Ministerul Educaţiei al Republicii Moldova Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatica si Microelectronica Departamentul Ingineria Software si Automatica

**RAPORT**

Lucrarea de laborator nr.3 *Programarea in retea*

## *Tema:* Comunicarea in Web: Protocol si aplicatie client HTTP

A efectuat: st.gr. SI-171 , Buga Grigore

A verificat:

asist. univ. Cebotari Daria

Chisinau 2020

Scopul lucrarii:

Crearea unei aplicații client HTTP

Cererile HTTP să fie făcute prin proxy

Să se utilizeze expresiile regulate

Sarcina:

Am utilizat serverul reqres.in pentru a testa metodele HTTP

Functionalitatea fiecarei metode in programul implementat:

Metoda GET - returneaza o pagina de utilizatori

Metoda PUT - adauga un nou utilizator

Metoda DELETE - sterge un utilizator

Metoda POST - autentificare cu succes

Metoda PATCH - actualizarea datelor despre un utilizator

Cod sursa fisierul HttpMetods.java

import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.URL;

import javax.net.ssl.HttpsURLConnection;

public class HttpMethods {

private final String USER\_AGENT = "Mozilla/5.0";

public static void main(String[] args) throws Exception {

HttpMethods http = new HttpMethods();

System.out.println("Http GET Method");

http.sendGet();

System.out.println("\nHttp POST Method");

http.sendPost();

System.out.println("\nHttp PUT Method");

http.sendPut();

System.out.println("\nHttp DELETE Method");

http.sendDelete();

System.out.println("\nHttp PATCH Method");

http.sendPatch();

}

//Http Get Request (page 2 of users)

private void sendGet() throws Exception {

String url = "https://reqres.in/api/users?page=2";

URL obj = new URL(url);

HttpURLConnection con = (HttpURLConnection) obj.openConnection();

//Setting request method

con.setRequestMethod("GET");

//Request header

con.setRequestProperty("User-Agent", USER\_AGENT);

int responseCode = con.getResponseCode();

System.out.println("\n'GET' URL: " + url);

System.out.println("Response Code: " + responseCode);

BufferedReader in = new BufferedReader(

new InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

//Result

System.out.println(response.toString());

}

//Http Post Request (succesful login)

private void sendPost() throws Exception {

String url = "https://reqres.in/api/login";

URL obj = new URL(url);

HttpsURLConnection con = (HttpsURLConnection) obj.openConnection();

//Request header

con.setRequestMethod("POST");

con.setRequestProperty("User-Agent", USER\_AGENT);

con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");

String urlParameters = "email=eve.holt@reqres.in&password=cityslicka";

//Sending Post Request

con.setDoOutput(true);

DataOutputStream wr = new DataOutputStream(con.getOutputStream());

wr.writeBytes(urlParameters);

wr.flush();

wr.close();

int responseCode = con.getResponseCode();

System.out.println("\n'POST' URL: " + url);

System.out.println("'POST' parameters: " + urlParameters);

System.out.println("Response Code: " + responseCode);

BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

//Result

System.out.println(response.toString());

}

//Http Put Request (user insert)

private void sendPut() throws Exception {

String url = "https://reqres.in/api/users/4";

URL obj = new URL(url);

HttpsURLConnection con = (HttpsURLConnection) obj.openConnection();

//Request header

con.setRequestMethod("PUT");

con.setRequestProperty("User-Agent", USER\_AGENT);

con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");

String urlParameters = "name=John&job=driver";

// Send post request

con.setDoOutput(true);

DataOutputStream wr = new DataOutputStream(con.getOutputStream());

wr.writeBytes(urlParameters);

wr.flush();

wr.close();

int responseCode = con.getResponseCode();

System.out.println("\n'PUT' URL: " + url);

System.out.println("'PUT' parameters: " + urlParameters);

System.out.println("Response Code: " + responseCode);

BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

//Result

System.out.println(response.toString());

