PROTOCOALE DE COMUNICAȚIE : LABORATOR 11

Protocolul HTTP

Responsabili: Ciprian DOBRE, Alexandru HOGEA

Cuprins

Obiective

In urma parcurgerii acestui laborator studentul va fi capabil sa:

- Inteleaga functionalitatea protocolului HTTP.
- Sa inteleaga rolul entitatilor implicate intr-un schimb de mesaje HTTP.
- Sa scrie un client simplu de HTTP si sa interactioneze cu un API de tip REST.

Ce este HTTP?

HTTP (HyperText Transfer Protocol) este protocolul standard folosit in Web. HTTP este un protocol ASCII simplu. O interactiune HTTP consta dintr-o cerere [1] de tip text, urmata de un raspuns [2] care consta in date transferate, cele mai uzuale formate de date fiind MIME si JSON [3]. Se poate spune ca protocolul HTTP este format din multimea cererilor (uzual GET, POST, PUT, DELETE) de la clienti catre servere si din multimea raspunsurilor returnate de acestea.

[1] https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html
[2] https://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html
[3] https://www.json.org/

HTTP opereaza automat pe portul 80, in timp ce HTTPS, pe portul 443. Cu toate aceasta, in etapele de dezvoltare, serverele sunt puse si pe alte porturi, in principal in intervalul 3000-3100 sau 8000-8100 (porturi uzuale sunt 3000, 8000, 8080, etc...)

HTTP cunoaste mai multe versiuni succesive si este in continua evolutie; In momentul de fata, HTTP/2.0 este ultima iteratie peste protocolul de HTTP, insa HTTP/1.1 este inca cea mai folosita varianta.

Cum arata un mesaj HTTP?

Formatul unui mesaj HTTP este:

```
<linia initiala, diferita pentru cerere si raspuns >\r\n
Header1: value1\r\n
Header2: value2\r\n
Header3: value3\r\n
\r\n
<corp>
```

Linia initiala este diferita pentru cerere si raspuns. O cerere uzuala de tip GET cu 2 cookie-uri arata astfel:

```
GET /test HTTP/1.1
Host: foo.example
Cookie: cheie1=valoare1; cheie2=valoare2
-- linie goala --
```

Cererile de tip POST pot fi de mai multe tipuri, in functie de modul in care se transmit parametri. O cerere uzuala de tip application/x-www-form-urlencoded arata in felul urmator:

! Atentie, intotdeauna trebuie specificate header-ele "Content-Type" si "Content-Length" cand se lucreaza cu POST. !

```
POST /test HTTP/1.1
Host: foo.example
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

field1=value1&field2=value2
```

Raspunsul cuprinde statusul, headere si, in unele situatii, date returnate. Mai jos aveti un exemplu de raspuns cu 5 headere, 2 cookie-uri si "Hello world!" ca data returnata de la server.

```
HTTP/1.1 200 OK
Date: Sun, 10 Oct 2010 23:26:07 GMT
Server: Apache/2.2.8 (Ubuntu) mod_ssl/2.2.8 OpenSSL/0.9.8g
Content-Length: 12
Connection: close
Content-Type: text/html
Set-Cookie: laborator=protocoale
Set-Cookie: dificultate=usor

Hello world!
```

Partile implicate intr-o conexiune HTTP

Intotdeauna o cerere de tip HTTP are o sursa si o destinatie. In mod normal, o cerere este trimisa de catre un client, catre un server, pentru a realiza o actiune.

