PROIECT FINAL

DUMITRESCU FLAVIUS VIRGIL

LINK GIT HUB: HTTPS://GITHUB.COM/FLAVIUS878/PROIECTEXAMEN_2024

04.09.2024

PARTEA I - NOTIUNI TEORETICE

1. EXPLICAȚI PE SCURT CE SUNT CERINȚELE DE BUSINESS, LA CE NE FOLOSESC ȘI CINE LE CREEAZĂ
Sunt specificații detaliate create de echipele de management sau de proprietarii afacerilor; descrieri detaliate ale funcționalităților, performanței și comportamentului așteptat al software-ului. Cerințe pot include funcționalități specifice, interacțiuni cu utilizatorii, performanță, securitate și compatibilitate

2. DIFERENȚA ÎNTRE TEST CONDITION SI TEST CASE
Test condition: Este un criteriu (conditie) care trebuie indeplinita pentru ca un test case sa fie considerat passed.
Test case: Acesta reprezintă o serie de pași pe care îi veți executa pentru a verifica o anumită funcționalitate. Prin urmare, este modalitatea prin care veți testa respectivul aspect.

3. ENUMERAȚI ȘI EXPLICAȚI PE SCURT ETAPELE PROCESULUI DE TESTARE
- Planificarea testării: Definirea strategiei și scopului testării.
- Analiza și proiectarea testelor: Identificarea cerințelor și proiectarea testelor corespunzătoare.
- Implementarea testelor: Crearea și configurarea mediului de testare și a cazurilor de test.
- Executarea testelor: Rularea cazurilor de test și înregistrarea rezultatelor.
- Evaluarea criteriilor de ieșire: Determinarea dacă testarea este completă.
- Raportarea: Documentarea și comunicarea rezultatelor testării.
- Închiderea testării: Finalizarea activităților de testare și arhivarea rezultatelor.

Mai multe detalii se regasesc in slide-urile de mai jos.

4. EXPLICAȚI DIFERENȚA ÎNTRE RETESTING ȘI REGRESSION TESTING

- Retesting este procesul de testare a unei componente sau funcționalități după ce a fost remediată o eroare. Regression testing este testarea sistemului pentru a se asigura că modificările recente nu au afectat funcționalitățile existente.

PARTEA I - NOTIUNI TEORETICE

5. EXPLICAȚI DIFERENȚA ÎNTRE FUNCTIONAL TESTING ȘI NON-FUNCTIONAL TESTING

- Functional testing verifică dacă sistemul îndeplinește cerințele specificate, concentrându-se pe ce face sistemul. Non-functional testing evaluează calitățile sistemului, cum ar fi performanța, securitatea și utilizabilitatea, concentrându-se pe cum funcționează sistemul.

6. EXPLICAȚI DIFERENȚA ÎNTRE BLACKBOX TESTING ȘI WHITEBOX TESTING

- Blackbox testing testează funcționalitatea sistemului fără a cunoaște structura internă a codului. Whitebox testing implică testarea structurii interne și a fluxurilor logice ale sistemului, necesitând cunoștințe minime de programare.

7. ENUMERAȚI TEHNICILE DE TESTARE ȘI GRUPAȚI-LE ÎN FUNCȚIE DE CATEGORIE (BLACKBOX, WHITEBOX, EXPERIENCE-BASED)

- Tehnici de testare:
  - Blackbox: Equivalence partitioning, Boundary value analysis, Decision table testing, State trasnistion testing.
  - Whitebox: Statement coverage, Branch coverage, Path coverage.
  - Experience-based: Exploratory testing, Error guessing.

8. EXPLICAȚI DIFERENȚA ÎNTRE VERIFICATION ȘI VALIDATION

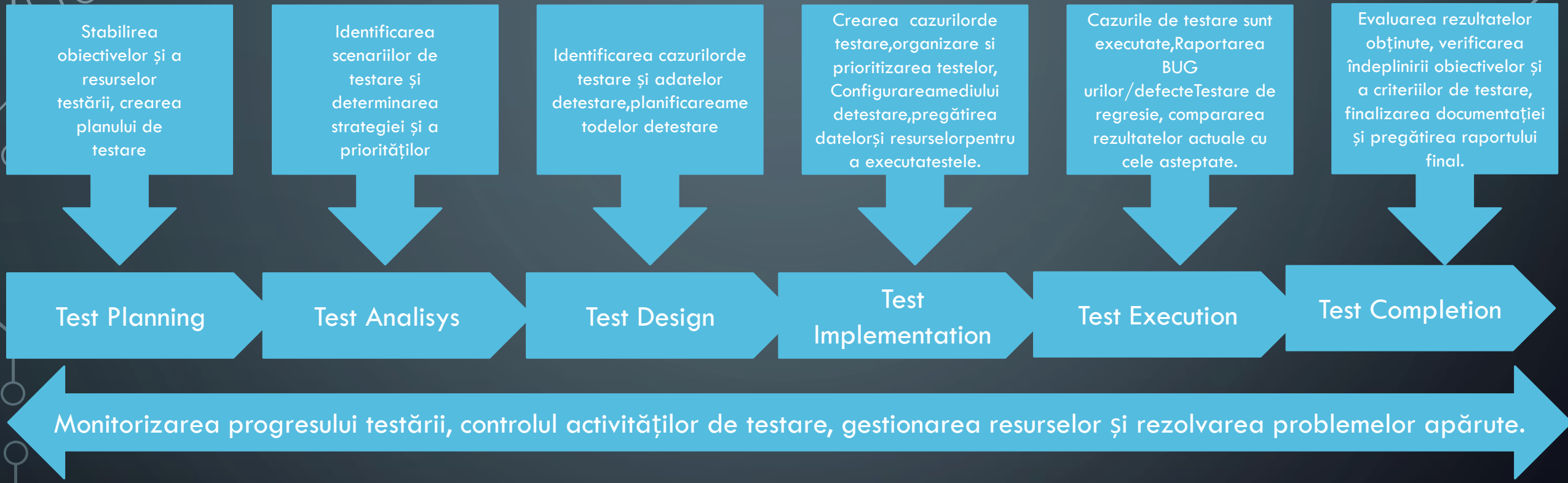
Verification: Acesta reprezintă procesul de evaluare prin care se determină dacă un produs sau sistem respectă specificațiile și cerințele stabilite. Cu alte cuvinte, verificarea se concentrează pe a verifica dacă ceea ce a fost construit corespunde cu ceea ce a fost planificat și specificat.
Validation: Acesta este procesul de evaluare prin care se asigură că un produs sau sistem îndeplinește nevoile și așteptările utilizatorilor finali. În esență, validarea se concentrează pe a verifica dacă ceea ce a fost construit este util și funcțional pentru utilizatori.

