

# Descriere generală

În practică, protocolul HTTP este folosit pentru mai multe tipuri de aplicații. Cele mai uzuale sunt:

1. Transferul de fișiere HTML, CSS și Javascript în browsere pentru pagini web.
2. Expunerea unor API-uri REST

Pentru a înțelege fundamentele arhitecturii REST, vom construi un client web în C++ care interacționează cu un REST API expus de un server prin protocolul HTTP.

## Obiective

- înțelegerea mecanismelor protocolului HTTP;
- interacțiunea cu un REST API;
- înțelegerea conceptelor des folosite în web precum JSON, sesiune, JWT;
- utilizarea unor biblioteci externe pentru manipularea obiectelor JSON REST API.

## Rezumat

Scopul temei este de a scrie un client HTTP în C/C++ care să interacționeze cu un REST API.

**Serverul** (infrastructură existentă) va expune un API (Application Programmable Interface) de tip REST (Representational State Transfer). Puteți să vă gândiți la el ca la o cutie neagră ce primește o serie de intrări reprezentate de rute HTTP. În urma cererilor HTTP, serverul efectuează o acțiune. În contextul temei, serverul simulează o bibliotecă online (de cărți) și este deja complet implementat.

**Clientul** (pe care îl veți construi) este un program scris în C/C++ care acceptă comenzi de la tastatură (stdin) și trimite, în funcție de comandă, cereri către server. Scopul lui este de a funcționa ca o interfață în linia de comandă (CLI) cu biblioteca virtuală.

# Descriere Server

## Datele de conectare

**HOST:** 34.246.184.49 **PORT:** 8080

## Mod de funcționare

Serverul va permite efectuarea următoarelor acțiuni:

### 1. Înregistrarea unui cont

Ruta de acces:

```
POST /api/v1/tema/auth/register
```

Tip payload:

```
application/json
```

Payload:

```
{  
  "username": String,  
  "password": String  
}
```

Erori tratate:



Întoarce eroare dacă username-ul este deja folosit de către cineva!

### 2. Autentificare

Ruta de acces:

```
POST /api/v1/tema/auth/login
```

Tip payload:

```
application/json
```

Payload:

```
{  
  "username": String,  
  "password": String  
}
```

Răspuns:

👍 Întoarce cookie de sesiune.

Erori tratate:

👎 Întoarce un mesaj de eroare dacă nu se potrivesc credențialele!

### 3. Cerere de acces în bibliotecă

Ruta de acces:

```
GET /api/v1/tema/library/access
```

Observații adiționale:

🍪 Trebuie să demonstrați că sunteți autentificați!

Răspuns:

👍 Întoarce cookie de sesiune.

Erori tratate:

👎 Întoarce un mesaj de eroare dacă nu se potrivesc credențialele!

### 4. Vizualizarea informațiilor sumare despre toate cărțile

Ruta de acces:

```
GET /api/v1/tema/library/books
```

Observații adiționale:

🍪 Trebuie să demonstrați că aveți acces la bibliotecă!

Răspuns:

Întoarce o listă de obiecte json:

```
[  
  {  
    id: Number,  
    title: String  
  }  
]
```

Erori tratate:



Întoarce un mesaj de eroare dacă nu demonstrați că aveți acces la bibliotecă!

## 5. Vizualizarea detaliilor despre o carte

Ruta de acces:

```
GET /api/v1/tema/library/books/:bookId
```

Observații adiționale:



Trebuie să demonstrați că aveți acces la bibliotecă!

În loc de ":bookId" este un id de carte efectiv.

e.g: `/api/v1/tema/library/books/123`

Răspuns:

Întoarce un obiect json:

```
{  
  "id": Number,  
  "title": String,  
  "author": String,  
  "publisher": String,  
  "genre": String,  
  "page_count": Number  
}
```

Erori tratate:



Întoarce un mesaj de eroare dacă nu demonstrați că aveți acces la bibliotecă!



Întoarce un mesaj de eroare dacă id-ul pentru care efectuați cererea este invalid!

## 6. Adăugarea unei cărți

Ruta de acces:

```
POST /api/v1/tema/library/books
```

Tip payload:

```
application/json
```

Payload:

```
{
  "title": String,
  "author": String,
  "genre": String,
  "page_count": Number
  "publisher": String
}
```

Observații adiționale:

🍪 Trebuie să demonstrați că aveți acces la bibliotecă!

Erori tratate:

👎 Întoarce un mesaj de eroare dacă nu demonstrați că aveți acces la bibliotecă!

👎 Întoarce un mesaj de eroare dacă informațiile introduse sunt incomplete sau nu respectă formatarea!

