



# PROIECT FINAL

DUMITRESCU FLAVIUS VIRGIL

LINK GIT HUB: [HTTPS://GITHUB.COM/FLAVIUS878/PROIECTEXAMEN\\_2024](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. 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

## 4. 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.

## 5. 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.

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

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

## 7. EXPLICAȚI DIFERENȚA ÎNTRE VERIFICATION ȘI VALIDATION

*Verification:* Acesta reprezintă tipul de testare 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 tipul de testare 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.

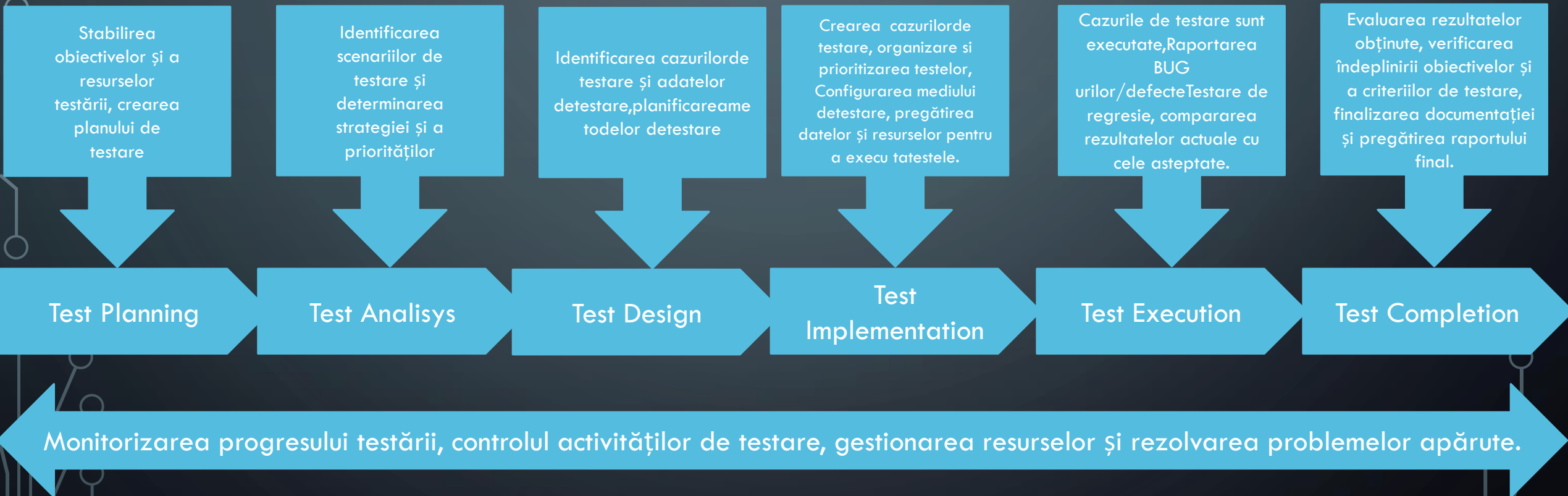
# PARTEA I - NOTIUNI TEORETICE

- 8. •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)
- 9. ENUMERAȚI ȘI EXPLICAȚI PE SCURT NIVELURILE DE TESTARE
  - Nivelurile de testare includ:
    - Component Testing: Se concentrează pe testarea unui singur modul dintr-o aplicație
    - Integration testing: Testarea interacțiunilor dintre componente.
    - System testing: Testarea întregului sistem integrat pentru a verifica conformitatea cu cerințele specificate.
    - Acceptance testing: Testarea realizată de utilizatori finali pentru a verifica dacă sistemul îndeplinește nevoile și așteptările acestora.