}

//Http Delete Request (user delete)

private void sendDelete() throws Exception {

String url = "https://reqres.in/api/users/4";

URL obj = new URL(url);

HttpURLConnection con = (HttpURLConnection) obj.openConnection();

//Setting request method

con.setRequestMethod("DELETE");

//Request header

con.setRequestProperty("User-Agent", USER\_AGENT);

int responseCode = con.getResponseCode();

System.out.println("\n'DELETE' URL: " + url);

System.out.println("Response Code: " + responseCode);

BufferedReader in = new BufferedReader(

new InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

//Result

System.out.println(response.toString());

}

// Http Patch Request

private void sendPatch() throws Exception {

String url = "https://reqres.in/api/users/4";

URL obj = new URL(url);

HttpsURLConnection con = (HttpsURLConnection) obj.openConnection();

//Request header

con.setRequestMethod("POST");

con.setRequestProperty("X-HTTP-Method-Override", "PATCH");

con.setRequestProperty("User-Agent", USER\_AGENT);

con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");

String urlParameters = "name=John&job=driver";

//Sending Post Request

con.setDoOutput(true);

DataOutputStream wr = new DataOutputStream(con.getOutputStream());

wr.writeBytes(urlParameters);

wr.flush();

wr.close();

int responseCode = con.getResponseCode();

System.out.println("\n'PATCH' URL: " + url);

System.out.println("'PATCH' parameters: " + urlParameters);

System.out.println("Response Code: " + responseCode);

BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

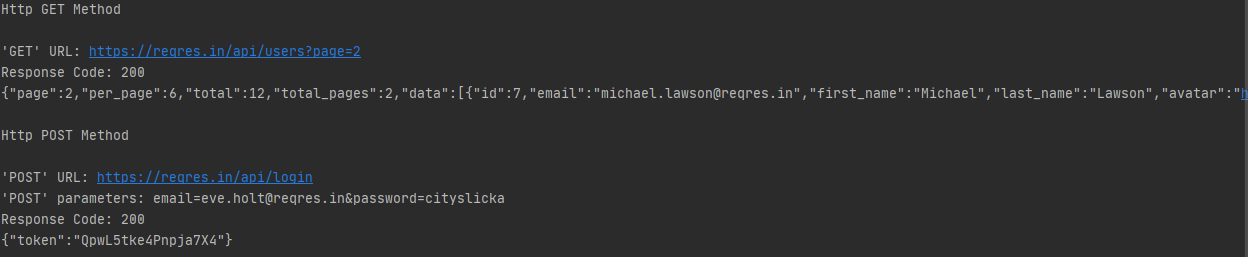
//Result

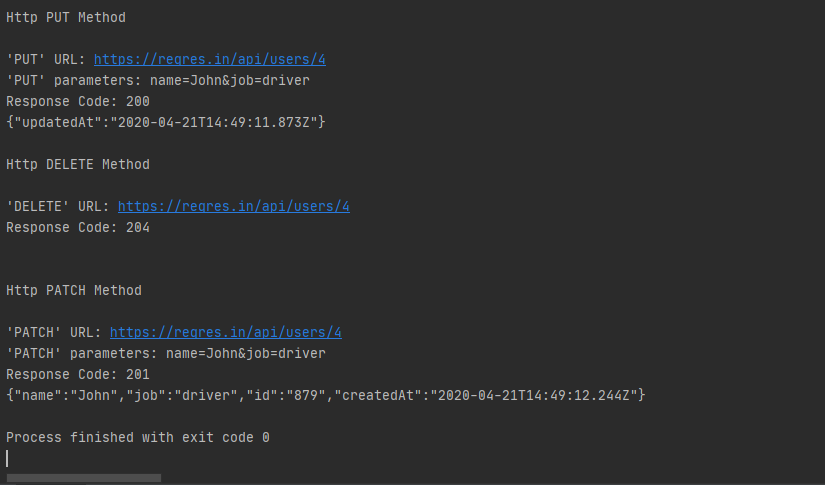
System.out.println(response.toString());

}

}

Rezultate executii:





**Intrebări la apărarea laboratorului:**

• Cum este formatat corpul unei cereri HTTP pentru o cerere HTTP de tip POST ?