PARTEA I - NOTIUNI TEORETICE

- 9. •EXPLICAȚI DIFERENȚA ÎNTRE POSITIVE TESTING ȘI NEGATIVE TESTING ȘI DAȚI CÂTE UN EXEMPLU DIN FIECARE
- Positive testing verifică dacă sistemul funcționează corect cu date valabile (ex: introducerea unei parole corecte).
- Negative testing verifică dacă sistemul gestionează adecvat date invalide (ex: introducerea unei parole greșite și verificarea mesajului de eroare)
- 10. ENUMERAȚI ȘI EXPLICAȚI PE SCURT NIVELURILE DE TESTARE
- • Nivelurile de testare includ:
- o Unit testing: Testarea componentelor individuale ale codului.
- Component Testing: Se concentrează petestarea unui singur modul dintr-o aplicație
- o Integration testing: Testarea interacțiunilor dintre componente.
- o System testing: Testarea întregului sistem integrat pentru a verifica conformitatea cu cerințele specificate.
- o Acceptance testing: Testarea realizată de utilizatori finali pentru a verifica dacă sistemul îndeplinește nevoile și așteptările acestora.

PARTEA I - NOTIUNI TEORETICE

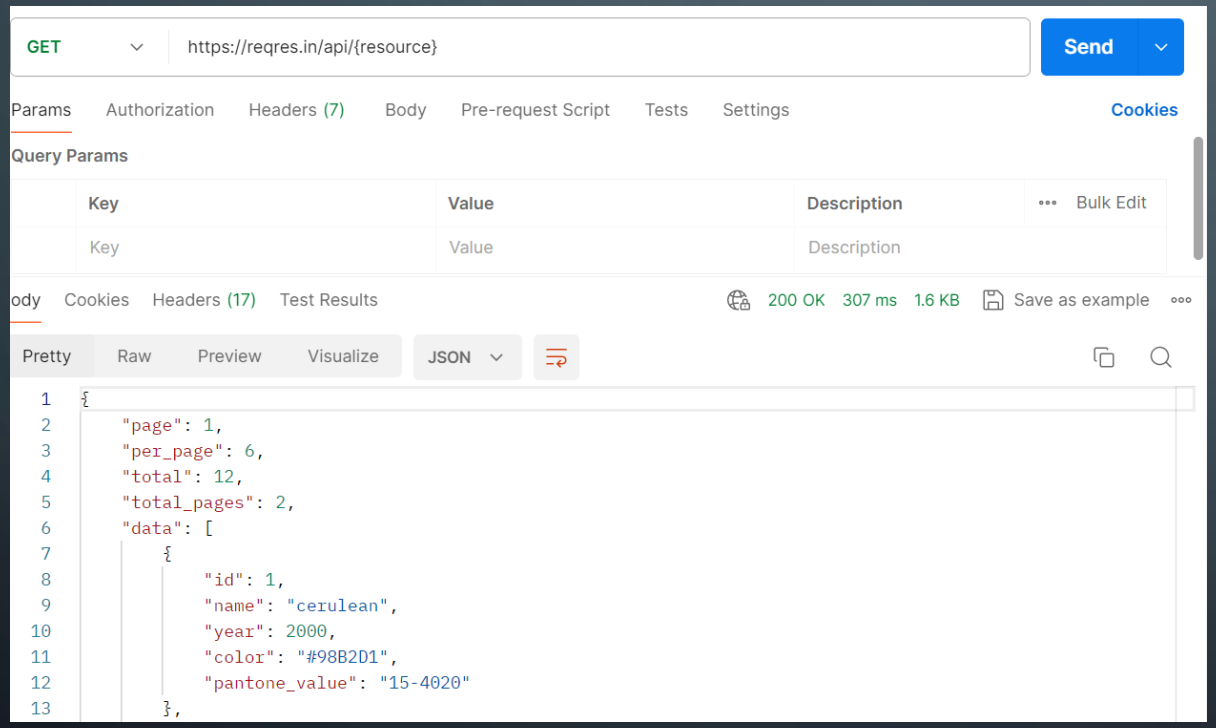
Etapele procesului de testare



PARTEA II - ASPECTE PRACTICE

Testarea API in Postman – site si documentatie https://reqres.in/api.