# PARTEA I - NOTIUNI TEORETICE

10. Enumerați și explicați pe scurt etapele procesului de testare

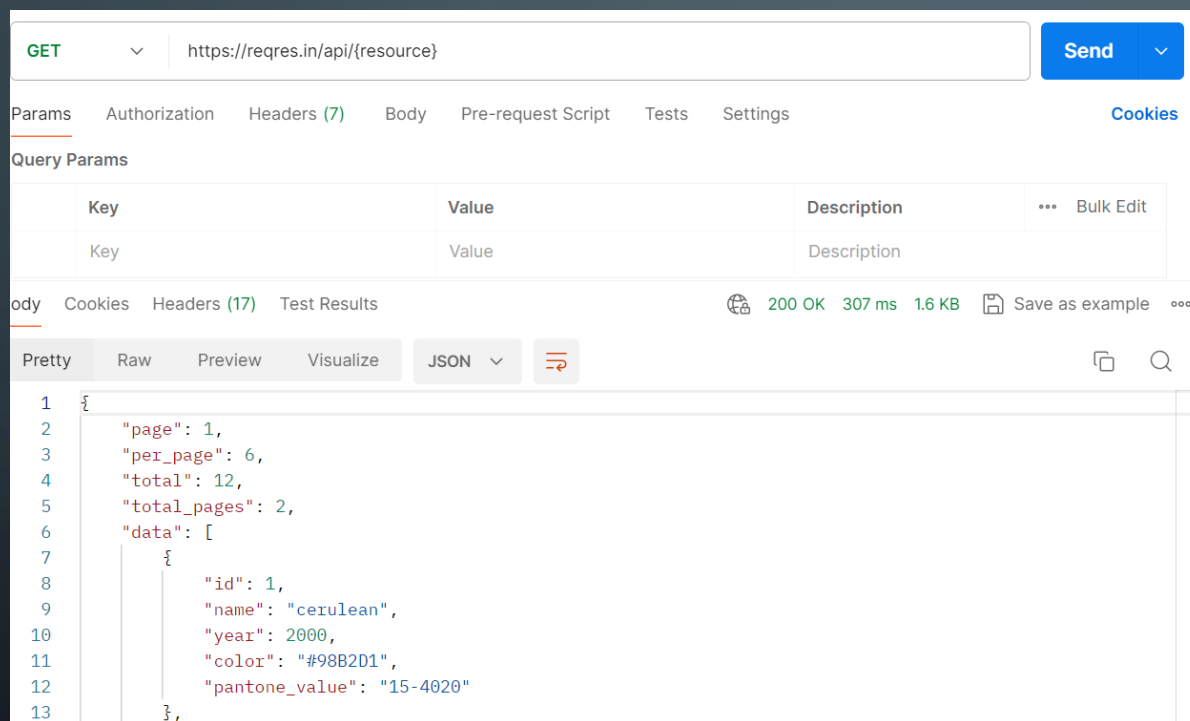
Etapele procesului de testare:



# PARTEA II - ASPECTE PRACTICE

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

## Metoda GET



200 OK 307 ms 1.6 KB

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

# PARTEA II - ASPECTE PRACTICE

## Metoda GET – lista users

The screenshot displays a REST client interface with a GET request to `https://reqres.in/api/users`. The response status is 200 OK, with a response time of 287 ms and a size of 1.88 KB. The response body is shown in JSON format, indicating pagination and a list of users.

**Query Params**

Key	Value	Description
Key	Value	Description

**Body** Cookies Headers (17) Test Results

200 OK 287 ms 1.88 KB Save as example

JSON

```
1 {
2   "page": 1,
3   "per_page": 6,
4   "total": 12,
5   "total_pages": 2,
6   "data": [
7     {
8       "id": 1,
9       "email": "george.bluth@reqres.in",
10      "first_name": "George",
11      "last_name": "Bluth",
12      "avatar": "https://reqres.in/img/faces/1-image.jpg"
```

Interogarea endpoint-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 eidentiaza datele uni user inregistrat in baza de date a API-ului.



# PARTEA II - ASPECTE PRACTICE

## Metoda DELETE – Delete user

DELETE

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

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (13) Test Results

Pretty Raw Preview Visualize Text

1



204 No Content 223 ms

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.

Aceasta interogare reprezintă o tehnică de testare pozitivă.



# PARTEA II - ASPECTE PRACTICE

Metoda GET – resource – identifica resursa/situl

The screenshot shows a REST client interface with a GET request to `https://reqres.in/api/{resource}`. The response is a 200 OK status with a 25 ms latency and 1.61 KB body. The response body is displayed in JSON format, showing a list of resources. The first resource in the list is:

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

```
{
  "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 tehnica de testare am putea afirma ca este o tehnica de testare pozitiva.

# PARTEA II - ASPECTE PRACTICE

## Comanda PUT – user ID

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

- Method:** PUT
- URL:** `https://reqres.in/api/{resource}/{id}`
- Buttons:** Send, Cookies
- Tabs:** Params, Authorization, Headers (8), Body, Pre-request Script, Tests, Settings
- Query Params Table:**

Key	Value	Description
Key	Value	Description
- Response Status:** 200 OK, 218 ms, 900 B
- Response Body (JSON):**

```
{
  "updatedAt": "2024-07-30T17:54:06.503Z"
}
```

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.

În situația prezentată rezultatul returnat din câmpul “Body” ne indică faptul că intenția de Update a fost recepționată la data și ora menționate.

Fiind un API de test comanda nu modifică literalmente nimic ci doar interoghează API cu privire la capacitatea de recepționare a unei comenzi PUT.

# PARTEA II - ASPECTE PRACTICE

## Comanda POST – user not found

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

- Method:** POST
- URL:** https://reqres.in/api/login
- Body (raw JSON):**

```
1 {  
2   "username": "George",  
3   "email": "george.bluth@reqres.in",  
4   "password": "Cerasela"  
5 }
```
- Status:** 400 Bad Request (194 ms, 872 B)
- Response Body (JSON):**

```
1 {  
2   "error": "user not found"  
3 }
```

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.
- În imaginea alaturat avem exemplu de accesare 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.
- În 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 asteptarea unui rezultat negativ.

