PROTOCOALE DE COMUNICAȚIE
Tema 3 Client Web. Comunicatie cu REST API.

Termen de predare: 10 Mai 2020

Responsabili Temă: **Catalin LEORDEANU, Alexandru HOGEA**
Adrian Ilie, Vlad Nastase, Andrei Stanca

1 Motivatie

Un procent foarte mare din aplicatiile moderne este reprezentat de aplicatii web. Multe dintre aceste aplicatii web, in ciuda complexitatii pe care o au, se bazeaza pe modelul clasic "client-server". Pentru a intelege cum functioneaza modelul client-server in contextul web modern, este nevoie sa intelegeti cum functioneaza protocolul HTTP.

2 Obiectivele temei de casă

Scopul temei este un client scris in C/C++ care sa interactioenze cu un REST API. Obiectivele pe care le urmarim sunt:

- intelegerea mecanismelor de comunicare prin HTTP
- interactiunea cu un *REST API*
- intelegerea conceptelor des folosite in web precum *JSON*, *sesiune*, *JWT*
- utilizarea unor biblioteci externe pentru manipularea obiectelor *JSON REST API*.

3 Descriere generală

Tema urmareste implementarea unui client web care sa interactioneze cu un server. Desi tehnologia de-facto pentru realizarea clientilor web este triada *html, css, javascript*, pentru sedimentarea conceptelor am ales sa folosim C/C++, pentru a ne apropia cat mai mult de protocol.

- **Serverul** expune un API (Application Programmable Interface) de tip REST (Representational State Transfer). Puteti sa va ganditi la el ca la o cutie neagra ce are expuse o serie de intrari, reprezentate de rute HTTP. In urma cererilor HTTP, serverul efectueaza o actiune. In contextul temei, serverul simuleaza o biblioteca online si este deja complet implementat.
- **Clientul** este un program scris in C/C++ care accepta comenzi de la tastatura si trimite, in functie de comanda, cereri catre server. Scopul lui este de a functiona ca o interfata (UI) cu biblioteca virtuala.

4 Interactiunea cu Serverul

Datele de conectare

HOST: `ec2-3-8-116-10.eu-west-2.compute.amazonaws.com` si **PORT**: `8080`

Mod de functionare

Serverul va permite efectuarea urmatoarelor actiuni:

Inregistrarea unui cont

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

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

- Intoarce eroare daca username-ul este deja folosit de catre cineva

Autentificare

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

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

- Intoarce cookie de sesiune
- Intoarce un mesaj de eroare daca credentialele nu se potrivesc

Cerere de acces in biblioteca

- Ruta de acces: **GET** /api/v1/tema/library/access
- Trebuie sa demonstrati ca sunteti autentificati
- Intoarce un token JWT, care demonstreaza accesul la biblioteca
- Intoarce un mesaj de eroare daca nu demonstrati ca sunteti autentificati

Vizualizarea informatiilor sumare despre toate cartile

- Ruta de acces: **GET** /api/v1/tema/library/books
- Trebuie sa demonstrati ca aveti acces la biblioteca
- Intoarce o lista de obiecte json:

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

- Intoarce un mesaj de eroare daca nu demonstrati ca aveti acces la biblioteca

Vizualizarea detaliilor despre o carte

- Ruta de acces: **GET** /api/v1/tema/library/books/:bookId. In loc de *:bookId* este un id de carte efectiv (ex: /api/v1/tema/library/books/1)
- Trebuie sa demonstrati ca aveti acces la biblioteca
- Intoarce un obiect json:

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

- Intoarce un mesaj de eroare daca nu demonstrati ca aveti acces la biblioteca
- Intoarce un mesaj de eroare daca id-ul pt. care efectuati cererea este invalid

Adaugarea unei carti

- Ruta de acces: **POST** /api/v1/tema/library/books
- Tip payload: **application/json**
- Trebuie sa demonstrati ca aveti acces la biblioteca
- Payload:

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

- Intoarce un mesaj de eroare daca nu demonstrati ca aveti acces la biblioteca
- Intoarce un mesaj de eroare daca informatiile introduse sunt incomplete sau nu respecta formatarea

Stergerea unei carti

- Ruta de acces: **DELETE** /api/v1/tema/library/books/:bookId. In loc de *:bookId* este un id de carte efectiv (ex: /api/v1/tema/library/books/1)
- Trebuie sa demonstrati ca aveti acces la biblioteca
- Intoarce un mesaj de eroare daca nu demonstrati ca aveti acces la biblioteca
- Intoarce un mesaj de eroare daca id-ul pt. care efectuati cererea este invalid

Logout

- Ruta de acces: **GET** /api/v1/tema/auth/logout
- Trebuie sa demonstrati ca sunteti autentificati
- Intoarce un mesaj de eroare daca nu demonstrati ca sunteti autentificati

Token JWT

Tokenurile JWT sunt o alta modalitate de transmitere a informatiilor dintre un client si un server. Informatiile sunt codificate binar si semnate pentru verificarea integritatii. Astfel se asigura ca un potential atacator nu poate modifica informatiile impachetate.

