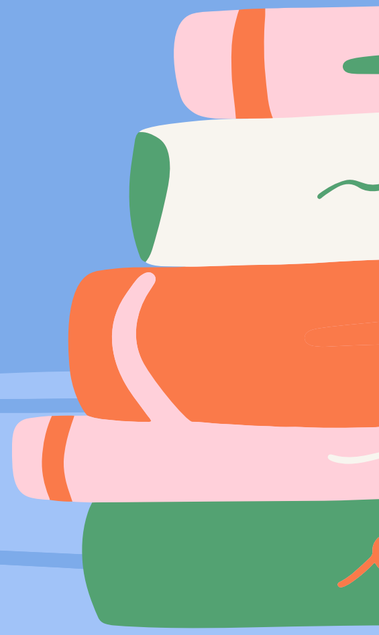
PROIECTAREA SISTEMELOR DE OPERARE- PROIECT

Mini server Web - cu suport pentru conexiuni multiple, cu thread-uri pooling

Sd. Sg. Andrei Roxana Geanina

Sd. Sg. Gruia Ștefan Bogdan



Ce este HTTP?

- HTTP (Hypertext Transfer Protocol) reprezintă un protocol la nivel aplicație care definește modalitatea de comunicatie dintre un server ce găzduiește un site cu pagini web și un client.
- Un server HTTP se mai numește și server web. Un client HTTP se mai numește și Browser..
- Protocolul HTTP folosește default portul TCP/80. Acesta este portul pe care serverul web ascultă conexiuni de la clienți.

CE REPREZINTĂ MESAJELE HTTP?

📌Orice mesaj HTTP este precedat de un HEADER care oferă informații cu privire la cererea sau răspunsul HTTP, sau cu privire la obiectul trimis în interiorul mesajului.

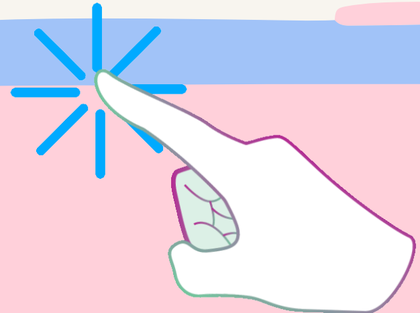
📌Protocolul HTTP prezintă 4 tipuri de headere:

- General-header: cu aplicabilitate generală, atât pentru cereri, cât și pentru răspunsuri
- Request-header: cu aplicabilitate doar pentru cereri
- Response-header: cu aplicabilitate doar pentru răspunsuri
- Entity-header: definește meta-date cu privire la obiectul din corpul mesajului sau cu privire la resursa identificată prin cerere.

CERERILE HTTP

📌Un client HTTP trimite o cerere HTTP unui server sub forma unui mesaj HTTP de tip Request care are următoarea structură:

- O linie de cerere (Request-line): Aceasta este prima linie dintr-o cerere HTTP și conține informații esențiale, cum ar fi metoda HTTP utilizată (GET, POST etc.), adresa resursei solicitate și versiunea protocolului HTTP.
- Zero sau mai multe anteturi (header fields): Acestea sunt câmpuri de antet care pot fi de tip General, Request sau Entity. Ele sunt urmate de CRLF (Carriage Return Line Feed), care marchează sfârșitul fiecărui antet.Anteturile pot include informații precum tipul de conținut acceptat, limbajul preferat, codificarea, tipul de browser etc.
- Emty line: Aceasta este o linie fără conținut, precedată doar de CRLF. Linia goală indică sfârșitul secțiunii de anteturi HTTP. După această linie, începe corpul mesajului (dacă există).
- Opțional, un corp de mesaj (message-body): Nu toate cererile HTTP au un corp de mesaj. Acesta este utilizat în special în cereri de tip POST sau PUT pentru a trimite date către server, cum ar fi formulare sau fișiere.



REQUEST-LINE

📌**Request-Line** (Linia de Cerere) este prima linie a unei cereri HTTP și conține trei componente principale:

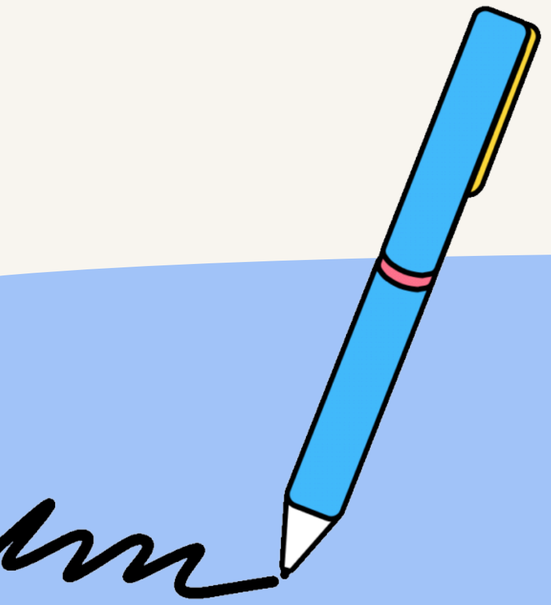
1. Metoda HTTP – un token care specifică tipul cererii (GET, POST, etc.).
2. URI-ul (Uniform Resource Identifier) – indică resursa solicitată.
3. Versiunea protocolului HTTP – de exemplu, HTTP/1.1 sau HTTP/2.