## 7. Ștergerea unei cărți

Ruta de acces:

```
DELETE /api/v1/tema/library/books/:bookId
```

Observații adiționale:

🍪 Trebuie să demonstrați că aveți acces la bibliotecă!

În loc de ":bookId" este un id de carte efectiv.

e.g: `/api/v1/tema/library/books/123`

Erori tratate:

👉 Întoarce un mesaj de eroare dacă nu demonstrați că aveți acces la bibliotecă!

👉 Întoarce un mesaj de eroare dacă id-ul pentru care efectuați cererea este invalid!

## 8. Logout

Ruta de acces:

```
GET /api/v1/tema/auth/logout
```

Observații adiționale:

🍪 Trebuie să demonstrați că sunteți autentificați!

Erori tratate:

👉 Întoarce un mesaj de eroare dacă nu demonstrați că sunteți autentificați!

## Testare server

Pentru a interacționa neprogramatic cu serverul, puteți folosi utilitare ce simulează clienții HTTP, precum Postman [1], Insomnia [2] sau chiar clienți scriși de mână în alte limbaje de programare.

[1] <https://www.postman.com/>

[2] <https://insomnia.rest/>

# JSON & Token JWT

## JSON

JSON (JavaScript Object Notation) este un format de reprezentare / schimb de date text foarte popular, mai ales în sfera de programare Web, deoarece este și relativ ușor de construit & parsat de către mașini (în orice limbaj), cât și relativ inteligibil de către oameni (fapt ce ajută deseori la depanare):

```
{
  "employee": {
    "name": "sonoo",
    "salary": 56000,
    "married": true
  }
}
```

Majoritatea API-urilor web (REST-ful) folosesc deseori această reprezentare pentru a structura atât datele de intrare (e.g. POST body), cât și cele de ieșire (corp răspuns HTTP), marcate prin antetul `Content-Type: application/json`. La fel este folosit și de către serverul temei.

Astfel, pentru a parsea răspunsurile primite de la server, puteți (și e recomandat) să folosiți o bibliotecă. Vă sugerez parson [1] pentru C sau nlohmann [2] pentru C++, însă puteți folosi orice (inclusiv o soluție proprie), justificând alegerea în README.

[1] <https://github.com/kgabis/parson>

[2] <https://github.com/nlohmann/json>

## Token JWT

Tokenurile JWT (JSON Web Token) sunt o modalitate de transmitere a informațiilor de autentificare/autorizare dintre un client și un server. Acestea sunt codificate binar și semnate pentru verificarea integrității. Astfel se asigură că un potențial atacator nu poate modifica datele împachetate.

Pentru a trimite tokenul către server, este necesară adăugarea acestuia în headerul **Authorization**. Valoarea tokenului trebuie să fie prefixată de cuvântul **Bearer**.

```
Authorization: Bearer eijjkwuqioueu9182712093801293
```

Pentru mai multe informații puteți să consultați documentația oficială:

<https://jwt.io/introduction>



# Clientul

Clientul va trebui să interpreteze comenzi de la tastatură pentru a putea interacționa cu serverul. În urma primirii unei comenzi, clientul va forma obiectul json (dacă e cazul), va executa cererea către server și va afișa răspunsul acestuia (de succes sau de eroare). Procesul se repetă până la introducerea comenzii exit.

Atât comenzile cât și câmpurile împreună cu valorile aferente se scriu pe linii separate!

Datele introduse de către utilizator de la tastatura sunt cele marcate cu text **astfel**!

## Comenzile sunt următoarele:

- register - efectuează înregistrarea
- login - efectuează autentificarea
- enter\_library - cere acces în bibliotecă
- get\_books - cere toate cărțile de pe server
- get\_book - cere informație despre o carte
- add\_book - adaugă o carte
- delete\_book - șterge o carte
- logout - efectuează delogarea utilizatorului
- exit - efectuează ieșirea din program

### 1. Register (1p)

Oferă prompt pentru username și password.

```
register
```

```
username=something
```

```
password=something
```

Mesajele scrise cu **albastru** sunt prompt-uri afișate de către program la stdout, restul fiind input (stdin)!

