

PROTOCOALE DE COMUNICAȚIE

Tema 3 Client Web. Comunicație cu REST API.

Deadline soft: 22.05.2022

Responsabili Temă: *Andrei STANCA, Daniel ANGHEL, Ioana-Alina TUDORACHE, Mihai-Cătălin PUSCOCI, Irina-Nicoleta UNGUREANU, Răzvan ȘERBAN, Cătălin LEORDEANU*

1. Motivație

Un procent foarte mare din aplicațiile moderne este reprezentat de aplicații web. Multe dintre aceste aplicații web, în ciuda complexității pe care o au, se bazează pe modelul clasic “client-server”. Pentru a înțelege cum funcționează modelul client-server în contextul web modern, este nevoie să înțelegeți cum funcționează protocolul HTTP.

2. Obiectivele temei de casă

Scopul temei este un client scris în C/C++ care să interacționeze cu un REST API. Obiectivele pe care le urmărim sunt:

- înțelegerea mecanismelor de comunicare prin 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.

3. Descriere generală

Tema urmărește implementarea unui client web care să interacționeze cu un server. Deși tehnologia de facto pentru realizarea clienților web este triada *html, css, javascript*, pentru sedimentarea conceptelor am ales să folosim C/C++, pentru a ne apropia cât mai mult de protocol.

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

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

4. Interacțiunea cu Serverul

Datele de conectare

HOST: 34.241.4.235 PORT: 8080

Mod de funcționare

Serverul va permite efectuarea următoarelor acțiuni:

4.1 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!**

4.2 Autentificare

Ruta de acces:

POST /api/v1/tema/auth/login

Tip payload:

application/json

Payload:

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

Raspuns:

- **Întoarce cookie de sesiune.**

Erori tratate:

- **Întoarce un mesaj de eroare dacă credențialele nu se potrivesc!**

4.3 Cerere de acces în bibliotecă

Ruta de acces:

GET /api/v1/tema/library/access

Observații adiționale:

- **Trebuie să demonstrați că sunteți autentificați!**

Raspuns:

- **Întoarce un token JWT, care demonstrează accesul la bibliotecă.**

Erori tratate:

- **Întoarce un mesaj de eroare dacă nu demonstrați că sunteți autentificați**

4.4 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!**

Raspuns:

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!**

4.5 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`

Raspuns:

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!**

4.6 Adaugarea 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!**

4.7 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!**

4.8 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!**

Token JWT

Tokenurile JWT sunt o altă modalitate de transmitere a informațiilor dintre un client și un server. Informațiile sunt codificate binar și semnate pentru verificarea integrității. Astfel se asigură că un potențial atacator nu poate modifica informațiile 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:

[1] <https://jwt.io/introduction>

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/>

Parsare JSON

Pentru a parsă răspunsurile primite de la server, puteți (și e recomandat) să folosiți o bibliotecă. Vă sugerăm *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>

5. 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 ROSU!**

Comenzile sunt următoarele:

- **register** - efectuează înregistrarea. 1p

Oferă prompt pentru username și password.

register

username=**something**

password=**something**

- **login** - efectuează autentificarea. 1p

Oferă prompt pentru username și password.

login

username=**something**

password=**something**

- **enter_library** - cere acces în bibliotecă. 1p

enter_library

- **get_books** - cere toate cărțile de pe server. 2p

get_books

- **get_book** - cere informație despre o carte. 1p

Oferă prompt pentru id.

get_book

id=**10**

- ***add_book*** - adaugă o carte.

2p

Oferă prompt pentru *title*, *author*, *genre*, *publisher*, *page_count*.

add_book

title=**testbook**

author=**student**

genre=**comedy**

publisher=**PCom**

page_count=**10**

- ***delete_book*** - șterge o carte.

1p

Oferă prompt pentru id.

delete_book

id=**10**

- ***logout*** - efectuează logout

0.5p

logout

- ***exit*** - se închide programul

0.5p

exit

5.1 Exemplu sesiune

Comenzile vor fi testate in flux. Exemplele de mai jos nu reprezinta o lista exhaustiva a cazurilor care vor fi testate pentru corectarea temei.

Exemplu

1. Login

login

username=**test**

password=**test**

Cum aceasta comanda este prima din flux, utilizatorul “*test*” cel mai probabil nu exista. Este datoria clientului sa primească răspunsul de la server și să informeze utilizatorul de acest fapt.

2. Register

register

username=**test**

password=**test**

Cum utilizatorul “*test*” nu exista, aceasta comanda va avea succes. Puteți afișa un mesaj de succes precum “200 - OK” SAU “200 - OK - Utilizator înregistrat cu succes!”. Formatul este la latitudinea voastra.

3. Get book

get_book

id=**10**

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

4. Login

login

username=**test**

password=**test2**

In acest moment, utilizatorul nu a executat o cerere de login cu parola incorecta. Apelul va eșua. Clientul va primi răspunsuri corespunzătoare care trebuie sa informeze si 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 există!”.

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)

6. Sistem de punctare

Punctarea se efectuează individual, pentru fiecare comandă realizată cu succes într-un flux de comenzi (vezi 5.1).

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 ofera 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.

7. 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 **Nume_Prenume_Grupa_Tema3PC**. Formatul arhivei trebuie să fie .zip.

8. Mențiuni

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.

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 10.

Numele comenzilor trebuie respectat întocmai cum este precizat în enunț. Scrierea comenzilor cu alt nume duce către depunțare 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 depunțarea totală a temei.

Lipsa README duce la depunțarea totală a temei.

9. FAQ

Q: Cum demonstrez ca utilizatorul are acces la biblioteca?

A: Un utilizator are acces la biblioteca dacă a executat un apel "access". În urma apelului ar trebui sa fie prezent un token JWT. Acest token este dovada accesului la biblioteca. Voi trebuie sa va bazati 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.

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

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

Q: Va trebui sa testam, dacă se introduce pentru page_count altceva în afara de un număr?

A: Da! Validarea datelor este responsabilitatea voastra.

Q: La username și la password este permis sa fie spații?

A: Nu!

Q: În nume sau titluri de cărți pot să apară și numere, si spatii, si cifre?

A: Da! Validarea datelor este responsabilitatea voastra.