POST /page.html HTTP/1.1

Host: www.server.com

attribute1=value1&...&attributen=valuen

• De unde știe un client HTTP ce tip de conținut trimite serverul HTTP ?

Dupa antetul mesajului:

-text/plain

-text/html

-text/css

-image/png

-audio/mpegapplication/javascript

-application/json

• Cum decide un client dacă ar trebui să aibă incredere in certificatul unui server ?

Conexiunile securizate HTTP au adrese URL care încep cu „https: //”. De aceea, site-urile web folosesc din ce în ce mai mult **HTTPS (Hypertext Transfer Protocol Secure)** pentru a proteja confidențialitatea utilizatorilor lor și pentru a preveni alterarea.

• Care este problema principală cu certificatele autosemnate ?

Este posibil să se creeze astfel un certificat fără a merge la [AC](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80_%D1%81%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8), numite [auto - semnat](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D0%B7%D0%B0%D0%B2%D0%B5%D1%80%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82) ( *auto-a semnat* ). Fără a verifica certificatul într-un alt mod (de exemplu, apelarea proprietarului și verificarea certificatului de verificare a certificatului), această utilizare a HTTPS este supusă unui atac om-la-mijloc.

• Conexiunea persistentă HTTP – care sunt principalele beneficii ?

-Prin deschiderea și închiderea mai puține conexiuni TCP, timpul CPU este economisit în routere și gazde (clienți, servere, proxy, gateway, tunele sau cache) și memoria folosită pentru controlul protocolului TCP blocurile pot fi salvate în gazde.

-Cererile și răspunsurile HTTP pot fi canalizate pe o conexiune.Pipelining permite unui client să facă mai multe cereri fără așteptând fiecare răspuns, permițând o conexiune TCP unică să fie utilizat mult mai eficient, cu un timp scurs mult mai mic.

-Congestionarea rețelei este redusă prin reducerea numărului de pachete cauzate de TCP se deschide și de a permite TCP timp suficient pentru determina starea de congestionare a retelei.

-Întârzierea la cererile ulterioare este redusă, deoarece nu există timp cheltuit în conexiunea TCP deschizând strângerea de mână.

• Ce este negocierea conținutului in HTTP și cum are loc ?

HTTP are prevederi pentru mai multe mecanisme pentru „negocierea conținutului” - procesul de selectare a celei mai bune reprezentări pentru un răspuns dat atunci când există mai multe reprezentări disponibile. Orice răspuns care conține o entitate-organism poate fi supus negocierii, inclusiv răspunsurile la eroare.

• Care sunt tipurile de negociere a conținutului HTTP ?

**Negociere bazată pe agent**, selecția celei mai bune reprezentări pentru un răspuns este realizată de agentul utilizator după primirea unui răspuns inițial de la serverul de origine.

**Negociere bazată pe server** se bazează pe reprezentările disponibile ale răspunsului (dimensiunile peste care poate varia; de exemplu, limbă, codare conținut etc.) și conținutul anumitor câmpuri de antet din mesajul de solicitare sau pe alte informații referitoare la solicitare (cum ar fi ca adresă de rețea a clientului).

**Negocierea transparentă** este o combinație atât de negociere condusă de server, cât și de agent. Când o memorie cache este furnizată cu o formă a listei de reprezentări disponibile ale răspunsului (ca în cazul negocierii axate de agent) și dimensiunile varianței sunt înțelese complet de cache, atunci cache-ul devine capabil să efectueze o negociere bazată pe server. a serverului de origine pentru solicitări ulterioare pe acea resursă.