## Input

```
register  
username=test  
password=test
```

## Output

```
Utilizator înregistrat cu succes!
```

---

**Atenție: Textul mesajelor returnate de server sunt alese de voi, însă trebuie să Țineți cont de următoarele convenții:**

- dacă rezultatul HTTP a fost un cod de eroare, mesajul să conțină unul din cuvintele `ERROR` sau `EROARE`;
  - dacă operațiunea a avut loc cu succes, atunci să conțină unul din cuvintele `SUCCESS` sau `SUCCES`.
- 

## 2. Login (1p)

Oferă prompt pentru username și password.

```
login
```

```
username=something
```

```
password=something
```

## Input

```
login  
username=test  
password=test
```

## Output

```
Utilizatorul a fost logat cu succes
```

### 3. Enter\_library (1p)

```
enter_library
```

#### Input

```
enter_library
```

#### Output

```
Utilizatorul are acces la biblioteca
```

### 4. Get\_books (2p)

```
get_books
```

#### Input

```
get_books
```

#### Output

```
[
  {
    "id": 1,
    "title": "Ion"
  },
  {
    "id": 2,
    "title": "Moara cu noroc"
  }
]
```

---

Am prefera să lăsați afișarea în format JSON (sau, cel puțin, să aibe formatul `"proprietate": valoare`) pentru ca procesul de verificare să fie automatizabil (vedeți checkerul).

---

## 5. Get\_book (1p)

Oferă prompt pentru id.

`get_book`

`id=10`

### Input

```
get_book
id=10
```

### Output

```
{
  "id": 10,
  "title": "Maitreyi",
  "author": "Mircea Eliade",
  "publisher": "Tana",
  "genre": "roman",
  "page_count": 175
}
```

## 6. Add\_book (2p)

Oferă prompt pentru title, author, genre, publisher, page\_count.

`add_book`

`title=testbook`

`author=student`

`genre=comedy`

`publisher=PCom`

`page_count=10`

## Input

```
add_book  
title=test  
author=test  
genre=test  
publisher=test  
page_count=this will not work
```

## Output

```
Tip de date incorect pentru numarul de pagini
```

## 7. Delete\_book (1p)

Oferă prompt pentru id.

```
delete_book
```

```
id=10
```

## Input

```
delete_book  
id=10
```

## Output

```
Cartea cu id 10 a fost stearsa cu succes!
```

## 8. Logout (0.5p)

```
logout
```

## Input

```
logout
```

## Output

```
Utilizatorul s-a delogat cu succes!
```

## 9. Exit (0.5p)

```
exit
```

## Input

```
exit
```

## Output

```
Inchidere program
```

---

**Închideți direct programul, fără să mai afișați un mesaj!**

---

# Exemplu sesiune

Comenzile vor fi testate in flux. Exemplele de mai jos nu reprezintă o listă exhaustivă a cazurilor care vor fi testate pentru corectarea temei.

## Exemplu

### 1. Login

```
login
```

```
username=test
```

```
password=test
```

Cum aceasta comandă este prima din flux, utilizatorul "test" cel mai probabil nu există. Este datorია clientului să primească răspunsul de la server și să informeze utilizatorul de acest fapt.

### 2. Register

```
register
```

```
username=test
```

```
password=test
```

Cum utilizatorul "test" nu există, aceasta comandă va avea succes. Puteți afișa un mesaj de succes precum "200 - OK" SAU "200 - OK - Utilizator înregistrat cu succes!". Formatul este la latitudinea voastră.

### 3. Get book

```
get_book
```

```
id=10
```

În acest moment, utilizatorul nu a executat o cerere de login și nici o cere de acces. Apelul va eșua. Clientul va primi răspunsuri corespunzătoare care trebuie să informeze și utilizatorul.

#### 4. Login

`login`

`username=test`

`password=test2`

În acest moment, utilizatorul nu a executat o cerere de login cu parola incorectă. Apelul va eșua. Clientul va primi răspunsuri corespunzătoare care trebuie să informeze și utilizatorul.

#### 5. Login

`login`

`username=test`

`password=test`

În acest moment, apelul se va executa cu succes. Puteți afișa mesaje precum "200 - OK - Bun venit!"

#### 6. Get access

`enter_library`

#### 7. Get book

`get_book`

`id=10`

În cazul în care cartea cu id-ul 10 nu există clientul va primi un răspuns corespunzător ce trebuie să informeze și utilizatorul de lipsa cărții, precum "Cartea cu id=10 nu exista!".

#### 8. Add book

`add_book`

`title=test`



author=`test`

genre=`test`

publisher=`test`

page\_count=`this will not work`

În acest caz clientul trebuie să execute o validare a datelor și să informeze utilizatorul de incorectitudinea datelor.

## 9. Logout

`logout`

## 10. Get book

`get_book`

id=`10`

În acest caz, cum un apel logout a fost executat înainte, clientul trebuie să piardă accesul la bibliotecă (chiar dacă tokenul poate să fie în continuare interpretat de server).

# Sistem de punctare

Punctarea se efectuează individual, pentru fiecare comandă realizată cu succes într-un flux de comenzi (vezi secțiunea [5.Exemplu sesiune](#)).

