FACULDADE ENGENHARIA

Departamento de Informática

Creating System Tests for a REST API with Postman

FACULDADE ENGENHARIA

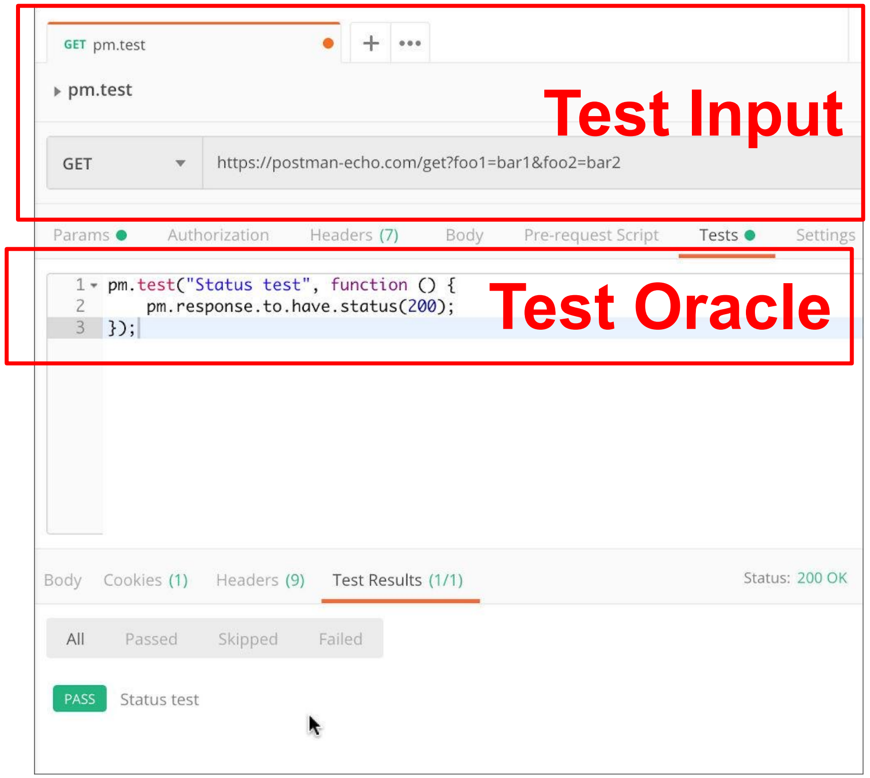
Departamento de Informática

Postman

- Testing and development framework for systems with a REST API.
  - A system interface with **endpoints** we can interact with.
  - At an endpoint, we can send HTTPS request to:
    - **GET** information that you are interested in.
    - **DELETE** the information stored.
    - **PUT** information into what is stored (ex: create a new entry)
    - **POST** information (ex: update an existing record)
- Create requests and test cases using Postman.

FACULDADE ENGENHARIA
Departamento de Informática

Writing Tests in Postman



```
https://postman-echo.com/get?foo1=bar1&foo2=bar2
```

