

Testarea pe API

Testarea pe API

Objective Intalnire 10

- Sa intelegem ce este un request si la ce ne foloseste
- Sa stim ce este un endpoint si care este rolul lui
- Sa reusim sa trimitem un request de GET si unul de POST
- Sa intelegem cum putem scrie corpul unui request de POST
- Sa parametrizam un request

Optional:

- Sa scriem teste direct in postman
- Sa trimitem requesturi de API direct din terminal prin Newman

Ce este un request?

Un request este o serie de actiuni pe care le putem face pentru a putea verifica o anumita functionalitate.

Componentele unui request sunt urmatoarele:

- Nume
- Metoda HTTP - o componenta care defineste ce fel de actiune poate requestul sa faca
- Endpoint - un loc unde solicitarea ajunge de la browser si pleaca catre server
- Headers - pentru autentificare, daca este cazul
- Body - corpul requestului, doar pentru request-urile de POST, PUT si PATCH

Request-uri de GET

GET este o metoda de HTTP prin intermediul careia putem solicita extragerea de informatii din baza de date.

- Pentru a trimite un request de GET, click pe butonul **New** din partea din stanga sus a postmanului si selecteaza HTTP Request din lista
- In fereastra care se deschide, selecteaza GET in dropdown-ul din stanga
- Unde scrie Enter Request URL scrie `https://opensource-demo.orangehrmlive.com//api/v1/user`, care este un endpoint
- In tab-ul de headers scrie sub key authorization, si sub value token-ul care a fost generat folosind metoda de autentificare de **aici**
- Apasa pe butonul de Send
- Rezultatele se vor gasi in partea de jos a request-ului

Pentru teste am folosit [documentatia de API](#)

Request-uri de POST

POST este o metoda de HTTP prin intermediul careia putem solicita scrierea de informatii in baza de date.

- Pentru a trimite un request de POST, click pe butonul **New** din partea din stanga sus a postmanului si selecteaza HTTP Request din lista
- In fereastra care se deschide, selecteaza POST in dropdown-ul din stanga
- Unde scrie Enter Request URL scrie `https://opensource-demo.orangehrmlive.com/api/v1/employee/1`, care este un endpoint
- In tab-ul de headers scrie sub key authorization, si sub value token-ul care a fost generat folosind metoda de autentificare de **aici**
- In tab-ul de body aflat in dreapta celui de headers scrie `{ "firstName": "Anton", "lastName": "Martinescu", "code": "124", "userName": "", "loginPassword": "passtest" }`
- Apasa pe butonul de Send
- Rezultatele se vor gasi in partea de jos a request-ului

```
{  
  "firstName": "Anton",  
  "lastName": "Martinescu",  
  "code": "124",  
  "userName": "",  
  "loginPassword": "passtest"}  
}
```

Exemplu de request de POST

The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** `https://opensource-demo.orangehrmlive.com/api/v1/employee/1`
- Headers:** 10 headers are listed, with 9 hidden. The visible header is:

KEY	VALUE	DESCRIPTION
Authorization	Bearer eda413e7d4415f716cb41ebc75766ac32d86779e	
Key	Value	Description
- Body:** The response is shown in JSON format:

```
{  "success": "Successfully Saved",  "id": "65"}
```
- Status:** 200 OK, Time: 198 ms, Size: 136 KB

Request-uri de PUT

PUT este o metoda de HTTP prin intermediul careia putem solicita modificarea de informatii in baza de date prin rescrierea intregului obiect.