Metoda GET



Status 200 OK – The request succeeded
307 ms – timpul de raspuns

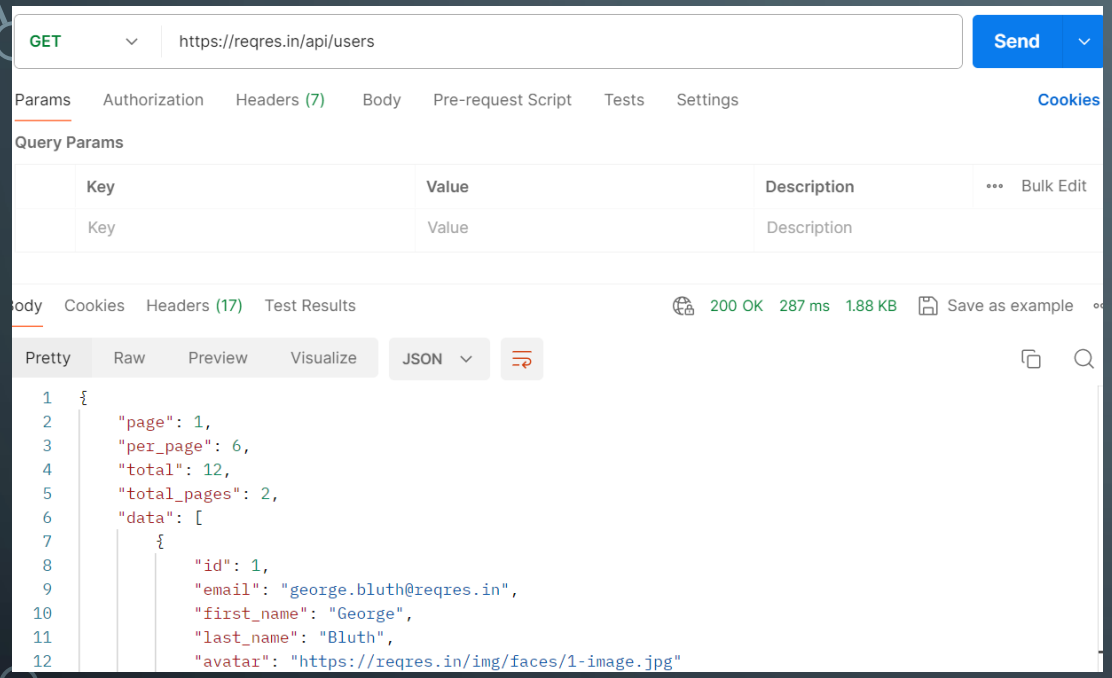
GET (Obține):
Rol: Solicită informații de la un server.

Descriere: Comanda GET este utilizată pentru a obține date de la un URL specific. Răspunsul conține informațiile cerute, cum ar fi pagini web, imagini sau alte resurse.

Interogarea endpoin-ului prin metoda get are rol de testare pozitiva si rezultatul adus reprezinta