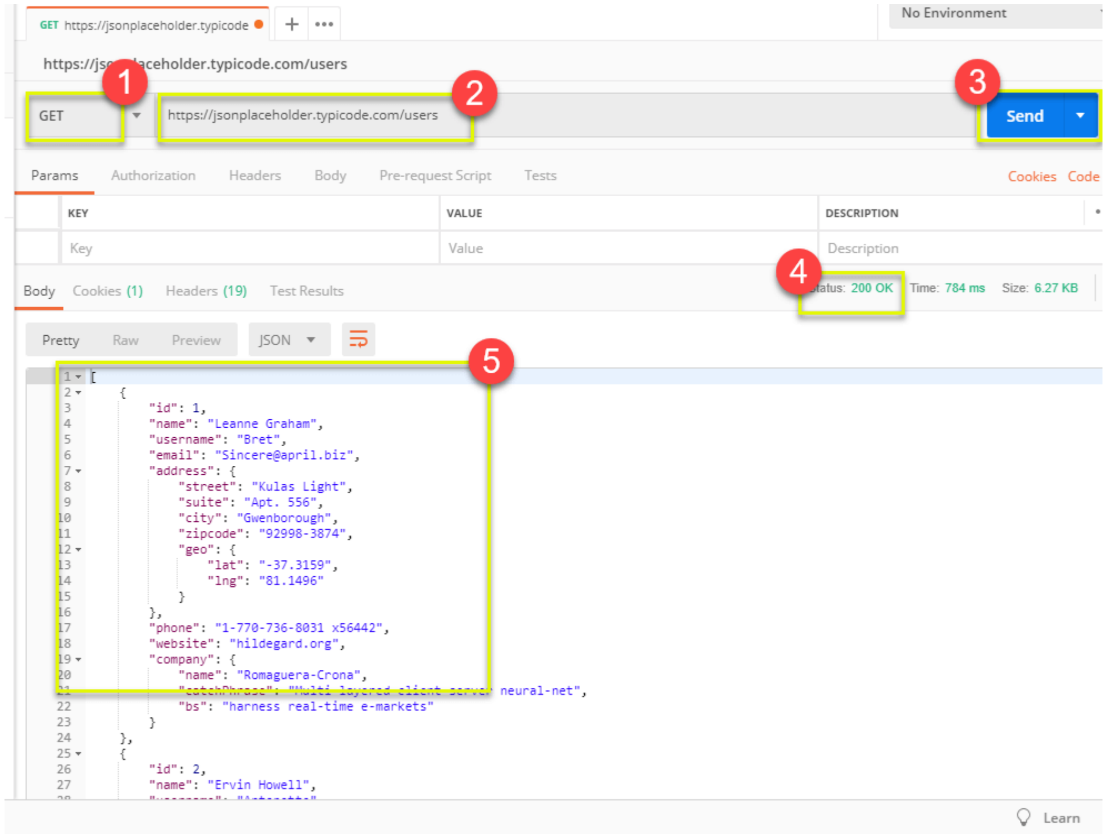
```
pm.test("Status test", function () {
    pm.response.to.have.status(200);
});
```

- Each tab is a request.
- The request is the **test input**.
  - (GET/POST/PUT/DELETE) to an endpoint.
  - Can specify body, header, authorization, etc. for the request.
- Tests tab allows creation of **test oracles**.
  - Write small JavaScript methods to check correctness of output.

Departamento de Informática

Input - GET

1. Select GET as the request type.
2. Set the endpoint URL.
3. Click “Send”
4. The response status is indicated.
5. The body contains the returned information.



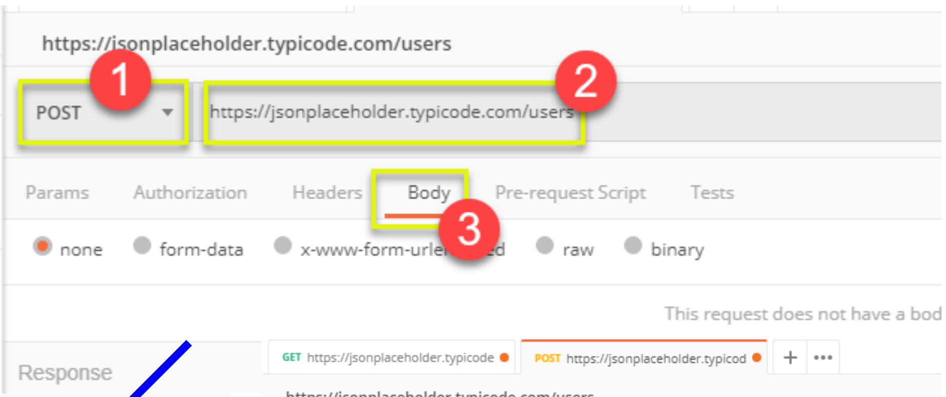
```
[
    {
        "id": 1,
        "name": "Leanne Graham",
        "username": "Bret",
        "email": "Sincere@april.biz",
        "address": {
            "street": "Kulas Light",
            "suite": "Apt. 556",
            "city": "Gwenborough",
            "zipcode": "92998-3874",
            "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"
        }
    },
    {
        "id": 2,
        "name": "Ervin Howell",
```

