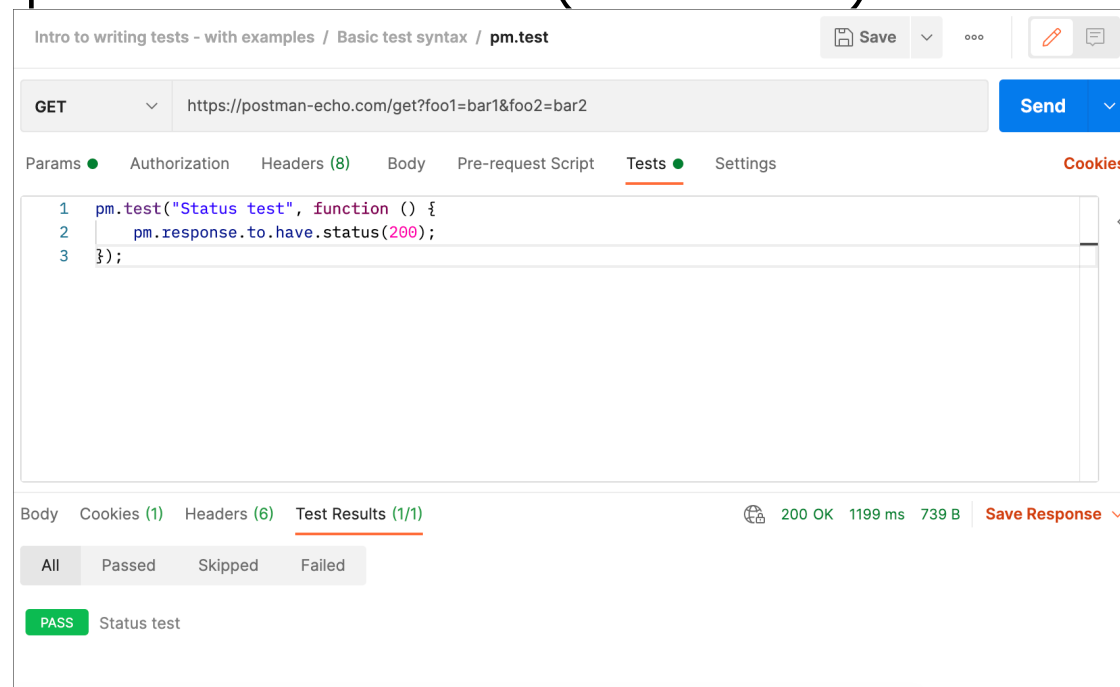
# + Sobre a Escrita de Testes

## ■ API **pm**

- A maioria da funcionalidade da Postman JavaScript API está disponível em **pm.\***, que fornece acesso aos dados dos pedidos e das respostas e às variáveis.

## ■ Métodos/funções comuns do objecto **pm**

- **pm.test** – tem como parâmetros de entrada o nome do teste e uma função que retorna um booleano (true ou false)