- Pentru a trimite un request de PUT, click pe butonul **New** din partea din stanga sus a postmanului si selecteaza HTTP Request din lista
- In fereastra care se deschide, selecteaza PUT in dropdown-ul din stanga
- Unde scrie Enter Request URL scrie `https://opensource-demo.orangehrmlive.com/api/v1/employee/1/dependent`, care este un endpoint
- In tab-ul de headers scrie sub key authorization, si sub value token-ul care a fost generat folosind metoda de autentificare de **aici**
- In tab-ul de body aflat in dreapta celui de headers scrie urmatorul cod json:
- Apasa pe butonul de Send
- Rezultatele se vor gasi in partea de jos a request-ului

```
{  
  "id":1,  
  "city":"Bucuresti",  
  "state":"Romania",  
  "workEmail":"test@gmail.com"  
}
```

Exemplu de request de PUT

The screenshot displays a REST client interface with a PUT request to the URL `https://opensource-demo.orangehrmlive.com/api/v1/employee/1/contact-detail`. The request body is a JSON object with the following fields: `"id": 1`, `"city": "Bucuresti"`, `"state": "Romania"`, and `"workEmail": "test@gmail.com"`. The response status is 200 OK, with a time of 186 ms and a size of 1.35 KB. The response body is a JSON object with the field `"success": "Successfully Updated"`.

Request:

```
PUT https://opensource-demo.orangehrmlive.com/api/v1/employee/1/contact-detail
```

Body (JSON):

```
{
  "id": 1,
  "city": "Bucuresti",
  "state": "Romania",
  "workEmail": "test@gmail.com"
}
```

Response:

```
{
  "success": "Successfully Updated"
}
```


Request-uri de DELETE

DELETE este o metoda de HTTP prin intermediul careia putem solicita stergerea de informatii din baza de date prin rescrierea intregului obiect.

- Pentru a trimite un request de DELETE, click pe butonul **New** din partea din stanga sus a postmanului si selecteaza HTTP Request din lista
- In fereastra care se deschide, selecteaza DELETE in dropdown-ul din stanga
- Unde scrie Enter Request URL scrie `https://opensource-demo.orangehrmlive.com/api/v1/employee/1/dependents`, care este un endpoint
- In tab-ul de headers scrie sub key authorization, si sub value token-ul care a fost generat folosind metoda de autentificare de **aici**
- In tab-ul de body aflat in dreapta celui de headers scrie urmatorul cod json:
- Apasa pe butonul de Send
- Rezultatele se vor gasi in partea de jos a request-ului

```
1 {  
2   "id":1,  
3   "sequenceNumber":1  
4 }
```

Exemplu de request de DELETE

The screenshot displays a REST client interface with a DELETE request configured. The URL is `https://opensource-demo.orangehrmlive.com/api/v1/employee/1/dependent`. The request body is a JSON object: `{ "id": 1, "sequenceNumber": 1 }`. The response status is 200 OK, and the response body is `{ "success": "Successfully Deleted" }`.

Request Configuration:

- Method: DELETE
- URL: `https://opensource-demo.orangehrmlive.com/api/v1/employee/1/dependent`
- Body:

```
{
  "id": 1,
  "sequenceNumber": 1
}
```

Response Details:

- Status: 200 OK
- Time: 189 ms
- Size: 1.35 KB
- Body:

```
{
  "success": "Successfully Deleted"
}
```

Intrebari de interviu:

- Care este diferenta intre metodele POST si PUT?
- Ce este un endpoint?

Intrebari & Curiozitati?



Intalnirea 10

Testarea de API
- Studiu Optional -

Folosirea si instalarea aplicatiei Newman

O alta modalitate de a rula o colectie este print intermediul Newman, direct din terminalul de la calculator/laptop.

Principalele diferente intre Newman si Collection Runner sunt urmatoarele:

- Newman este un add-on for Postman. Vei avea nevoie sa il instalezi separat
- Newman foloseste command line in timp ce Collection Runner are un GUI.
- Newman poate fi folosit pentru [continuous integration](#).

Pentru a instala Newman, trebuie sa faci urмatorii pasi:

- Instaleaza nodejs folosind urmatorul [link](#):
- Deschide terminalul din calculatorul personal (pe windows intrati in modulul de cautare si scrieti cmd, pe mac intrati in Launchpad si scrieti terminal)
- In fereastra neagra care se deschide scrieti **npm install -g newman**
- Daca folositi mac OS puteti folosi comanda **brew install newman**

Rularea Colectiilor cu Newman

- In primul rand va trebui sa exportati colectiile din postman. In sectiunea Collections, dati click pe cele trei puncte din dreapta numelui colectiei. In fereastra cu optiuni care se deschide dati click pe **Export**
- Alege Export Collection as Collection v2.1 (Recommended) si apoi click pe Export.