PARTEA II - ASPECTE PRACTICE

Metoda GET – lista users

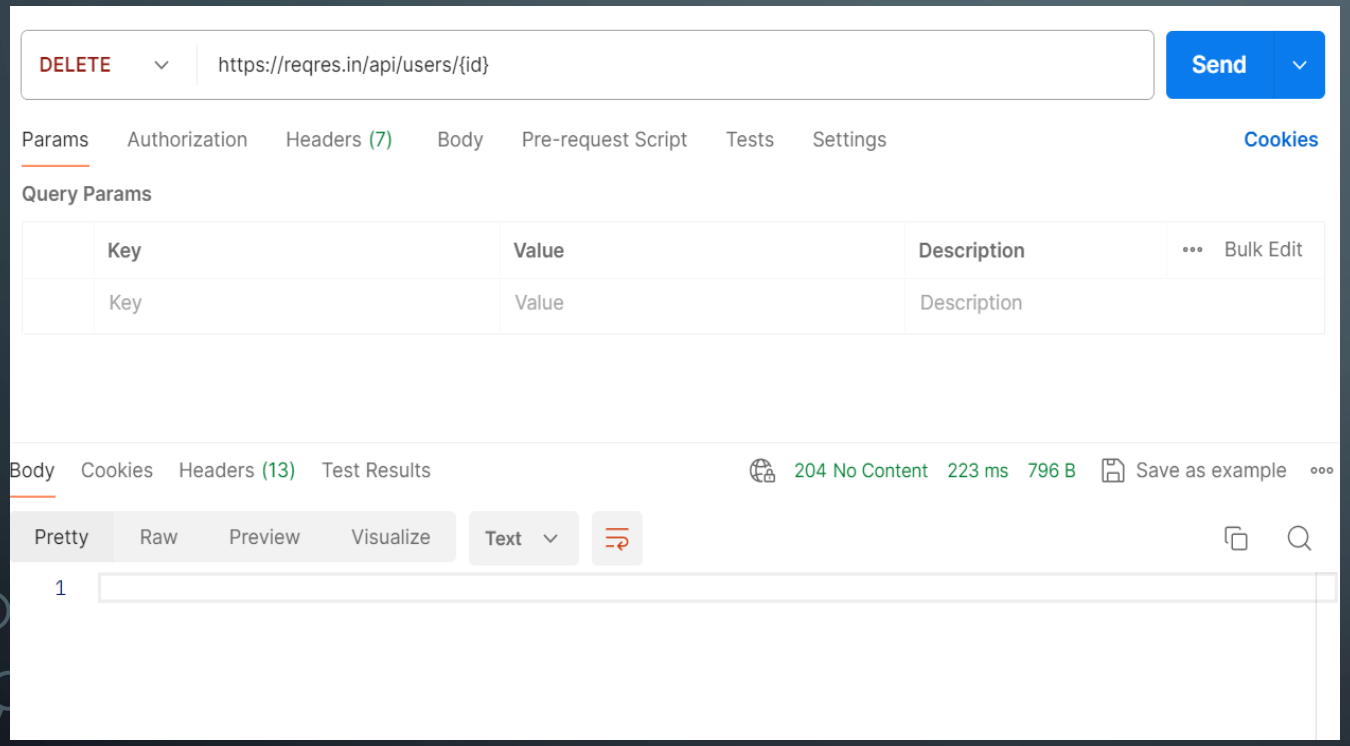


Interogarea endpoin-ului prin metoda get are rol de testare pozitiva si rezultatul adus reprezinta un exemplu de testare pozitiva.

Aici a fost interogat endpoint-ul users iar rezultatul din campul Body ecidentiaza datele uni user inregistrat in baza de date a API-ului.

PARTEA II - ASPECTE PRACTICE

Metoda DELETE – Delete user



Status 204: Fara continut. Status pozitiv conform comenzii.

DELETE (Șterge)

Rol: Șterge o resursă de pe server.

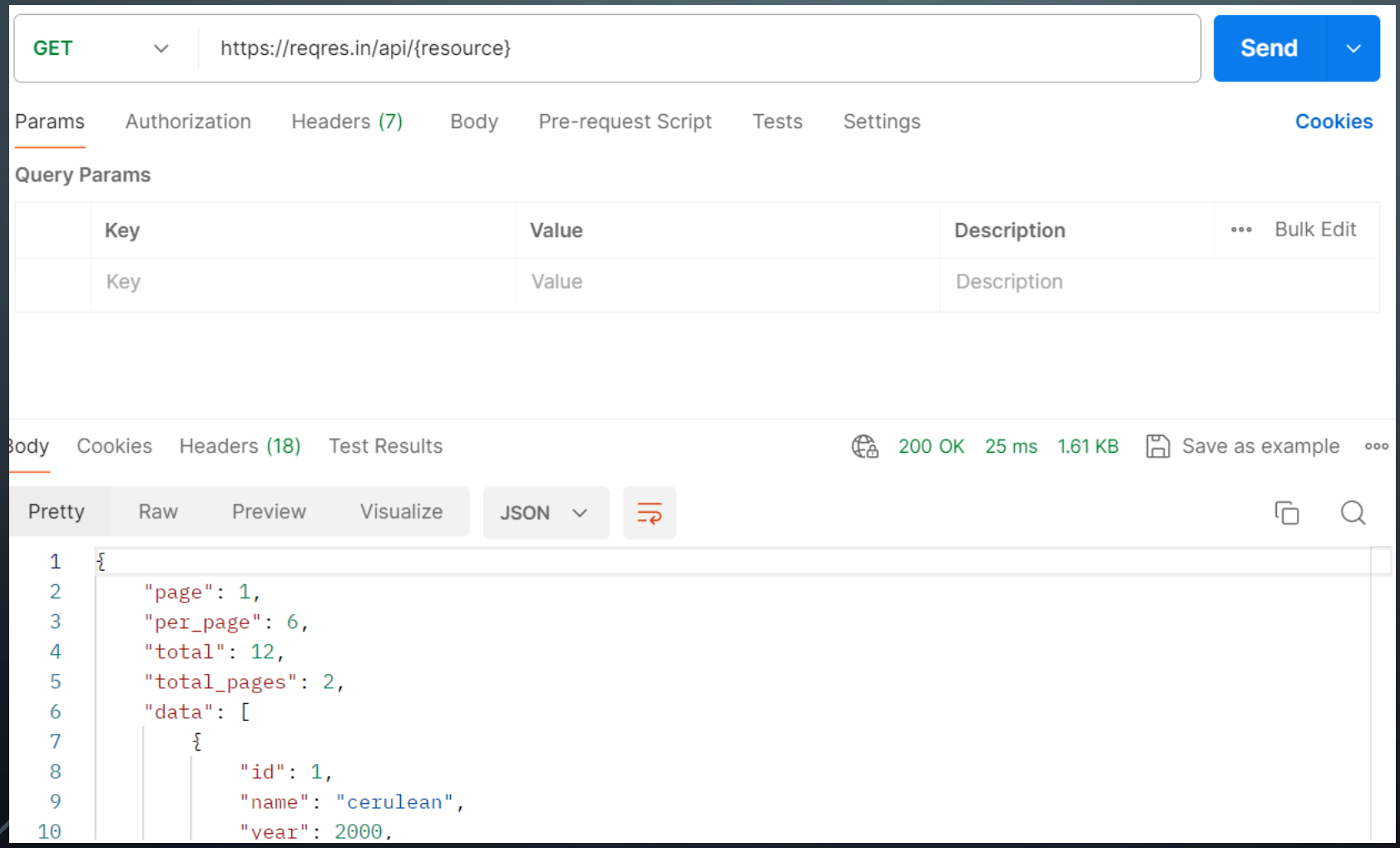
NU AVEM rezultat returnat in campul body deoarece userul a fost sters iar actiunea este confirmata de status.