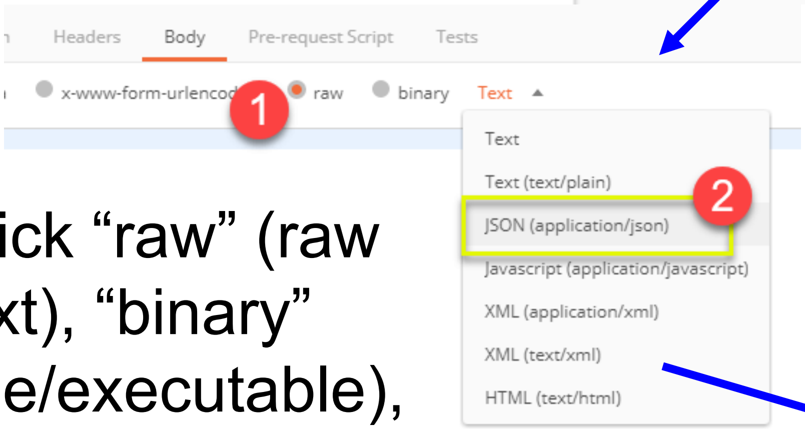
https://www.guru99.com/postman-tutorial.html

Input - POST

Departamento de Informática

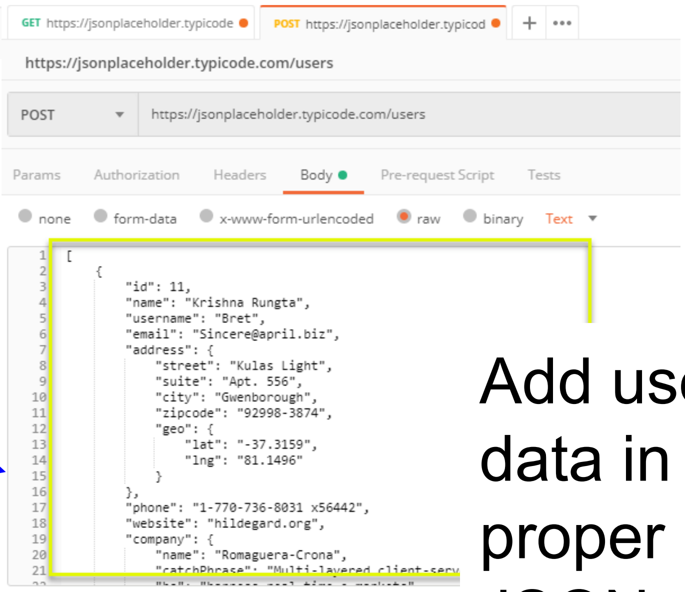
1. Set request to POST.
2. Set the endpoint URL.
3. Select the “Body” tab.


https://jsonplaceholder.typicode.com/users
1
POST
2
https://jsonplaceholder.typicode.com/users
Params
Authorization
Headers
Body
3
Pre-request Script
Tests
none
form-data
x-www-form-urlencoded
raw
binary
This request does not have a bod
Response



1. Click “raw” (raw text), “binary” (file/executable), etc.
2. Select data format (JSON, XML, etc.)

https://www.guru99.com/postman-tutorial.html


GET https://jsonplaceholder.typicode
POST https://jsonplaceholder.typicod
https://jsonplaceholder.typicode.com/users
POST
https://jsonplaceholder.typicode.com/users
Params
Authorization
Headers
Body
Pre-request Script
Tests
none
form-data
x-www-form-urlencoded
raw
binary
Text

```
[
    {
        "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-serv
```