Selecteaza locatia in care vrei sa salvezi si apasa pe Save. Este recomandat sa creati un folder specific pentru teste Postman.

Vom avea nevoie de asemenea de variabilele globale daca acestea au fost create. Asadar dati click pe iconita in forma de ochi din partea din dreapta sus a Postman-ului, si apasa pe Edit in dreptul variabilelor globale pe care le-ai creat. Apasa pe butonul Export, alege folderul pe care l-ai creat anterior si da save

- Mergi inapoi in terminal si muta-te in folderul in care ai salvat colectia si variabilele globale. Poti folosi comanda cd: `cd C:\Users\Desktop\Postman Tutorial`
- Poti rula colectia utilizand urmatoarea comanda:

```
newman run jsonplaceholder.postman_collection.json -e Testing.postman_globals.json
```

Exemplu de rulare

```
(base) C02ZL0CALVCF:~ gradulescu$ cd /Users/gradulescu/Desktop/Personal/Cursuri/ITF/"Manual Testing"/"Prezentari Finale"/"Testing Classes"/"Postman Testing"
(base) C02ZL0CALVCF:Postman Testing gradulescu$ newman run jsonplaceholder.postman_collection.json -e workspace.postman_globals.json
newman
```

jsonplaceholder

+ Create Users

POST https://jsonplaceholder.typicode.com/users [201 Created, 1.71KB, 410ms]

+ Get Users

GET https://jsonplaceholder.typicode.com/users [200 OK, 6.57KB, 72ms]

✓ Check the results with return status code 200

✓ Check if user with id 1 is Leanne Graham

+ Get request with parameters

GET https://jsonplaceholder.typicode.com/users [200 OK, 6.58KB, 60ms]

	executed	failed
iterations	1	0
requests	3	0
test-scripts	1	0
prerequest-scripts	0	0
assertions	2	0
total run duration: 611ms		
total data received: 11.59KB (approx)		
average response time: 180ms [min: 60ms, max: 410ms, s.d.: 162ms]		

```
(base) C02ZL0CALVCF:Postman Testing gradulescu$ █
```

Comenzi de baza in Newman pentru a rula o colectie

- **Rularea doar a unei colectii** - Aceasta optiune poate fi folosita atunci cand nu avem variabile globale sau alte dependinte: `newman run <collection name>`
 - **Rularea unei colectii cu variabile globale**. Optiunea `-e` vine de la environment iar `environment_name` reprezinta variabilele globale care au fost exportate: `newman run <collection name> -e <environment name>`
 - **Rularea unei colectii cu un anumit numar de iteratii (adica de cate ori vrem sa se ruleze colectia respectiva)**. `newman run <collection name> -n <no.of iterations>`
 - **Rularea unei colectii cu un fisier extern**: `newman run <collection name> --data <file name> -n <no.of iterations> -e <environment name>`
 - **Set delay time**. This is important as tests may fail if it is run without delay due to requests being started without the previous request completing processing on the endpoint server `newman run <collection name> -d <delay time>`
- [Newman tutorial](#)
- [Lista optiuni newman](#)

Note:

- Testele vor trebui adaptate in functie de request-ul pe care il faceti
- Orice neclarificari legate de API vor fi trimise pe adresa gabriela.radulescu@mindeck-tech.com



