# PROTOCOALE DE COMUNICAȚIE
# Tema 3 Client HTTP. Comunicație cu REST API.

Termen de predare: 17 Mai 2019

Responsabili Temă: **Alexandru HOGEA, Ciprian DOBRE**
Adrian NEGRU, Adrian ILIE, Dan RADULESCU, Ionut MIHALACHE, Mihai DUMITRU

## 1 Motivație

Conceptul de REST API este un concept larg întâlnit. Acesta se bazează pe modelul clasic de client-server, serverul fiind implementat ca un black box, și este transpus peste detaliile de funcționare ale protocolului HTTP.

## 2 Obiectivele temei de casă

Scopul temei este realizarea unui client scris în C/C++, care să interacționeze cu un REST API. Obiectivele temei sunt:

- înțelegerea mecanismelor de comunicare prin HTTP
- familiarizarea cu concepte precum REST API, JSON, JWT
- utilizarea unor biblioteci externe pentru parsarea si crearea de structuri *JSON*
- dezvoltarea unei aplicații client HTTP care comunica cu un *REST API*.

## 3 Descriere generală

Tema este structurată sub forma unui *treasure hunt* si urmează paradigma deja întâlnită de client-server. Ea are două componente:

- **Serverul** de tip REST API este prima componentă. Acesta este implementat și oferit de noi pentru a fi interogat de clientul scris de voi. Serverul va avea un set de URL-uri expuse de noi, ce vor trebui apelate (programatic) din client.
- **Clientul HTTP** este a doua componentă. Acesta va fi implementat de studenți și va trebui să apeleze rutele expuse de către server folosind metode de `GET` și `POST`.

Tema are 5 etape, fiecare etapă valorând 2p. Enunțul cuprins în acest PDF vă oferă informații despre cum anume să interacționați cu serverul, despre restricțiile care vă sunt impuse în realizarea clientului și cerințele pentru etapa 1. Totuși, enunțul referitor la rezolvarea cerințelor vă va fi oferit pe măsură ce rezolvați o etapă corect. Așadar, cerințele pentru etapa X+1 vor fi disponibile după ce rezolvați etapa X. O etapă este considerată rezolvată dacă răspunsul primit de la server, în urma unei cereri `GET` sau `POST` nu conține un mesaj de eroare.

# 4 Serverul

## Mod de funcționare

Serverul este implementat sub forma unui API [1] de tip REST [2]. Acesta are o serie de URL-uri expuse pe care voi va trebui să le accesați prin cereri `HTTP` de tip `GET` sau `POST`. Dacă cererea a fost efectuată corect, serverul va întoarce un obiect de tip JSON [3], cu mai multe câmpuri, printre care enunțul pentru etapa următoare.
IP-ul serverului este: **185.118.200.35**, iar portul este: **8081**

## Structura raspunsurilor

Răspunsurile venite de la server vor avea structura clasică a unui răspuns HTTP 1.1, înseamnând:

```
Status_line
Header1
Header2
...
HeaderN
Set-Cookie: cookie1=value1 (în cazul în care aveți cookie returnat)
Set-Cookie: cookie2=value2
...
Set-Cookie: cookieN=valueN

Body
```

În body veți regăsi o structură de tip JSON sau un mesaj de eroare. În cazul în care cererea este făcută cu succes, veți primi o serie de câmpuri care vor apărea în toate răspunsurile, mai puțin după etapă 5, acestea fiind:

`enunț, url, method (dacă cererea a fost făcută bine)`

Dacă cererea nu se realizează cu succes, veți primi un mesaj de eroare descriptiv:

`Forbidden!`

Pe lângă aceste câmpuri obligatorii, în unele cazuri vor mai exista câmpurile `data` și `type`.
*SPOILER ALERT* Mai jos aveți un exemplu de răspuns de la server:

```
{
    "enunt":"Felicitări! Pentru etapa următoare va trebui să ...",
    "url": "/login",
    "method": "POST",
    "type": "application/x-www-form-urlencoded",
    "data": {
        "username": "XXX",
        "password": "XXX"
    }
}
```

Ce înseamnă fiecare câmp în parte?