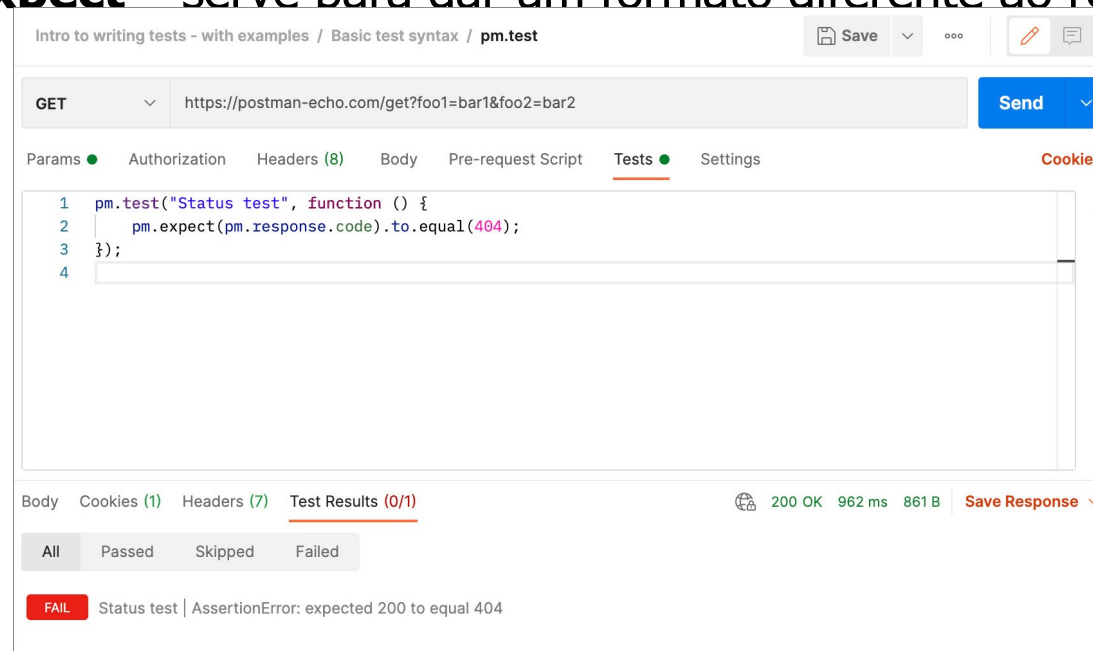


# + Sobre a Escrita de Testes

- Métodos/funções comuns do objecto **pm** (cont.)
  - **pm.response** – serve para validar os dados retornados numa resposta a um request. No exemplo se o Código de estado for 200 OK o teste passa senão o teste falha.

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

- **pm.expect** – serve para dar um formato diferente ao resultado do teste





## + Sobre a Escrita de Testes

- Métodos/funções comuns do objecto **pm** (cont.)
  - **pm.response** – permite também obter o body da resposta

```
pm.test("Person is Jane", function () {  
    const responseJson = pm.response.json();  
    pm.expect(responseJson.name).to.eql("Jane");  
    pm.expect(responseJson.age).to.eql(23);  
});
```

- **pm.environment** – serve para obter o valor de uma variável de ambiente.

```
pm.test("environment to be production", function () {  
    pm.expect(pm.environment.get("env")).to.equal("production");  
});
```





## Exemplo Pedido POST

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

---

- Para aprender muito mais sobre a criação de Testes de APIs REST com o POSTMAN consultar a documentação
  - <https://learning.postman.com/docs/writing-scripts/test-scripts/>
- Também podemos encontrar vários exemplos de Testes em:
  - <https://learning.postman.com/docs/writing-scripts/script-references/test-examples/>



# Exemplo Pedido POST

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

- Depois de consultar o segundo link final
  - Crie um teste que verifica se o Header Content-Type é do tipo 'application/json'

The screenshot shows a REST client interface with a POST request to `{{url}}/users/5`. The 'Tests' tab is active, displaying a JavaScript test script. The script includes three tests: checking the status code is 200, verifying the response is JSON, and checking the Content-Type header is 'application/json; charset=utf-8'. On the right, a list of snippets is visible, including 'Response body: JSON value check' and 'Response headers: Content-Type header check'.

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("Your test name", function () {
6   var jsonData = pm.response.json();
7   pm.expect(jsonData.id).to.eql(5);
8 });
9
10 pm.test("Content-Type header is application/json", () => {
11   pm.expect(pm.response.headers.get('Content-Type')).to.eql(
12     'application/json; charset=utf-8');
13 });
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about [tests scripts](#)

**Snippets**

- response body. Contains string
- Response body: JSON value check
- Response body: Is equal to a string
- Response headers: Content-Type header check
- Response time is less than 200ms
- Status code: Successful POST request
- Status code: Code name has string
- Response body: Convert XML body to a JSON Object



# Collection Runner

## *Testes Manuais e Automáticos*

Widget Collection

Widget Collection

+ ...