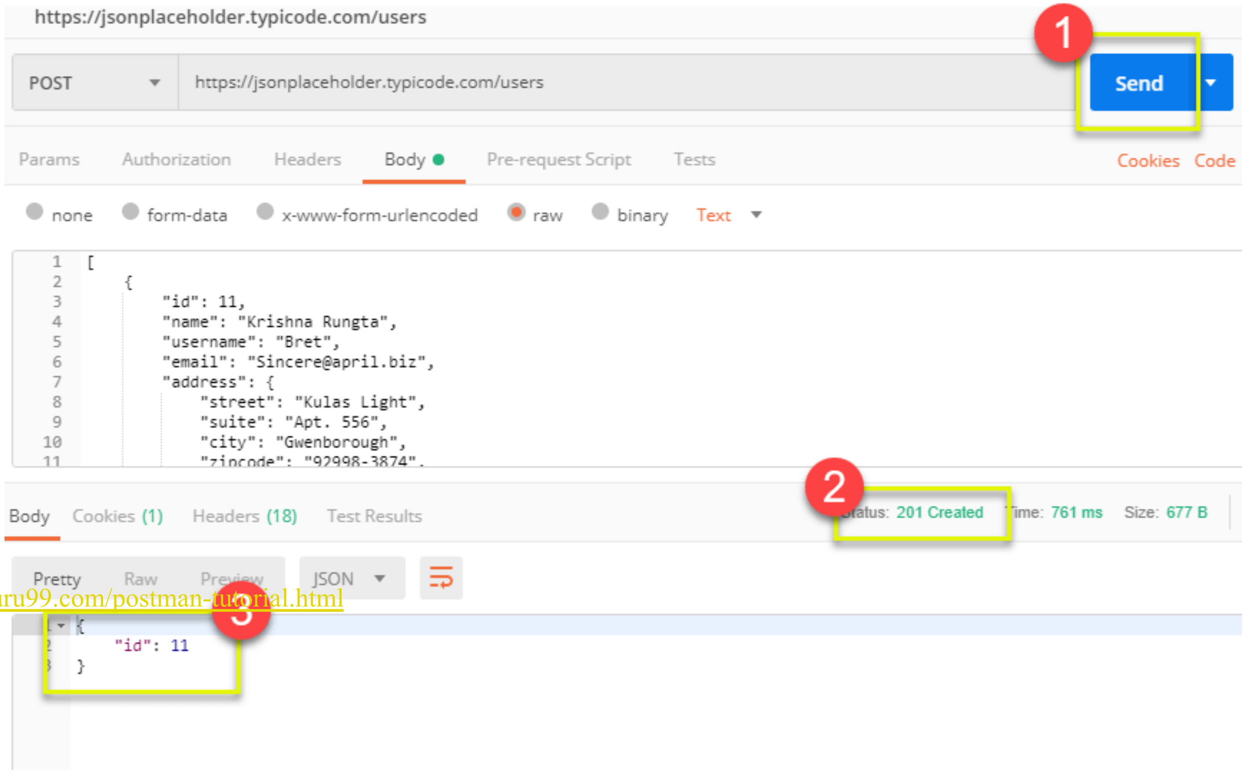
Add user data in proper JSON format.

FACULDADE ENGENHARIA

Departamento de Informática

Output - POST

1. Click Send to send request.
2. Response status is indicated (201, data created)
3. Body indicates record “11” was created.



https://jsonplaceholder.typicode.com/users

POST https://jsonplaceholder.typicode.com/users Send

Params Authorization Headers Body ● Pre-request Script Tests Cookies Code

none form-data x-www-form-urlencoded ● raw binary Text

```
[
    {
        "id": 11,
        "name": "Krishna Rungta",
        "username": "Bret",
        "email": "Sincere@april.biz",
        "address": {
            "street": "Kulas Light",
            "suite": "Apt. 556",
            "city": "Gwenborough",
            "zipcode": "92998-3874",
```