• Ce este un ETag in HTTP și cum funcționează ?

**ETag** sau **etichetă de entitate**  - anteturile serviciului [protocolului HTTP / 1.1](https://ru.wikipedia.org/wiki/HTTP) , care poate fi setat de [serverul web](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) în faza generarii unui răspuns la [o](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80#%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82%D1%8B) solicitare primită de la un [client](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80#%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82%D1%8B) . Conținutul antetului ETag este un identificator a cărui valoare depinde direct de starea resursei încărcate de [client](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80#%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82%D1%8B).  În viitor, acest identificator este utilizat pentru a actualiza starea resursei descărcate la originalul său, localizat pe [serverul Web](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) . Ce se realizează prin trimiterea unei cereri către serverul [HTTP / 1.1](https://ru.wikipedia.org/wiki/HTTP) care indică identificatorul ETag ca valoare a antetului -[If-None-Match](https://tools.ietf.org/html/rfc7232#section-3.2) . După ce a găsit un astfel de antet, serverul, pe baza unei comparații a valorii sale cu starea actuală a resursei, informează clientul că copia stocată în cache-ul clientului este curentă, adică. nu este necesar să reîncărcați sau, în caz contrar, trebuie să descărcați versiunea curentă. ETag poate fi utilizat în paginile web pentru monitorizarea modificărilor și notificărilor.

• Diferența dintre protocoalele fără stare și cele cu stare. Cărui tip ii aparține HTTP ?

HTTP îi aparține tipului de protocol cu stare, aceste coduri de stare sun clasificate in 5 clase:

-1xx - erori informaționale

-2xx - răspuns reușit

-3xx – redirectări

-4xx - erori ale utilizatorilor

-5xx - erori de server

• Avantajele cheie ale HTTP/2 in comparație cu HTTP/1.1

Protocolul HTTP / 2 este binar. În comparație cu standardul anterior, metodele de împărțire a datelor în fragmente și transportul acestora între server și client au fost schimbate.

În HTTP / 2, serverul are dreptul de a trimite conținut care nu a fost încă solicitat de client. Acest lucru va permite serverului să trimită imediat fișiere suplimentare de care browserul trebuie să afișeze paginile, fără a fi nevoie ca browserul să analizeze pagina principală și să solicite completările necesare.

De asemenea, unele dintre îmbunătățiri au fost primite (în primul draft al HTTP / 2, care era o copie a specificației [SPDY](https://ru.wikipedia.org/wiki/SPDY) ) datorită multiplexării cererilor și răspunsurilor pentru a depăși problema „blocării capului de linie” a protocoalelor HTTP 1; compresia anteturilor transmise și introducerea prioritizării explicite a cererilor.

• Ce este un tip MIME, din ce constă și pentru ce se folosește ?

**Tipuri Internet Media**, de asemenea **tipuri MIME** - [tipuri de date](https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) care pot fi transmise prin [Internet](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82) utilizând standardul [MIME](https://ru.wikipedia.org/wiki/MIME). Un standart care descrie transmiterea diferitelor [tipuri de date](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_MIME-%D1%82%D0%B8%D0%BF%D0%BE%D0%B2) prin [e-mail](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D1%87%D1%82%D0%B0) , precum și, în general, pentru a codifica specificația de informații și de formatare mesaje, pentru a putea fi trimise pe [Internet](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). MIME definește mecanisme de transmitere a tuturor tipurilor de informații în interiorul datelor text (în special, prin e-mail), și anume: text în limbi pentru care sunt utilizate codificări, altele decât ASCII, și date fără text, cum ar fi imagini, muzică, filme și Program. MIME este, de asemenea, o componentă fundamentală a protocoalelor de comunicare, precum [HTTP](https://ru.wikipedia.org/wiki/HTTP) , care necesită transmiterea datelor în contextul mesajelor precum e-mail-ul, chiar dacă datele nu sunt cu adevărat e-mail.