```
1 {  
2   "error": "user not found"  
3 }
```

# PARTEA II - ASPECTE PRACTICE

## Metoda POST – Creare user

The screenshot displays a REST client interface with a POST request to the endpoint `https://reqres.in/api/users/{id}`. The request body is a JSON object with the following structure:

```
1 {
2   "username": "string",
3   "email": "string",
4   "password": "string"
5 }
```

The interface includes tabs for Params, Authorization, Headers (9), Body (selected), Pre-request Script, Tests, and Settings. The Body tab is active, showing the raw JSON request. Below the request, the response is displayed in the 'Body' tab, showing a successful creation of a user with the following JSON:

```
1 {
2   "username": "string",
3   "email": "string",
4   "password": "string",
5   "id": "611",
6   "createdAt": "2024-07-28T16:02:01.479Z"
7 }
```

The response status is 201 Created, with a response time of 237 ms and a body size of 951 B. The interface also includes a 'Send' button and a 'Beautify' link.

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

API-ul folosit permite testarea unui comenzi generice de creare a unui 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)

ReqRes API - Run results						Run Again	Automate Run	+ New Run	Export Results
Run today at 13:35:39 · View all runs									
Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time				
Runner	none	1	2s 870ms	47	105 ms				
RUN SUMMARY						View Results			
+ GET Aduce o lista de resurse						4/1	X		
Pass: Response status code is 200									
Pass: Content-Type header is application/json									
Fail: Data array is present and has the expected...							X		
Pass: Support object should exist and be an obj...									
Pass: Verify data properties									
+ GET Aduce o lista de useri						5/0			
+ GET Aduce un user						5/0			
+ PUT Aduce la zi intrerug user						3/0			
+ PATCH Aduce la zi anumite informatii specifice al...						4/0			
DELETE Sterge un user						0/0			
GET Aduce detalii despre o resursa in functie d...						0/0			
+ PUT Modifica o resursa in functie de mentinere...						4/1	X		
+ PATCH Modifica Anumite detalii ale unei resurse L...						4/1	X		
DELETE Sterge o anumita resursa						0/0			
+ POST Inregistrare/Crearea unui user						5/0			
+ POST Logareaza un user						5/0			
+ POST Delogareaza un user						4/1	X		

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.

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

## Si rezultatele (dreapta)

ReqRes API - Run results						Run Again	Automate Run	+ New Run	Export Results
Run today at 13:40:02 · View all runs									
Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time				
Runner	none	1	3s 95ms	47	123 ms				
All Tests Passed (43) Failed (4) Skipped (0)						View Summary			
Iteration 1									
GET Aduce o lista de resurse						View Results			
https://reqres.in/api/resources									
PASS: Response status code is 200									
PASS: Content-Type header is application/json									
FAIL: Data array is present and has the expected number of elements   AssertionError: expected [ ... ] to have a length of 1 but got 6									
PASS: Support object should exist and be an object									
PASS: Verify data properties									
GET Aduce o lista de useri						200 OK			
https://reqres.in/api/users									
PASS: Response status code is 200									
PASS: Content-Type header is application/json									
PASS: Data array in the response should exist and be an array									
PASS: Email is in a valid format									
PASS: Support object is present and contains expected fields									
GET Aduce un user						404 Not Found			
https://reqres.in/api/users/									
PASS: Response status code is 404									
PASS: Response has the required Content-Type header with value application/json									
PASS: Response body is an empty object									
PASS: Ensure that the response body is a valid JSON									
PASS: Response does not contain sensitive information									
PUT Aduce la zi intrerug user						200 OK			
https://reqres.in/api/users/00									
PASS: Response status code is 200									

Response

Headers

Request

200 25 ms 1632 KB

Pretty

```
[{"id": 3, "name": "true red", "year": 2002, "color": "#FF1932", "pantone_value": "19-1664"}, {"id": 4, "name": "aqua sky", "year": 2003, "color": "#00B0F0", "pantone_value": "14-4811"}, {"id": 5, "name": "tigerlily", "year": 2004, "color": "#E6397C", "pantone_value": "17-1456"}, {"id": 6, "name": "blue turquoise", "year": 2005, "color": "#008080", "pantone_value": "15-5211"}], {"support": {"url": "https://reqres.in/#support-heading", "text": "To keep ReqRes free, contributions towards server costs are appreciated!"}}
```

# CONCLUZII FINALE

## 1. Validarea API-ului:

- **Funcționalitate:** API-ul a fost testat demonstrativ 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, ținând cont că este un API demonstrativ.

## LESSONS LEARNED

### • Procesul de testare:

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

Nu întodeauna o documentație API este corectă; drept urmare o testare statică este foarte necesară pentru a aduce corecțiile necesare din timp și a evita îngreunarea fazei de execuție a testării practice.

Testarea minuțioasă a fiecărui rezultat adus încă de la prima interogare poate evidenția erori ce nu sunt bănuite din simpla citire a rezultatelor. În acest caz un rezultat poate fi contraprobabil prin 2 sau mai multe tehnici de testare pentru a identifica un bug.

Utilizarea a macar două tehnici de testare (negative și pozitivă în acest caz) este crucială în descoperirea bug-urilor/erorilor unui API.

The image features a dark blue gradient background with a large, faint, light blue circle centered behind the text. In the four corners, there are decorative white line art elements resembling electronic circuit traces or a stylized city skyline, with small circles at various points along the lines.

**MULTUMESC PENTRU ATENTIE!**