Body Cookies (1) Headers (18) Test Results Status: 201 Created Time: 761 ms Size: 677 B

Pretty Raw Preview JSON

```
{
    "id": 11
}
```

https://www.guru99.com/postman-tutorial.html

Departamento de
Informática

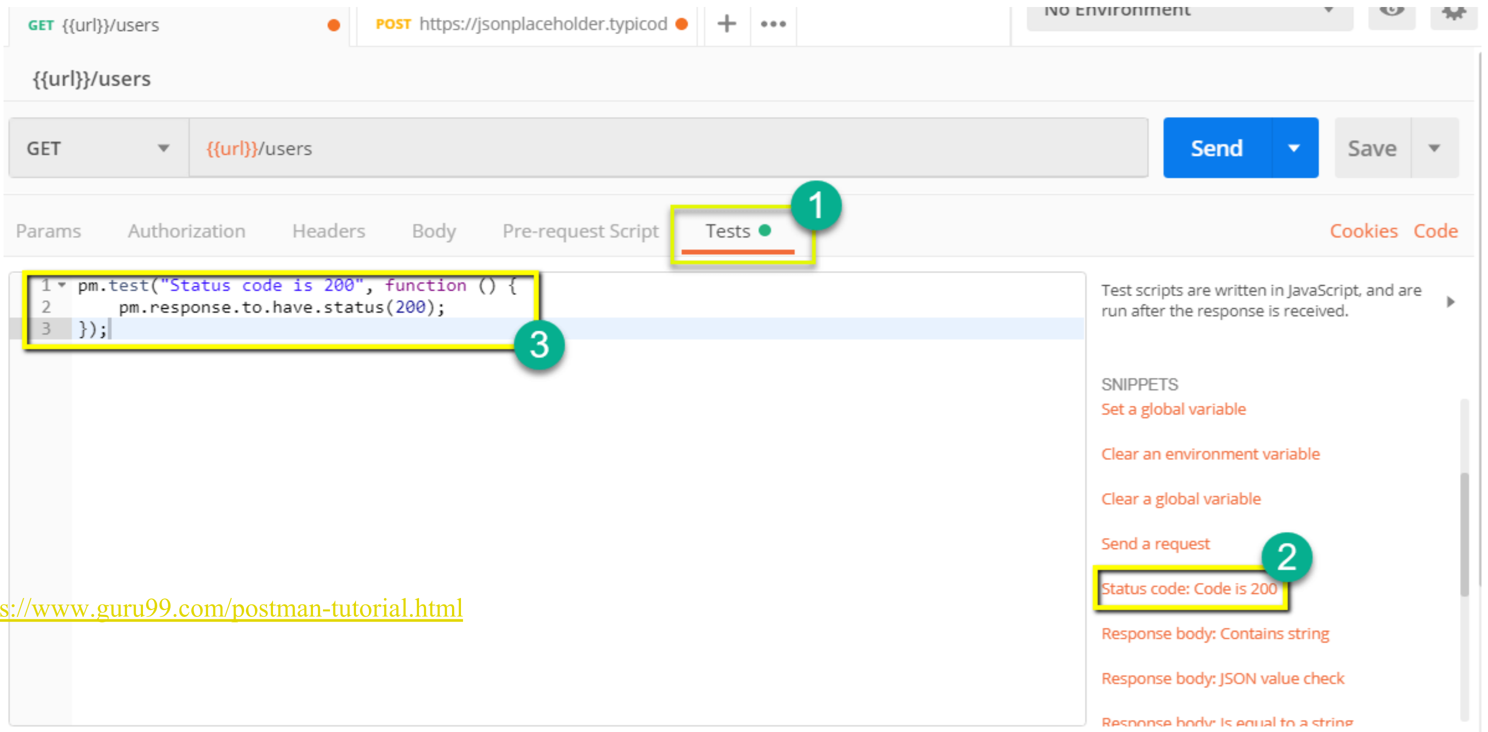
Creating Test Oracles

- Tests tab allows creation of JavaScript blocks used to verify results.
  - These are “test oracles”.
  - Embed expectations on results and code to compare expected and actual values.
- pm.test library gives variety of commands to make assertions on output.
  - https://learning.postman.com/docs/writing-scripts/script-references/test-examples/ (many example scripts!)