Descriere: Comanda DELETE este folosită pentru a șterge o resursă specifică de pe server. De exemplu, poți șterge un articol dintr-un blog sau un utilizator dintr-o bază de date.

Aceats interogare reprezinta o tehnica de testare pozitiva.

PARTEA II - ASPECTE PRACTICE

Metoda GET – resource – identifica resursa/situl




```
{
    "id": 1,
    "name": "cerulean",
    "year": 2000,
    "color": "#98B2D1",
    "pantone_value": "15-4020"
},
```

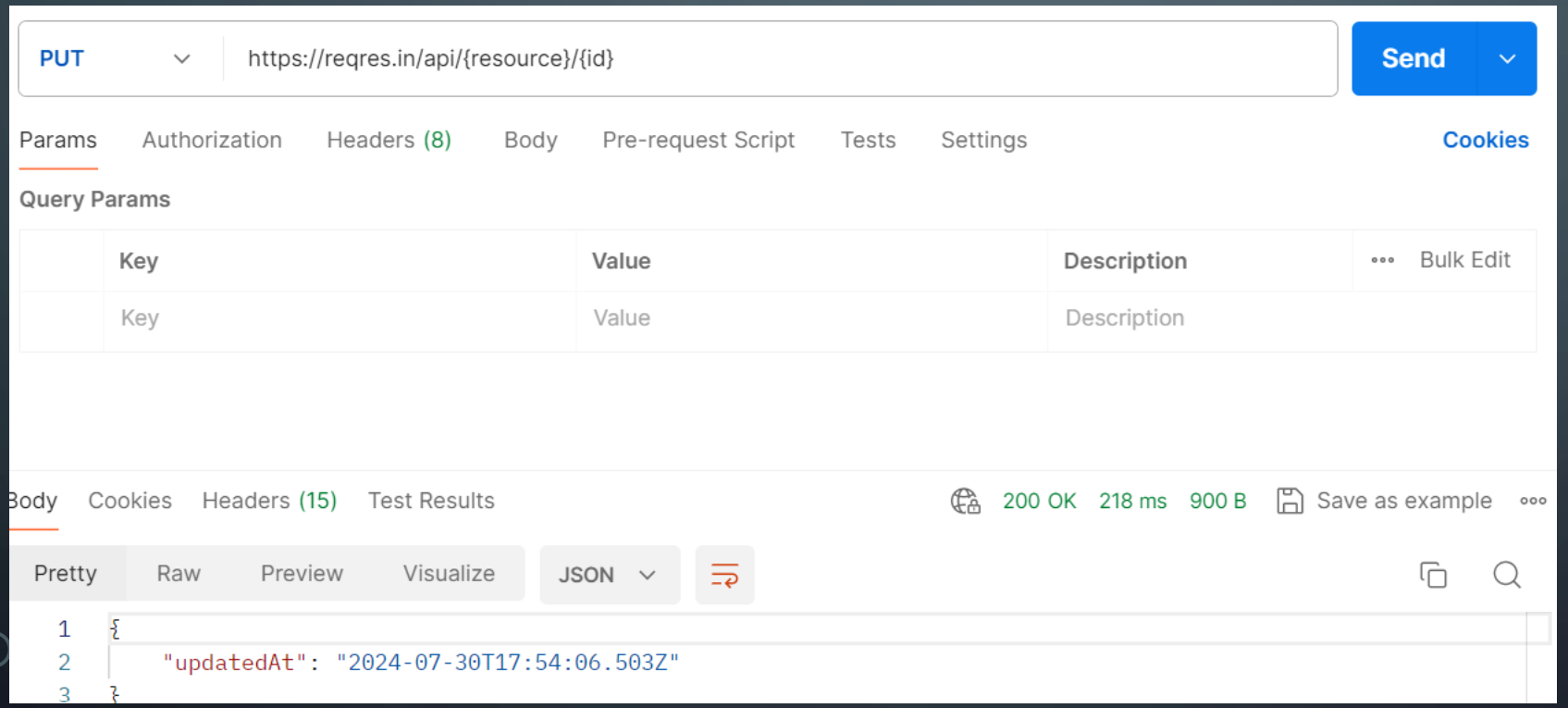

Rezultatul adus de comnda in urma interogarii. Aici avem detaliile unei reurse/site in limbaj de programare JSON.

Rezultatul adus se refera la specificatiile unei culori dintr-un catalog din baza de date a Api-ului si reprezinta un rezultat pozitiv asteptat.

Daca ar fi sa incadram aceasta interogare intr-o tehnbica de testare am putea afirma ca este o tehnica de testare pozitiva.