- **Enunț**: Ce va trebui făcut pentru etapa următoare
- **Url**: URL-ul care va trebui apelat pentru etapa următoare

[1] https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82
[2] https://www.codecademy.com/articles/what-is-rest
[3] https://www.w3schools.com/whatis/whatis_json.asp.

- **Method**: Tipul cererii ce va trebui făcută în etapa următoare
- **Type**: În cazul în care "method" este POST, modul în care vor fi transmise datele (*x-www-form-urlencoded sau JSON*) – **poate lipsi**
- **Data**: Ce va trebui extras și folosit în cererea pentru etapa următoare. Va fi detaliat în enunțul atașat răspunsului. – **poate lipsi**

### Testare server

Pentru a vedea în prealabil răspunsul fiecărui task în parte, sau modul de funcționare, serverul se poate testa folosind utilitare ce simulează clienți HTTP, precum *Postman* [4] sau chiar clienți scriși de mână în alte limbaje de programare.

## 5 Clientul

### Mod de funcționare

Clientul va trebui să realizeze cereri de tip `GET` și `POST` pe URL-urile specificate în corpul răspunsului de pe server, respectând cerințele impuse. Toate cererile trebuie executate în aceeași rulare de program. Informațiile din răspunsul serverului, care țin de formularea cererii HTTP, vor fi extrase programatic.

### Schelet de cod

Se poate folosi absolut orice din laboratorele de protocoale. Atenție totuși că laboratorul de HTTP este doar un punct de plecare, nefiind implementată logica de parsare de răspuns și de completare automată a anumitor câmpuri din cerere.

### Restricții

Pentru a putea folosi datele din răspunsul serverului, este nevoie ca voi, utilizând o bibliotecă de parsare de JSON precum *Parson* [5], să extrageți aceste date. Este nevoie să faceți acest lucru, deoarece hardcodarea anumitor părți în cererea HTTP este interzisă. Lista următoare descrie părțile ce trebuie introduse programatic în cerere:

- **URL** - extras din câmpul "url", folosit în prima linie din cererea HTTP
- **Method** - extras din câmpul "metodă", folosit în prima linie din cererea HTTP
- **Content-Type** - dacă este cazul (header necesar pentru cerere de tip POST), tipul este extras din câmpul "type"
- **Cookie** - dacă este cazul (serverul returnează cookies), cookie-urile sunt extrase din antetul răspunsului de pe server (Set-Cookie: "cookie1")
- Parametri de cerere (pentru GET) sau date ce trebuie trimise (pentru POST), precizate în câmpul "data". Trebuie extras atât numele parametrilor, cât și valoarea lor.

De ce este interzisă hardcodarea? În viața reală, clienții de HTTP preiau automat anumite date din răspunsurile venite de pe servere sau populează automat câmpuri. Gândiți-vă cum ar fi dacă ar sta cineva în spatele browserului vostru să scrie manual antete și cookie-uri.

### Etapa 1

Pentru realizarea primei etape, trebuie să efectuați o cere de tip `GET` pe URL-ul `/task1/start`

[4] https://www.getpostman.com/
[5] https://github.com/kgabis/parson

# 6 Punctare

Fiecare etapă implementată corect are 2 puncte. Punctajele se vor acorda doar dacă soluția care a dus la rezolvarea etapei este implementată conform cerințelor și restricțiilor.

De exemplu, dacă ajungeți în etapa 3, dar cererea compusă la etapa 2 are informații care ar fi putut fi preluate automat din răspunsul primit de pe server, scrise de mână, nu veți primi punctajul pentru acea etapă.

Tema se va testa rulând clientul implementat de voi.