Departamento de
Informática

Oracle Example - Status Check

1. Create test in “tests tab”
2. Snippets offer pre-built test oracles.
3. Example - “status code must be 200”



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

Test scripts are written in JavaScript, and are run after the response is received.

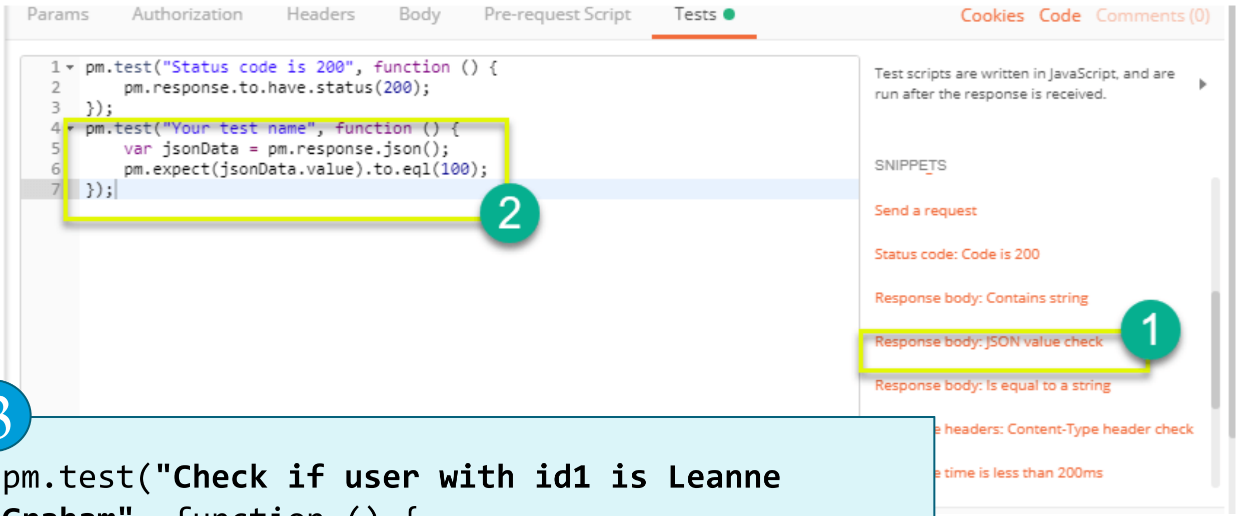
SNIPPETS
Set a global variable
Clear an environment variable
Clear a global variable
Send a request
Status code: Code is 200
Response body: Contains string
Response body: JSON value check
Response body: Is equal to a string

https://www.guru99.com/postman-tutorial.html

FACULDADE ENGENHARIA
Departamento de Informática

Oracle Example - Expected Value

1. Choose snippet "JSON value check"
2. This inserts generic test body.
3. Change **test name**, **variable to check** (name of the first user), **value to check** (check for name "Leanne Graham").



```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Your test name", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.value).to.eql(100);
});
```

Test scripts are written in JavaScript, and are run after the response is received.