• Care este diferența dintre GET și POST ?

-o cerere GET poate fi reținută în cache, fapt ce nu este valabil și pentru o cerere POST;

-o cerere GET rămâne în istoricul aplicației de navigare, fapt ce nu este valabil și pentru o cerere POST;

-o cerere GET poate fi reținută printre paginile Internet favorite din cadrul programului de navigare, fapt ce nu este valabil și pentru o cerere POST;

-o cerere GET impune unele restricții cu privire la lungimea (maxim 2048 caractere) și la tipul de date (doar caractere ASCII) transmise (prin URL), fapt ce nu este valabil și pentru o cerere POST;

-o cerere GET nu trebuie folosită atunci când sunt implicate informații critice (acestea fiind vizibile în URL), fapt ce nu este valabil și pentru o cerere POST;

-o cerere GET ar trebui să fie folosită doar pentru obținerea unei resurse, fapt ce nu este valabil și pentru o cerere POST.

• Care este diferența dintre PUT și POST ?

|  |  |
| --- | --- |
| PUT | POST |
| Metodă idempotentă. | Această metodă nu este idempotentă. |
| Metoda PUT este apelul atunci când trebuie să modificați o singură resursă, care face deja parte din colectarea resurselor. | Metoda POST este apelul atunci când trebuie să adăugați o resursă copil în colectarea resurselor. |
| RFC-2616 descrie că metoda PUT trimite o cerere pentru o entitate închisă stocată în URI-ul solicitat la livrare. | Această metodă solicită serverului să accepte entitatea care este inclusă în cerere. |
| Răspunsul metodei PUT poate fi pus în cache. | Nu puteți cache răspunsurile metodei PUT. |
| Sintaxa metodei PUT este PUT / questions / {question-id} | Sintaxa metodei POST este POST / întrebări |
| PUT / vi / juice / comenzi / 1234 indică faptul că actualizați o resursă care este identificată prin „1234”. | Comenzile POST / vi / juice / indică faptul că creați o nouă resursă și returnați un identificator pentru a descrie resursa. |
| Dacă trimiteți aceeași solicitare de mai multe ori, rezultatul va rămâne același. | Dacă trimiteți aceeași solicitare POST de mai multe ori, veți primi rezultate diferite. |
| PUT funcționează ca specific. | POST funcționează ca abstract. |
| Folosim interogarea UPDATE în PUT. | În metoda POST, serverul decide ce resursă URI ar trebui să aibă. |

• Care sunt metodele idempotente in HTTP și care sunt scopul lor.

Metodele GET, HEAD, PUT, OPTIONS,DELETE și TRACE sun idempotente. Această metodă poate fi numită de mai multe ori fără ca aceasta să ofere rezultate diferite. Nu contează dacă metoda a fost numită o singură dată sau de zece ori, rezultatul trebuie sa fie același.

• Cum sunt identificate resursele in protocolul HTTP ?

In HTTP, identificatorii de resurse uniformi sunt simple șiruri de format care identifică - prin nume, locație sau orice altă caracteristică - o resursă.

• Care sunt metodele sigure și nesigure in HTTP ?

-Sigure: GET, HEAD, OPTIONS și TRACE

-Nesigure: POST, PUT, DELETE și PATCH

• Pentru ce este nevoie de cURL ?

Un localizator de resurse uniform ( URL ), denumit în mod clasic o adresă web , este o referire la o [resursă web](https://en.wikipedia.org/wiki/Web_resource) care specifică locația sa pe o [rețea de calculatoare](https://en.wikipedia.org/wiki/Computer_network) și un mecanism pentru preluarea acesteia. O adresă URL este un tip specific de [identificator de resurse uniforme](https://en.wikipedia.org/wiki/Uniform_Resource_Identifier) (URI), deși multe persoane folosesc intercambiabil cei doi termeni. URL-urile apar cel mai frecvent la [paginile web de](https://en.wikipedia.org/wiki/Web_page) referință ( [http](https://en.wikipedia.org/wiki/Http) ), dar sunt de asemenea utilizate pentru transfer de fișiere ( [ftp](https://en.wikipedia.org/wiki/File_Transfer_Protocol) ), e-mail ( [mailto](https://en.wikipedia.org/wiki/Mailto) ), acces la baza de date ( [JDBC](https://en.wikipedia.org/wiki/Java_Database_Connectivity)) și multe alte aplicații.