O comandă este considerată funcțională dacă, prin introducerea ei se trimite cererea bună către server și se afișează răspunsul acestuia (de succes sau de eroare). Formatul răspunsurilor este la latitudinea voastră, cât timp oferă suficientă informație legată de statusul comenzii. În plus, trebuie ținut cont de accesul la bibliotecă. De exemplu, dacă în urma apelului `logout`, utilizatorul încă mai are acces la bibliotecă, comanda `logout` nu va fi considerată funcțională.

Comenzile care trimit date la server (cele care execută `POST` sau `GET` și `DELETE` pe id-uri) sunt considerate funcționale dacă reușesc să trimită cu succes informația bună la server. De exemplu, o comandă de autentificare care parsează prost de la tastatură `username` sau `password`, dar totuși trimite către server informația preluată greșit și afișează întotdeauna răspunsul de eroare al serverului va fi considerată o comandă nefuncțională, deci nu va fi punctată.

## Checker

Pentru a verifica ușor corectitudinea, aveți la dispoziție un checker scris în Python pe GitLab: [pcom/homework4-public](https://pcom/homework4-public).

Vă recomandăm citirea Readme-ului acestuia pentru instrucțiuni de utilizare.

**Atenție:** checkerul (încă) nu calculează punctajul final, însă este un bun indicator al corectitudinii aplicației (pe scenarii precum ALL).

## Arhiva

Arhiva temei trebuie să conțină sursele de cod, un Makefile și un Readme prin care să explicați implementarea soluției voastre. Trebuie justificată și explicată și utilizarea bibliotecii de parsare JSON pe care ați ales să o folosiți.

Arhiva va avea numele **Grupa\_Nume\_Prenume\_Tema4PC**. Formatul arhivei trebuie să fie `.zip`.

# Mentiuni

**IMPORTANT!** A fost implementat în server și un mecanism de protecție pentru limitarea numărului de cereri. Dacă trimiteți prea multe cereri prea rapid veți primi un răspuns de forma "Too many requests, please try again later." Nu este nevoie să țineți cont de acest aspect sau să adăugați un caz special în implementare.

**IMPORTANT!** Atenție la tratarea titlurilor de cărți ce conțin spații.

Datele, headere-le, cookie-urile și token-urile trebuie puse în cerere și extrase din răspuns automat. Hardcodarea acestora va duce la anularea punctajului pentru cerința în care au fost utilizate. Pentru mai multe detalii legate de modul de funcționare consultați [Laboratorul 9](#).

Numele comenzilor trebuie respectat întocmai cum este precizat în enunț.

**!! Scrierea comenzilor cu alt nume duce către depunere pentru comanda respectivă.**

Numele câmpurilor din obiecte trebuie respectat întocmai cum este precizat în enunț. Altfel, veți primi permanent erori de pe server, deci comanda va fi depunctată.

Formatul comenzilor trebuie respectat întocmai cum este precizat în enunț. Fiecare comandă, respectiv câmp și valoarea sa pe linii separate. **Formatul nerespectat duce la pierderea punctajului pentru acea comandă.**

Schema de denumire a arhivei trebuie respectată întocmai. Modificarea acesteia duce la depunerea totală a temei.

**Lipsa README duce la depunerea totală a temei.**

# FAQ

- **Q: Cum demonstrez că utilizatorul are acces la bibliotecă?**

A: Un utilizator are acces la bibliotecă dacă a executat un apel "access". În urma apelului ar trebui să fie prezent un token JWT. Acest token este dovada accesului la bibliotecă. Voi trebuie să vă bazați întotdeauna pe răspunsul venit din partea serverului.

- **Q: Putem folosi codul din laborator?**

A: Da! [Laboratorul de HTTP](#) poate fi folosit ca schelet de cod pentru această temă. Dacă vreți să integrați altă bibliotecă externă, ar trebui să cereți o aprobare pe forumul temei.

- **Q: Am observat că dacă folosesc comanda logout avem în continuare acces la bibliotecă cu token-ul obținut anterior. De ce?**

A: Serverul nu invalidează tokenele. Puteți însă să eliberați zona de memorie asociată tokenului pe client.

- **Q: Va trebui să testăm, dacă se introduce pentru page\_count altceva în afara de un număr?**

A: Da! Validarea datelor este responsabilitatea voastră.

- **Q: La username și la password este permis să fie spații?**

A: Nu!

- **Q: În nume sau titluri de cărți pot să apară și numere, și spații, și cifre?**

A: Da! Validarea datelor este responsabilitatea voastră.