PARTEA II - ASPECTE PRACTICE

Comanda PUT – user ID



PUT (Înlocuiește):
Rol: Înlocuiește complet o resursă existentă sau o creează dacă nu există.

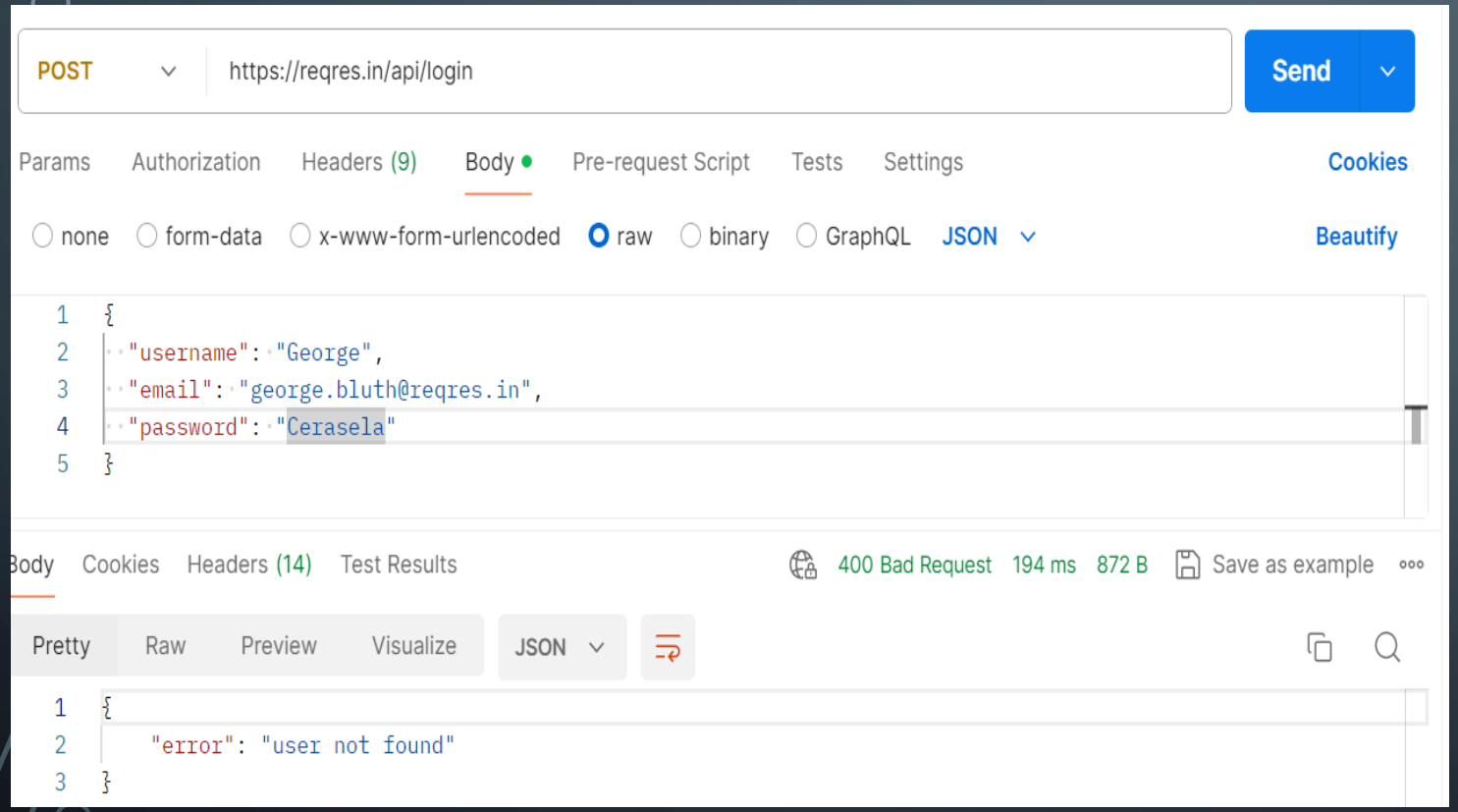
Descriere: Comanda PUT este folosită pentru a înlocui complet o resursă existentă cu una nouă. Dacă resursa nu există, este creată. Este utilizată pentru actualizări complete.

In situatia prezentata rezultatul returnat din campul “Body” ne indica faptul ca intentia de Update a fost receptionata la data si ora mentionate.

Fiind un API de test comanda nu modifica literalmente nimic ci doar interogheaza API cu privire la capacitatea de receptionare a unei comenzi PUT.

PARTEA II - ASPECTE PRACTICE

Comanda POST – user not found



400 Bad Request 194 ms 872 B

Statusul 400 reprezinta faptul ca serverul nu poate procesa solicitarea datorita unei erori in cerere facuta(cazul de fata solicitarea de login a unui user inexistent.