SNIPPETS

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

3

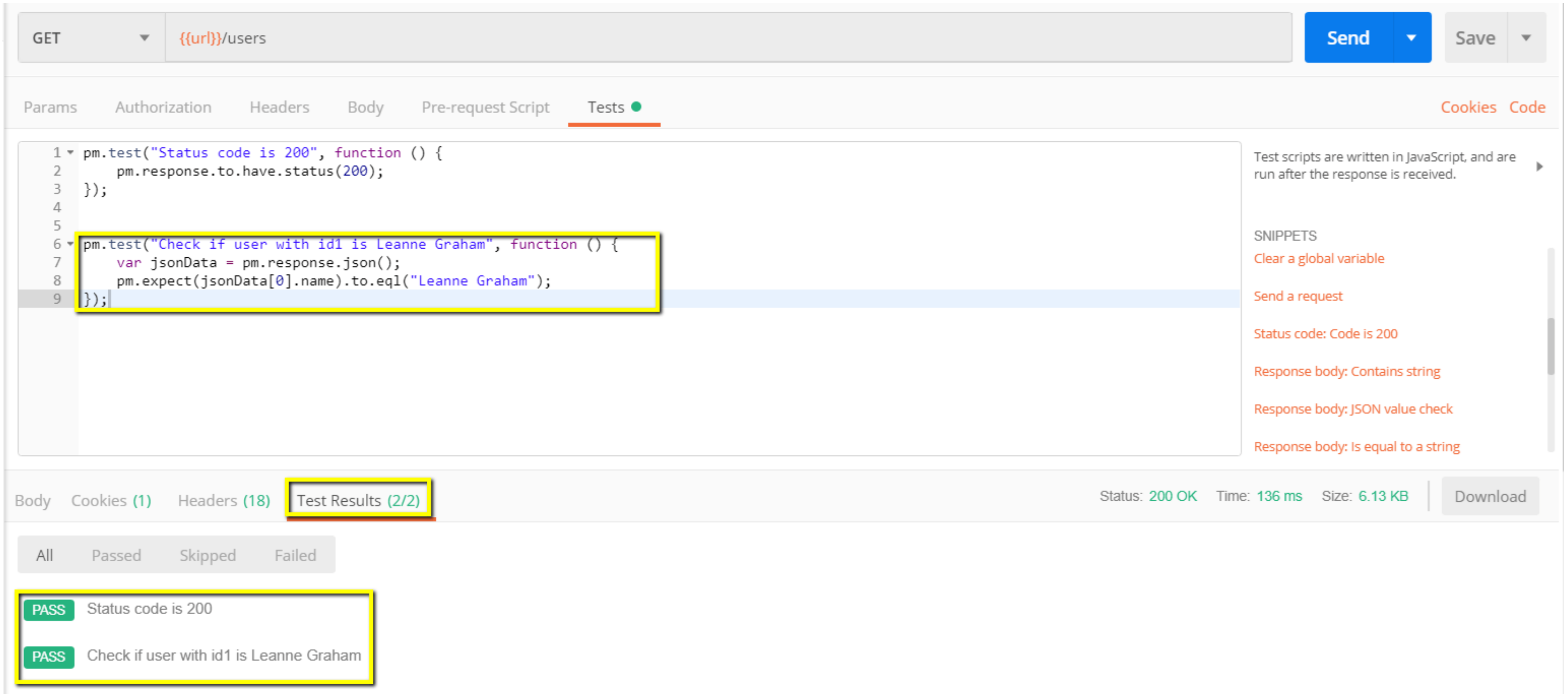
```
pm.test("Check if user with id1 is Leanne
Graham", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData[0].name).to.eql("Leanne
Graham");
});
```

https://www.gur

Departamento de Informática

Test Execution Results

Both tests should pass. Status and test names indicated in GUI.


GET {{url}}/users

Params Authorization Headers Body Pre-request Script Tests

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


pm.test("Check if user with id1 is Leanne Graham", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData[0].name).to.eql("Leanne Graham");
});
```

Test scripts are written in JavaScript, and are run after the response is received.

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

Body Cookies (1) Headers (18) Test Results (2/2)   Status: 200 OK Time: 136 ms Size: 6.13 KB

All Passed Skipped Failed

PASS Status code is 200

PASS Check if user with id1 is Leanne Graham

Departamento de
Informática