Intalnirea 10

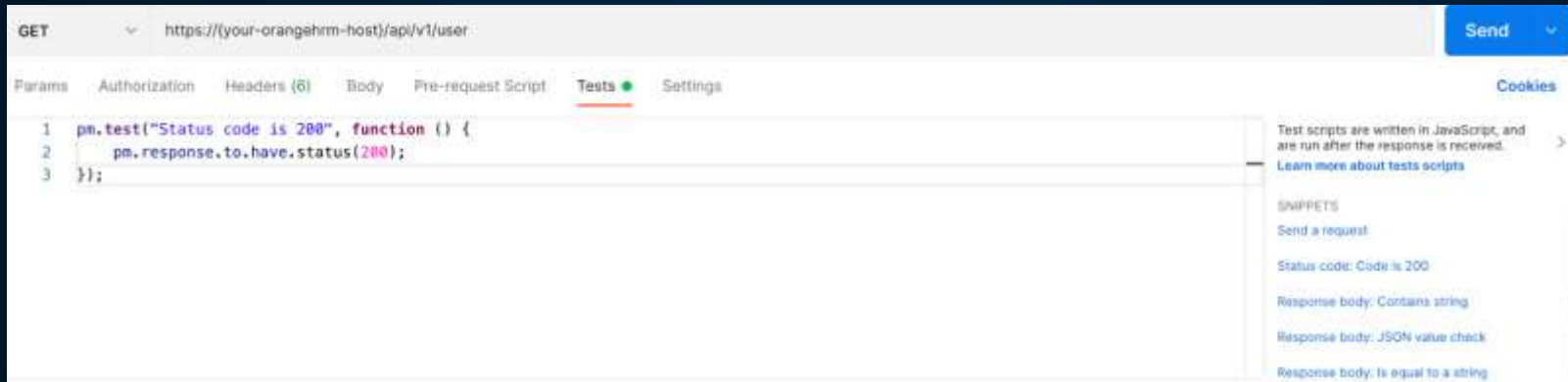
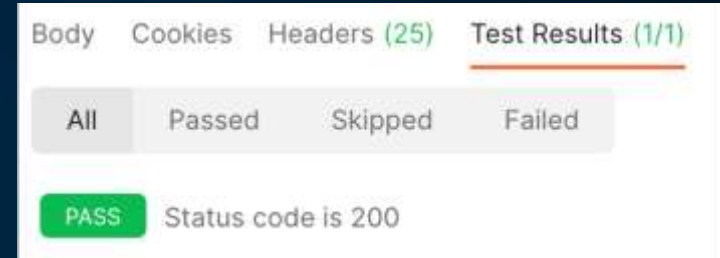
Testarea de API
- Studiu Individual -

Ce sunt testele Postman?

- . Testele Postman sunt bucati de cod JavaScript care sunt adaugate unui request pentru a verifica rezultatul unui test, cum ar fi statusul, mesajul, raspunsul etc
- . De obicei incepe cu keyword-urile ***pm.test.***

Testele Postman - status code

- Aceseaza unu request de GET si da click pe tabul tests.
- In partea dreapta vor fi disponibile un set de snippets.
- Din sectiunea snippets click pe **"Status code: Code is 200"**. Corpul sectiunii **Tests** va fi populat automat
- Click pe butonul **Send**. Rezultatele testelor ar trebui sa fie afisate in zona de raspuns, tab-ul Test Results.



Postman tests

- Verifica rezultatele-

Mergem inapoi in tab-ul de tests si vom face un nou test, dar de data asta vom testa raspunsul requestului.

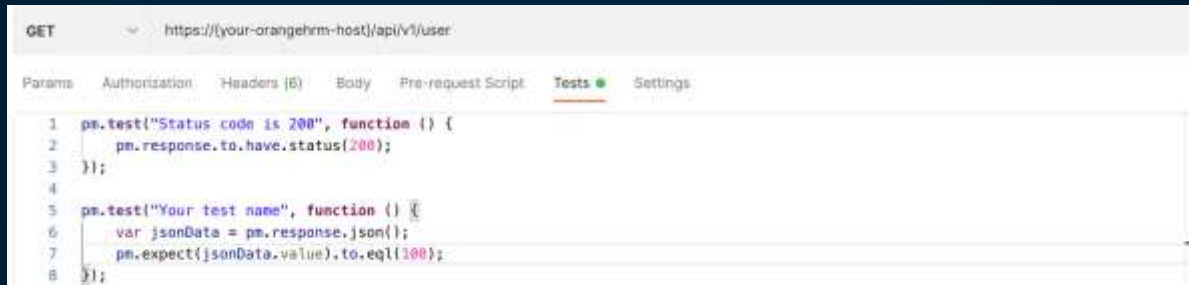
Din sectiunea de snippets click pe **“Response body:JSON value check”**. Vom verifica daca Leanne Graham are id-ul 1.

