PROTOCOALE DE COMUNICAŢIE
Tema 4 Client Web. Comunicaţie cu REST API.

Deadline soft: 30.05.2024

Responsabili Temă: *Florin-Alexandru STANCU, Ioana-Alina TUDORACHE,Corneliu ROTARI, Alin-Călin DUTU, Răzvan POPA, Cătălin LEORDEANU*

1. Prezentare generală

Un procent foarte mare din aplicaţiile moderne este reprezentat de aplicaţii web. Multe dintre aceste aplicaţii web, în ciuda complexităţii pe care o au, se bazează pe modelul clasic "client-server". Pentru a înţelege cum funcţionează modelul client-server în contextul web modern, este nevoie să înţelegeţi cum funcţionează protocolul HTTP.

2. Obiective

- înţelegerea mecanismelor de comunicare prin HTTP
- interacţiunea cu un REST API
- înţelegerea conceptelor des folosite în web precum JSON, sesiune, JWT
- utilizarea unor biblioteci externe pentru manipularea obiectelor JSON REST API.

3. Descriere generală

În practica, 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 in C++ care interacționează cu un REST API expus de un server prin protocolul HTTP.

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

Serverul (infrastructură existentă) va expune un API (Application Programmable Interface) de tip REST (Representational State Transfer). Puteţi să vă gândiţi la el ca la o cutie neagră ce primește o serie de intrări reprezentate de rute HTTP. În urma cererilor HTTP, serverul efectuează o acţiune. În contextul temei, serverul simulează o bibliotecă online (de cărți) şi este deja complet implementat.
Clientul (pe care îl veți construi) este un program scris în C/C++ care acceptă comenzi de la tastatură (stdin) şi trimite, în funcţie de comandă, cereri către server. Scopul lui este de a funcţiona ca o interfaţă în linia de comandă (CLI) cu biblioteca virtuală.

4. Descriere Server

Datele de conectare

HOST: 34.246.184.49 **PORT**: 8080

Mod de funcţionare

Serverul va permite efectuarea următoarelor acţiuni:

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 credenţialele nu se potrivesc!**

4.3 Cerere de acces în bibliotecă

Ruta de acces:
GET /api/v1/tema/library/access

Observații adiţionale:

- **Trebuie să demonstraţi că sunteţi autentificaţi!**

Raspuns:

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

Erori tratate:

- **Întoarce un mesaj de eroare dacă nu demonstraţi că sunteţi autentificaţi**

4.4 Vizualizarea informaţiilor sumare despre toate cărţile

Ruta de acces:
GET /api/v1/tema/library/books
Observații adiţionale:

- **Trebuie să demonstraţi că aveţi acces la bibliotecă!**

Raspuns:
Întoarce o listă de obiecte json:

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

Erori tratate:

- **Întoarce un mesaj de eroare dacă nu demonstraţi că aveţi acces la bibliotecă!**

4.5 Vizualizarea detaliilor despre o carte

Ruta de acces:
GET /api/v1/tema/library/books/:bookId.

Observații adiționale:

- **Trebuie să demonstraţi că aveţi 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 demonstraţi că aveţi acces la bibliotecă!**
- **Întoarce un mesaj de eroare dacă id-ul pentru care efectuaţi cererea este invalid!**

4.6 Adaugarea unei cărţi

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!**

Testare server

Pentru a interacţiona neprogramatic cu serverul, puteţi folosi utilitare ce simulează clienţii HTTP, precum *Postman* [1], *Insomnia* [2] sau chiar clienţi scrişi de mână în alte limbaje de programare.

[1] **https://www.postman.com/**
[2] **https://insomnia.rest/**

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 parsa răspunsurile primite de la server, puteţi (şi e recomandat) să folosiţi o bibliotecă. Vă sugerăm parson [1] pentru C sau nlohmann [2] pentru C++, însă puteţi folosi orice (inclusiv o soluţie proprie), justificând alegerea în README.

[1] **https://github.com/kgabis/parson**
[2] **https://github.com/nlohmann/json**

Token JWT

Tokenurile JWT sunt o altă modalitate de transmitere a informaţiilor dintre un client şi un server. Informaţiile sunt codificate binar şi semnate pentru verificarea integrităţii. Astfel se asigură că un potenţial atacator nu poate modifica informaţiile 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 informaţii puteţi să consultaţi documentaţia oficială:

[1] **https://jwt.io/introduction**

5. Clientul

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

Atât comenzile cât şi 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 şi password.

register
username=**something**
password=**something**

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

Oferă prompt pentru username şi password.

login
username=**something**
password=**something**

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

enter_library

• ***get_books*** **- cere toate cărţile de pe server.** **2p**

get_books

• ***get_book*** **- cere informaţie 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*** **- şterge 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**

In 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 exista clientul va primi un răspuns corespunzător ce trebuie sa informeze si utilizatoru 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 sa 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 funcţională dacă, prin introducerea ei se trimite cererea bună către server şi se afișează răspunsul acestuia (de succes sau de eroare). Formatul răspunsurilor este la latitudinea voastra, cat timp ofera suficienta informatie legata de statusul comenzii. In plus, trebuie ținut cont de accesul la biblioteca. De exemplu, dacă în urma apelului logout, utilizatorul încă mai are acces la biblioteca, comanda logout nu va fi considerata functionala.

Comenzile care trimit date la server (cele care execută POST sau GET şi DELETE pe id-uri) sunt considerate funcţionale dacă reuşesc să trimită cu succes informaţia bună la server. De exemplu, o comandă de autentificare care parsează prost de la tastatură username sau password, dar totuşi trimite către server informaţia preluată greşit şi afişează întotdeauna răspunsul de eroare al serverului va fi considerată o comandă nefuncţională, deci nu va fi punctată.

7. Checker

Pentru a verifica ușor corectitudinea, aveți la dispoziție un checker scris în Python pe GitLab: 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).

8. Arhiva

Arhiva temei trebuie să conţină sursele de cod, un Makefile şi un Readme prin care să explicaţi implementarea soluţiei voastre. Trebuie justificată şi explicată şi utilizarea bibliotecii de parsare JSON pe care aţi ales să o folosiţi.

Arhiva va avea numele **Grupa_Nume_ Prenume _Tema4PC**. Formatul arhivei trebuie să fie .zip.

9. Mentiuni

IMPORTANT! A fost implementat în server şi un mecanism de protecţie pentru limitarea numărului de cereri. Dacă trimiteţi prea multe cereri prea rapid veţi primi un răspuns de forma *"Too many requests, please try again later."* Nu este nevoie să ţineţi 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 şi token-urile trebuie puse în cerere şi extrase din răspuns automat. Hardcodarea acestora va duce la anularea punctajului pentru cerinţa în care au fost utilizate. Pentru mai multe detalii legate de modul de funcţionare consultaţi Laboratorul 10.

Numele comenzilor trebuie respectat întocmai cum este precizat în enunţ. **Scrierea comenzilor cu alt nume duce către depunctare pentru comanda respectivă**.

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

Formatul comenzilor trebuie respectat întocmai cum este precizat în enunţ. Fiecare comandă, respectiv câmp şi 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 depunctarea totală a temei.

Lipsa README duce la depunctarea totală a temei.

10. 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. Daca vreti sa integrati alta biblioteca externa, ar trebui sa cereti o aprobare pe forumul temei.

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.