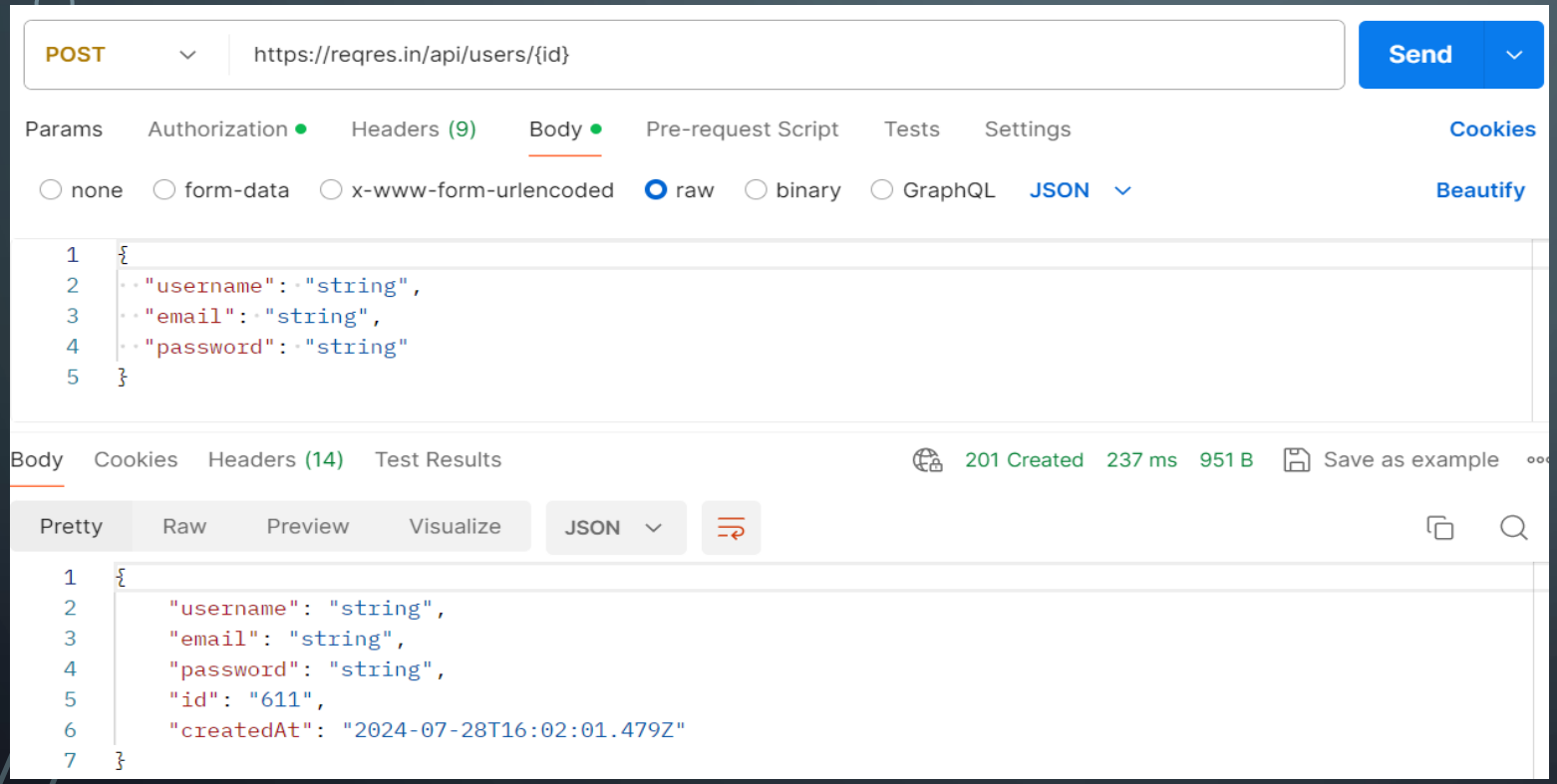
POST (Trimite):

- **Rol**: Trimite date către un server pentru a fi procesate.
- **Descriere**: Comanda POST este folosită pentru a trimite date către server, de exemplu, pentru a crea un nou obiect sau a efectua o acțiune. De obicei, este utilizată în formulare web sau pentru a adăuga date într-o bază de date.
- In imaginea alaturat avem exemplu de accesare a aceesare a unui endpoint de logare al API-ului cu un user, parola si adresa de email insa putem observa din raspunsul oferit in campul Body cum userul nu poate fi gasit in baza de date a Api-ului.
- Motivul pentru care userul nu se regaseste este ca nu a fost inregistrat ca user valid in baza de date anterior.
- In campul Body putem vedea mesajul returnat de catre Api ce insemna ca userul nu a fost gasit.
- Tehnica de testare este una de testare *negativa*: introducerea unor date inexistente in campul body al interogarii cu asteparea unui rezultat negative.

```
{
    "error": "user not found"
}
```

PARTEA II - ASPECTE PRACTICE

Metoda POST – Creare user



Aici prin comanda POST se acceseaza endpointul specificat pentru a crea un user nou.