Este utilizat ca standard pentru înregistrarea linkurilor către obiecte de pe [Internet](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82) ( link-uri [hipertext](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82) pe World Wide Web [www](https://ru.wikipedia.org/wiki/Www) ).

• Pentru ce este nevoie de HTTP Proxy?

Un server proxy (din *proxy în*[engleză](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)   - un reprezentant, autorizat; adesea doar un *proxy* , *un* server intermediar) - un server intermediar (un set de programe) din [rețelele de calculatoare](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) care acționează ca un intermediar între utilizator și serverul țintă (în acest caz, pot să știe despre mediere și nu știu ambele părți), ceea ce permite [clienților să](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) efectueze ambele solicitări indirecte (primirea și trimiterea lor printr-un server proxy) către alte servicii de rețea și să primească răspunsuri.

• Diferența dintre autentificare și autorizare.

Autentificarea si autorizarea sunt necesare daca o pagina Web trebuie sa fie accesata doar de anumiti utilizatori. Autentificarea in sine asigura ca cineva este cine pretinde ca este. De obicei, este nevoie de un nume de utilizator si o parola, dar poate include si alte metode pentru demonstrarea identitatii, precum smart card, amprente, etc. Autorizarea este calea prin care o persoana, o data ce a fost identificata (autentificata), are permisiunea sa acceseze/modifice anumite resurse. Acest lucru se face de obicei verificand daca respectiva persoana are un anumit rol care are permisiunea de a accesa resursele in cauza.

• Care sunt metodele de autentificare HTTP ?

-**Anonim**-înseamnă că nu există nici o autentificare implicată. Prin urmare, utilizatorii nu trebuie să furnizeze un nume de utilizator și o parolă pentru a se autentifica pe server. Site-urile publice publice permit accesul anonim, deoarece conținutul lor este destinat consumului public.

-**De bază** de bază este cea mai puțin sigură autentificare, deoarece permite numele de utilizator și parolele să fie trimise într-un text clar. Prin urmare, autentificarea de bază este de obicei folosită cu Secure Socket Layer (SSL), care criptează traficul pentru a împiedica hackerii să fure numele de utilizator și parola.

-**Digest** este mai bună decât autentificarea de bază, în sensul că folosește o metodă de răspuns la provocări pentru a face oarecum dificilă pentru un hacker să fure parola sau numele de utilizator.

-**NTLM** oferă o metodă mai bună de răspuns la provocare în comparație cu autentificarea Digest. Cu toate acestea, este o tehnologie învechită care nu este considerată foarte sigură.

• Modalități de identificare a utilizatorilor in HTTP

• HTTP cookies – pentru ce se folosește ?

Cookie-ul HTTP (cookie web, cookie-ul browserului) este o mică bucată de date trimise de server către browserul utilizatorului, pe care utilizatorul îl poate salva și trimite înapoi cu o nouă solicitare către acest server. Acest lucru vă permite, în special, să aflați dacă ambele solicitări provin dintr-un browser (de exemplu, pentru autentificarea unui utilizator). Stochează informații despre starea protocolului HTTP, care de la sine nu știe cum să facă acest lucru.

Cookie-urile sunt utilizate în principal pentru:

-Management Managementul sesiunii (logins, coșuri de cumpărături pentru cumpărături virtuale)  
- Personalizare (preferințe ale utilizatorului)  
- Monitorizare (urmărirea comportamentului utilizatorului)