No Environment

Widget Collection - Run results

Run Again

Automate Run

+ New Run

Run on Today, 10:36:41 · [View all runs](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	992ms	4	137 ms

All Tests

Passed (3)

Failed (1)

Skipped (0)

[View Summary](#)

Iteration 1

1

GET GET widget https://postman-echo.com/get / GET widget 200 OK 176 ms 835 B

Pass Status code is 200

GET GET widgets https://postman-echo.com/get / GET widgets 200 OK 123 ms 839 B

Fail Status code name has string | AssertionError: expected response to have status reason 'Created' but got 'OK'

POST POST widget https://postman-echo.com/post / POST widget 200 OK 124 ms 887 B

Pass Response time is less than 200ms

DELETE DELETE widget https://postman-echo.com/delete / DELETE widget 200 OK 124 ms 885 B

Pass Status code is 200



# Collection Runner

## Testes Manuais - Exemplo

RUN ORDER

Deselect All | Select All | Reset

Choose how to run your collection

- ☒ **GET** Pedido GET
- ☒ **POST** Pedido POST

- ☒ **Run manually**  
Run this collection in the Collection Runner.
- ☐ **Schedule runs**  
Periodically run collection at a specified time on the Postman Cloud.
- ☐ **Automate runs via CLI**  
Configure CLI command to run on your build pipeline.

Run configuration

Iterations

100

Delay

0

ms

Data

Select File

Advanced settings

- ☐ Save responses ⓘ
- ☒ Keep variable values ⓘ
- ☐ Run collection without using stored cookies
- ☒ Save cookies after collection run ⓘ



# Collection Runner

## *Testes Manuais - Exemplo*

---

- **Iterations** - The number of iterations for your collection run. You can also run collections multiple times with different data sets to [build workflows](#).
- **Delay** - An interval delay in milliseconds between each request.
- **Data** - A [data file](#) for the collection run.
- **Advanced settings**
  - **Save responses** - Save response headers and bodies to the log to review them later. For large collection runs, this setting can affect performance.
  - **Keep variable values** - Persist the variables used in the run, so that any variables updated by the run will remain changed after it completes. If you don't persist variables, changes aren't saved after the run completes. *Note that persisting variables in the collection run will update the current value only.*
  - **Run collection without using stored cookies** - If your requests use cookies, you can optionally deactivate them for a collection run.
  - **Save cookies after collection run** - Save the cookies used in this session to the cookie manager. Any values changed by requests during the run will remain after it completes.



# Collection Runner

## Testes Manuais - Exemplo

### Visualização detalhada dos Pedidos e Respostas

Aula AD 9 dezembro - Run results

Run Again

Automate Run

New Run

Export Results

Run on Today, 17:19:53 · [View all runs](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	Ambiente Produção	10	4s 256ms	40	151 ms

All Tests Passed (30) Failed (10) Skipped (0)

[View Summary](#)

Iteration 1

**GET** Pedido GET {{urlglobal}}/users/{{idcol}} / Pedido GET 200 OK 214 ms 1.546 KB

Pass Status code is 200

Request URL

Pass https://jsonplaceholder.typicode.com/users/10

**POST** Pedido POST {{urlglobal}}/users/{{idcol}} / Pedido POST 201 Created 241 ms 1.631 KB

Request Headers (6)

Pass Accept application/json

Pass Accept-Encoding gzip, deflate, br

Pass Connection keep-alive

Pass Host jsonplaceholder.typicode.com

Pass Postman-Token 75e81ff6-9961-4eb7-be9e-43cc28548e93

Pass User-Agent PostmanRuntime/7.29.2

Iteration 2

**GET** Pedido GET {{urlglobal}}/users/{{idcol}} / Pedido GET 200 OK 67 ms 1.552 KB

Pass Status code is 200

Request URL

Pass https://jsonplaceholder.typicode.com/users/10

**POST** Pedido POST {{urlglobal}}/users/{{idcol}} / Pedido POST 201 Created 219 ms 1.639 KB

Request Body

Pass

Response Headers

Response Body



# Collection Runner

## *Testes Automáticos*

RUN ORDER

Deselect All | Select All | Reset

- ☒ GET Pedido GET
- ☒ POST Pedido POST

### 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 Postman Cloud.
- ☐ Automate runs via CLI  
Configure CLI command to run on your build pipeline.

### Schedule configuration

Your collection will be automatically run on the Postman Cloud at the configured frequency. Learn more about [scheduling collection runs](#) ↗

#### Schedule name

Teste AD

#### Run Frequency ⓘ

High frequency helps catch issues quicker but increases [resource usage](#).

Week timer

Every day

▼

at

6:00 PM

▼

### Run configuration

#### Environment

Ambiente Testes

▼

#### Iterations (max 50)

10

#### Data file ⓘ

Only JSON and CSV files are accepted. Max 1 MB



**FIM**