PROTOCOALE DE COMUNICAŢIE
Tema 3 Client Web. Comunicaţie cu REST API.

Deadline hard: 30 Mai 2021

Responsabili Temă: **Andrei Stanca, Andrei Ilie, Andrei Raţă**
Irina-Nicoleta Ungureanu, Cătălin Leordeanu

1 Motivaţie

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 Obiectivele temei de casă

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

- 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

Tema urmăreşte implementarea unui client web care să interacţioneze cu un server. Deşi tehnologia de-facto pentru realizarea clienţilor 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). Puteţi să vă gândiţi 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 acţiune. În contextul temei, serverul simulează o bibliotecă online şi este deja complet implementat.
- **Clientul** este un program scris în C/C++ care acceptă comenzi de la tastatură şi trimite, în funcţie de comandă, cereri către server. Scopul lui este de a funcţiona ca o interfaţă (UI) cu biblioteca virtuală.

4 Interacţiunea cu Serverul

Datele de conectare

HOST: 34.118.48.238 si **PORT**: 8080

Mod de funcţionare

Serverul va permite efectuarea urmatoarelor acţiuni:

Înregistrarea unui cont

- Ruta de acces: **POST** /api/v1/tema/auth/register
- Tip payload: **application/json**
- Payload:

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

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

Autentificare

- Ruta de acces: **POST** /api/v1/tema/auth/login
- Tip payload: **application/json**
- Payload:

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

- Întoarce cookie de sesiune
- Întoarce un mesaj de eroare dacă credenţialele nu se potrivesc

Cerere de acces in bibliotecă

- Ruta de acces: **GET** /api/v1/tema/library/access
- Trebuie sa demonstraţi că sunteţi autentificaţi
- Întoarce un token JWT, care demonstrează accesul la bibliotecă
- Întoarce un mesaj de eroare dacă nu demonstraţi că sunteţi autentificaţi

Vizualizarea informaţiilor sumare despre toate cărţile

- Ruta de acces: **GET** /api/v1/tema/library/books
- Trebuie sa demonstraţi că aveţi acces la bibliotecă
- Întoarce o listă de obiecte json:

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

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

Vizualizarea detaliilor despre o carte

- Ruta de acces: **GET** /api/v1/tema/library/books/:bookId. În loc de *:bookId* este un id de carte efectiv (ex: /api/v1/tema/library/books/1)
- Trebuie sa demonstraţi că aveţi acces la bibliotecă
- Întoarce un obiect json:

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

- 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

Adaugarea unei cărţi

- Ruta de acces: **POST** /api/v1/tema/library/books
- Tip payload: **application/json**
- Trebuie să demonstraţi că aveţi acces la bibliotecă
- Payload:

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

- Întoarce un mesaj de eroare dacă nu demonstraţi că aveţi acces la bibliotecă
- Întoarce un mesaj de eroare dacă informaţiile introduse sunt incomplete sau nu respectă formatarea

Ştergerea unei cărţi

- Ruta de acces: **DELETE** /api/v1/tema/library/books/:bookId. În loc de *:bookId* este un id de carte efectiv (ex: /api/v1/tema/library/books/1)
- Trebuie să demonstraţi că aveţi acces la bibliotecă
- 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

Logout

- Ruta de acces: **GET** /api/v1/tema/auth/logout
- Trebuie să demonstraţi că sunteţi autentificaţi
- Întoarce un mesaj de eroare dacă nu demonstraţi că sunteţi autentificaţi

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: https://jwt.io/introduction

Testare server

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

Parsare JSON

Pentru a parsa răspunsurile primite de la server, puteţi (şi e recomandat) să folosiţi o bibliotecă. Vă sugerăm *parson*[2] pentru C sau *nlohmann*[3] pentru C++, însă puteţi folosi orice (inclusiv o soluţie proprie), justificând alegerea în README.

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 repeta până la introducerea comenzii *exit*.

Atât comenzile cât şi câmpurile împreună cu valorile aferente se scriu pe linii separate! Datele introduse de catre utilizator de la tastatura sunt cele marcate cu ROSU!

Comenzile sunt următoarele:

- **register** - efectuează înregistrare. Oferă prompt pentru *username* şi *password* **1p**

  register
  username=something
  password=something

- **login** - efectuează autentificarea. Oferă prompt pentru *username* şi *password* **1p**

  login
  username=something
  password=something

[1] https://www.getpostman.com/
[2] https://github.com/kgabis/parson
[3] https://github.com/nlohmann/json

- **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. Oferă prompt pentru *id* **1p**

  get_book
  id=10

- **add_book** - adaugă o carte. Oferă prompt pentru *title*, *author*, *genre*, *publisher*, *page_count* **2p**

  add_book
  title=Testbook
  author=student
  genre=comedy
  publisher=PCom
  page_count=10

- **delete_book** - şterge o carte. Oferă prompt pentru *id* **1p**

  delete_book
  id=10

- **logout** - efectuează logout **0.5p**

  logout

- **exit** - se închide programul **0.5p**

  exit

6 Sistem de punctare

Punctarea se efectuează individual, pentru fiecare comandă realizată cu succes.

O comandă este considerată funcţională dacă, prin introducerea ei se trimite cererea bună către server şi se afisează răspunsul acestuia (de succes sau de eroare).

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

8 Menţiuni

IMPORTANT! A fost implementat în server şi un mecanism de protecţie pentru limitarea numarului 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 adaugaţi un caz special în implementare.

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.