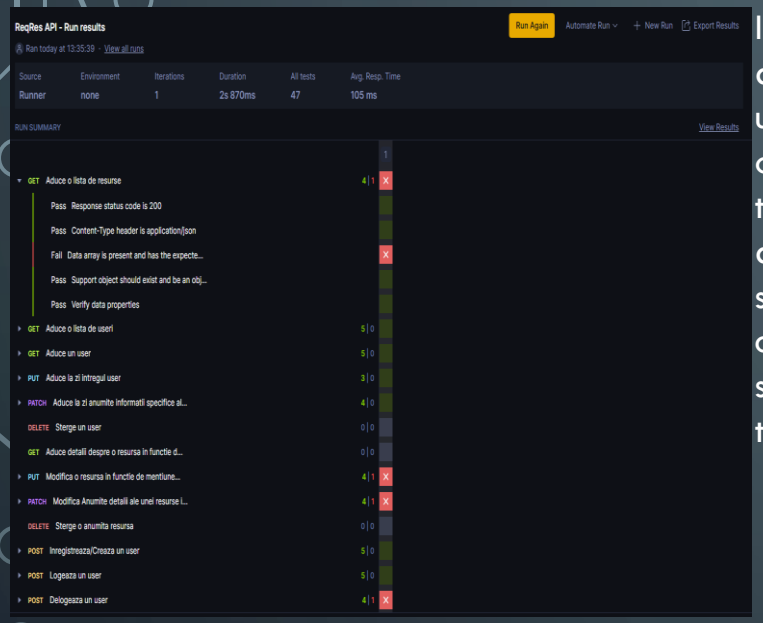
API-ul folosit permite testarea unui comenzi generice de creare aunui user respectand cerintele recomandate in documentarie.

In sectiunea de jos, campul Body, putem observa rezultatul returnat de creare user conform instructiunilor din documentatie. Acesta ne returneaza un raspuns pozitiv cu privire la creare si functioneaza conform documentatiei.

Motivul pentru care nu s-au introdus date reale este datorat dorintei de a pastra acest API de test intact si de a nu oferi posibilitatea crearii unui

PARTEA II - ASPECTE PRACTICE

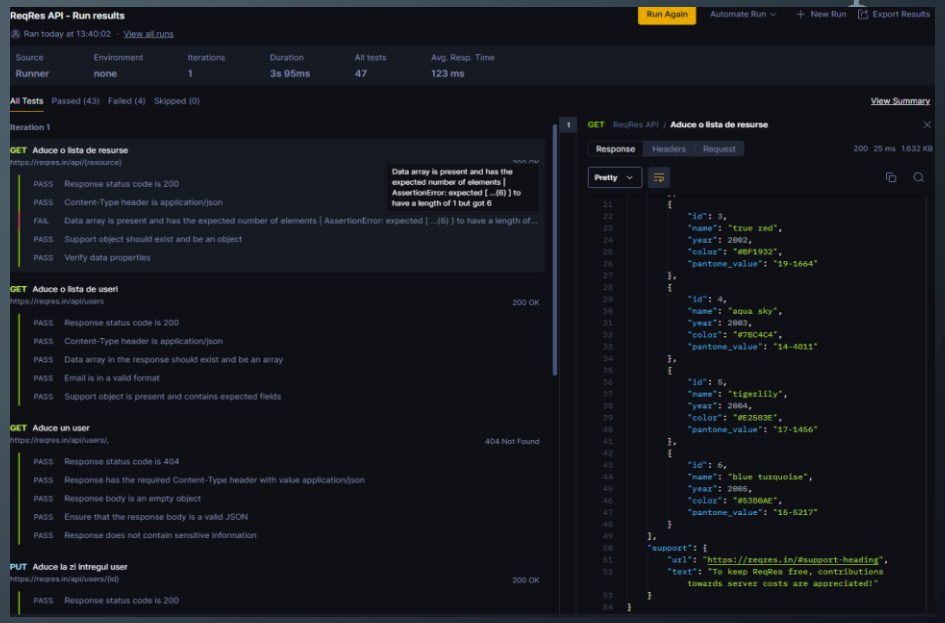
Raport final (stanga)



In partea stanga avem un raport final unde se poate observa o serie de teste executate cu ajutorul Postman cat si cele care au trecut conform asteptarilor si cele care nu au trecut.

Si rezultatele (dreapta)

In partea dreapta avem o ferestra ce ne arata scriptul testelor cat si rezultatele/raspunsul primit de la API in urma interogarii sale.



CONCLUZII FINALE

1. Validarea API-ului:

•**Funcționalitate:** API-ul a fost testat demonstrative pentru a se asigura că toate funcționalitățile sale corespund specificațiilor inițiale.

•**Erori și rezolvarea lor:** Au fost identificate diverse bug-uri și erori care ar fi putut afecta funcționarea aplicației.

2. Performanța:

•**Timp de răspuns:** Testele de performanță au arătat un timp de răspuns adecvat pentru majoritatea cererilor, asigurând astfel o experiență de utilizator fluentă.

•**Stabilitate:** API-ul a demonstrat stabilitate și fiabilitate sub diferite condiții de încărcare, tinand cont ca este un API demonstrativ.

LESSONS LEARNED

• **Procesul de testare:**

Importanța unei metodologii riguroase de testare pentru asigurarea calității produsului final este un must.

Nu intodeauna o documentatie API este corecta drept urmare o testare statica este foarte necesara pentru a aduce corectiile necesare din timp si a evita ingreunarea fazei de executie a testarii practice.

Testarea minutioasa a fiecarui rezultat adus inca de la prima interogare poate evidentia erori ce nu sunt banuite din simpla citire a rezultatelor.

Utilizarea a macar doua tehnici de testare (negative si pozitiva in acest caz) este cruciala in descoperirea bu-urilor/erorilor unui API

MULTUMESC PENTRU ATENTIE!