Pentru a trimite tokenul catre server, este necesara adaugarea acestuia in headerul **Authorization**. Valoarea tokenului trebuie sa fie prefixata de cuvantul **Bearer**.

```
Authorization: Bearer eijjkwuqioueu9182712093801293
```

Pentru mai multe informatii puteti sa consultati documentatia oficiala: `https://jwt.io/introduction`

Testare server

Pentru a interactiona neprogramatic cu serverul, puteti folosi utilitare ce simuleaza clientii HTTP, precum *Postman* [1] sau chiar clienti scrisi de mana in alte limbaje de programare.

Parsare JSON

Pentru a parsa raspunsurile primite de la server puteti (si e recomandat) sa folositi o biblioteca. Va sugeram *parson*[2] pentru C sau *nlohmann*[3] pentru C++, insa puteti folosi orice (inclusiv o solutie proprie), justificand alegerea in README.

5 Clientul

Clientul va trebui sa interpreteze comenzi **de la tastatura** pentru a putea interactiona cu serverul. In urma primirii unei comenzi, clientul va forma obiectul json (daca e cazul), va executa cererea catre server si va afisa raspunsul acestuia (de succes sau de eroare). Procesul se repeta pana la introducerea comenzii *exit*.

Atat comenzile cat si campurile impreuna cu valorile aferente se scriu pe linii separate!

Comenzile sunt urmatoarele:

- **register** - efectueaza inregistrare. Ofera prompt pentru *username* si *password* **1p**

```
register
username=something
password=something
```

- **login** - efectueaza autentificarea. Ofera prompt pentru *username* si *password* **1p**

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

```
login
username=something
password=something
```

- **enter_library** - cere acces in biblioteca **1p**

```
enter_library
```

- **get_books** - cere toate cartile de pe server **2p**

```
get_books
```

- **get_book** - cere informatie despre o carte. Ofera prompt pentru *id* **1p**

```
get_book
id=10
```

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

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

- **delete_book** - sterge o carte. Ofera prompt pentru *id* **1p**

```
delete_book
id=10
```

- **logout** - efectueaza logout **0.5p**

```
logout
```

- **exit** - se inchide programul **0.5p**

```
exit
```

6 Sistem de punctare

Punctarea se efectueaza, individual, pentru fiecare comanda realizata cu succes.

O comanda este considerata functionala daca, prin introducerea ei se trimite cererea buna catre server si se afiseaza raspunsul acestuia (de succes sau de eroare).

Comenzile care trimit date la server (cele care executa `POST` sau `GET` si `DELETE` pe id-uri) sunt considerate functionale daca reusesc sa trimita cu succes informatia buna la server. **De exemplu**, o comanda de autentificare care **parseaza prost** de la tastatura username sau password, dar totusi trimite catre server informatia preluata gresit si afiseaza intotdeauna raspunsul de eroare al serverului va fi considerata o comanada **nefunctionala**, deci nu va fi punctata.

7 Arhiva

Arhiva temei trebuie sa contina sursele de cod, un Makefile si un Readme prin care sa explicati implementarea solutiei voastre. Trebuie justificata si explicata si utilizarea bibliotecii de parsare JSON pe care ati ales sa o folositi.

Arhiva va avea numele **Nume_Prenume_Grupa_Tema3PC**. Formatul arhivei trebuie sa fie **.zip**.

8 Mentiuni

Datele, headere-le, cookie-urile si token-urile trebuie puse in cerere si extrase din raspuns **automat**. Hard-codarea acestora va duce la anularea punctajului pentru cerinta in care au fost utilizate. Pentru mai multe detalii legate de modul de functionare consultati Laboratorul 10.

Numele comenzilor trebuie respectat intocmai cum este precizat in enunt. Scrierea comenzilor cu alt nume duce catre depunctare pentru comanda respectiva.

Numele campurilor din obiecte trebuie respectat intocmai cum este precizat in enunt. Altfel, veti primi permanent erori de pe server, deci comanda va fi depunctata.

Formatul comenzilor trebuie respectat intocmai cum este precizat in enunt. Fiecare comanda, respectiv camp si valoarea sa pe linii separate. Formatul nerespectat duce la pierderea punctajului pt acea comanda.

Schema de denumire a arhivei trebuie respectata intocmai. Modificarea acesteia duce la depunctarea totala a temei.

Lipsa README duce la depunctarea totala a temei.