Inlocuieste textul **“Your Test Name”** cu numele testului tau, adica **“Check if user with id 1 is Leanne Graham”**

Inlocuieste **jsonData.value** cu **jsonData[0].name**.

Pentru a verifica ce trebuie sa scrii, intra in body-ul raspunsului de la rezultatul requestului de GET. Din moment ce Leanne Graham are userid 1, jsonData este primul rezultat din lista. Daca ai nevoie de un al doilea rezultat, vei scrie jsonData[1] si asa mai departe pentru alte rezultate. In to eql, vei scrie **“Leanne Graham”**

Testul ar trebui sa arate cam asa:



```
GET https://(your-orangehrm-host)/api/v1/user

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

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.value).to.eql(100);
8 });
```

Click pe send. Acum ar trebui sa fie doua teste passed in sectiunea de rezultate ale request-ului tau;

***Nota:** Sunt multe teste care pot fi create cu postman. Incearca sa le explorezi sa vezi cum poti sa le folosesti.

Alte Teste

```
pm.test("The response it's an object"), () =>
{
    const responseJson = pm.response.json();
    pm.expect(responseJson).to.be.an("object");

};

-----

pm.test("The response has the ID field present"), () => {
    const responseJson = pm.response.json();
    pm.expect(responseJson).to.include("id");
};

-----

pm.test("The response has the BookID field present"), () => {
    const responseJson = pm.response.json();
    pm.expect(responseJson).to.include("bookId");
};

pm.test("The response has the Timestamp field present"), () =>
{
    const responseJson = pm.response.json();
    pm.expect(responseJson).to.include("Timestamp");
};
```

```
pm.test("The response has the CustomerName field present"), () =>
{
    const responseJson = pm.response.json();
    pm.expect(responseJson).to.include("CustomerName");

};

-----

pm.test("The response has the CreatedBy field present"), () => {
    const responseJson = pm.response.json();
    pm.expect(responseJson).to.include("CreatedBy");

};

-----

pm.test("The response has the Quantity field present"), () => {
    const responseJson = pm.response.json();
    pm.expect(responseJson).to.include("quantity");

};
```

Alte Teste

```
const raspuns = pm.response.json();
  console.log("Afisam datele pentru comanda cu id-ul:",
  raspuns[0].id)
  console.log("Id-ul este:", raspuns[0].id)
  console.log("Book Id-ul este:", raspuns[0].bookId)
  console.log("CustomerName-ul este:", raspuns[0].customerName)
  console.log("Created by:", raspuns[0].createdBy)
  console.log("Cantina este:", raspuns[0].quantity)
  console.log("Timestamp-ul este:", raspuns[0].timestamp)
```

```
-----

  console.log("Afisam datele pentru comanda cu id-ul:",
  raspuns[1].id)
  console.log("Id-ul este:", raspuns[1].id)
  console.log("Book Id-ul este:", raspuns[1].bookId)
  console.log("CustomerName-ul este:", raspuns[1].customerName)
  console.log("Created by:", raspuns[1].createdBy)
  console.log("Cantina este:", raspuns[1].quantity)

  console.log("Timestamp-ul este:", raspuns[1].timestamp)
```

```
const response = pm.response.json();
console.log(response.status);
```

```
pm.test("Parsing the status from the response body ", () =>
{
    pm.expect(response.status).to.eql("OK");
})
```

```
-----

const response = pm.response.json()
const nonFiction = response.filter((book) =>
book.available === true);
console.log(nonFiction[0])
const book = nonFiction[0];
pm.test("Carte gasita", () => {
    pm.expect(book).to.be.an('object');
    pm.expect(book.available).to.be.true;
    pm.expect(book.available).to.eql(true);
});
```

Note:

- Testele vor trebui adaptate in functie de request-ul pe care il faceti
- Orice neclarificari legate de API vor fi trimise pe adresa gabriela.radulescu@minideck-tech.com