Serverul HTTP

Serverul este un program, avand ca identitate IP-ul (sau numele de host) si un port, care asteapta cereri pe rute (URL-uri) definite si, pe baza acestora realizeaza actiuni. Actiunile pot varia, de la interactiuni cu o baza de date, la returnat fisiere HTML Asa cum am spus anterior, cele mai folosite actiuni ale protocolului HTTP sunt:

- GET - cere resurse de pe server
- POST - adauga resursa pe server
- PUT - modifica resursa pe server
- DELETE - sterge resursa de pe server

Clientul HTTP

Clientul este un program ce inainteaza cereri catre un server, pe baza "contractului" oferit de catre server, contract reprezentat prin documentatia serverului. Documentatia cuprinde, de obicei, rutele disponibile, metodele prin care se acceseaza acele rute si datele returnate. Cel mai comun client de HTTP este browser-ul, care efectueaza GET pentru a accesa pagini si POST pentru a inregistra formulare.

Sesiunea HTTP

Cererile realizate catre un server sunt *stateless* insemnand ca un server nu stie sa discearna intre doua cereri succesive. Pentru acest lucru s-a introdus conceptul de sesiune, prin care serverul poate salva informatii despre fiecare client in parte. Probabil sunteti familiari cu conceptul de *cookies*.

Cookies reprezinta o metoda, ilustrata si mai sus, prin care serverul trimite clientului date de tip "cheie-valoare", pe care apoi clientul poate sa le includa in cererile HTTP ulterioare. Astfel, serverul, pe baza datelor din cookies, este capabil sa retina informatii despre clienti si intre mai multe cereri. Aceste cookies nu retin doar date, ci pot avea mai multe optiuni, ce tin de durata de viata a cookie-ului, tipul de encoding, samd... [4]

O alta metoda, foarte raspandita in serverele de tip Web API este JWT Token, care deserveste si rolul de autorizare.

Tabelul 1: Schimburi de mesaje pentru site de cumparaturi online

Sens	Headere	Semnificatie
C ->S	POST /blah/login HTTP/1.1 [form data]	Utilizatorul se autentifica
S ->C	HTTP/1.1 200 OK Set-Cookie: Customer="USER";Version="1"; Path="/blah"	Salvare utilizator
C ->S	POST /blah/pickitem HTTP/1.1 Cookie: $Version="1"; Customer="USER"; Path="/blah" [form data]	Selectare produs
S ->C	HTTP/1.1 200 OK Set-Cookie: Part_Number="SmartTV";Version="1";Path="/blah"	Vizualizare cos cumparaturi

[4] https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

Aplicație

1. Conectare cu un server ce returneaza o pagina

Scrieti un program client care se conecteaza la un server web (precum cs.curs.pub.ro) si cere o pagina folosind protocolul HTTP.

Indicatii: Programul se va conecta pe TCP la adresa citita si va trimite o linie text continand:

```
GET /file_name HTTP/1.1\r\n
Host: host\r\n
\r\n
```

Atentie: numele trebuie specificat in format absolute; cererea e terminate de trimiterea unei linii goale. Ulterior aplicatia va citi raspunsul (pagina html intoarsa).

2. Conectare cu un API

Scrieti un program client care se conecteaza la un server web de tip API, ce are mai multe rute expuse, pentru a realiza urmatoarele cerinte:

- Vizualizarea unei colectii de filme (GET 185.118.200.37:8081/videos)
- Adaugarea unui film in colectie (POST 185.118.200.37:8081/videos). Filmul este definit de doua campuri, `id` si `name`.
- Autentificarea pe un server (POST 185.118.200.37:8081/weather/login). Autentificarea se face prin doua campuri, `username` si `password`. Dupa ce va veti autentifica, veti primi un cookie.
- (Bonus) Cererea unei chei de pe server folosind cookie-ul primit anterior (POST 185.118.200.37:8081/weather/key)

Dupa ce realizati ultima cerinta, veti primi un nou URL si o cheie si, pe baza unei cereri de tip GET pe acel URL, veti obtine starea vremii in Bucuresti. Cererea va fi transmisa catre platforma Open Weather Map [5]

Datele transmise in POST vor fi de tip *form-data*, deci **Content-Type** va trebui sa fie **application/x-www-form-urlencoded**

[5] https://openweathermap.org/