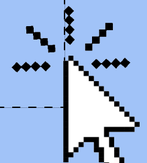
📌**Exemplu de Request-line:**

GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1

PRINCIPALELE METODE HTTP

Metodă HTTP	Descriere
GET	Această metodă este folosită pentru a prelua informații de la un anumit server, pe baza unui URI.
HEAD	Această metodă seamănă cu GET, dar transferă doar o linie de status și secțiunea de Header.
POST	Metoda POST produce transmiterea de date către server. Spre exemplu, informațiile utilizatorului, încărcarea unui fișier, etc.
PUT	Metoda PUT produce înlocuirea tuturor reprezentărilor resurselor cu un anumit conținut.
DELETE	Șterge reprezentările curente din resursele serverului, pe baza unui URI.
CONNECT	Stabilește un tunel de conexiune cu serverul.
OPTIONS	Descrie opțiuni de comunicare cu resursele.
TRACE	Folosit pentru testarea transmiterii mesajelor, generând și calea către resurse.





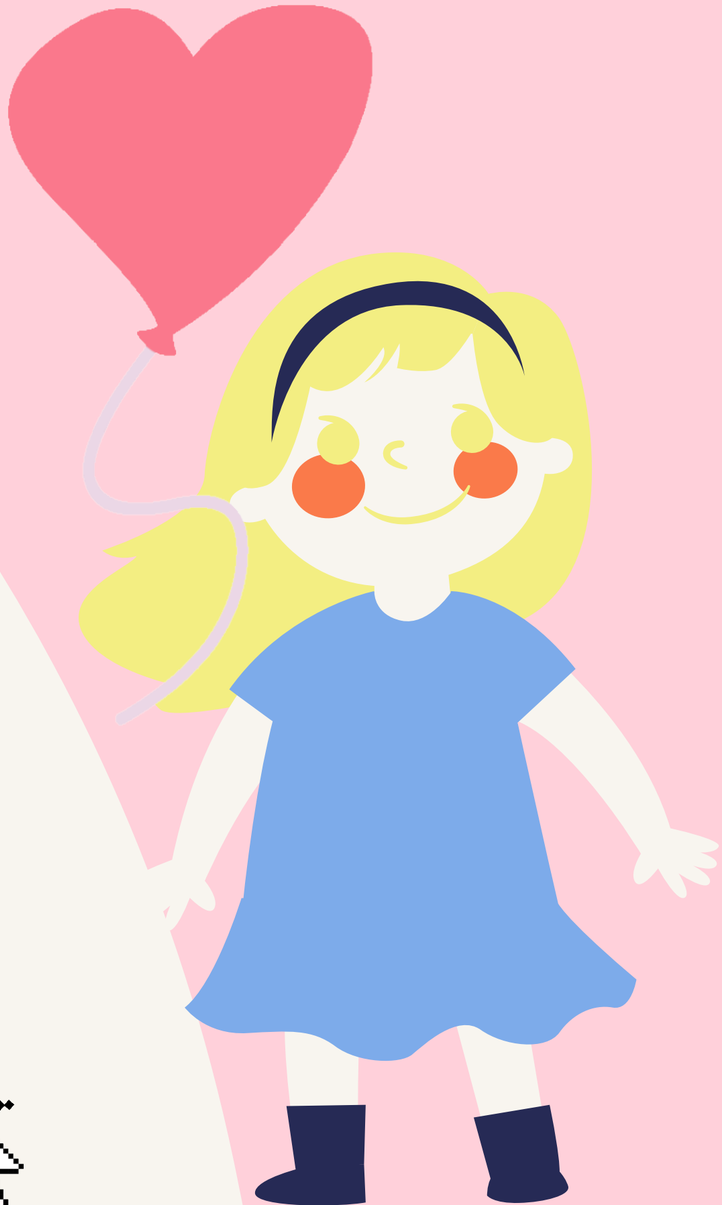
RĂSPUNSURILE HTTP ȘI CODURILE DE STARE

📌Când un client (browser, API, aplicație) face o cerere către un server web, acesta răspunde cu un cod de stare HTTP pentru a indica rezultatul cererii. Aceste coduri sunt reprezentate de un număr întreg format din trei cifre, iar prima cifră determină categoria răspunsului.

📌Există 5 clase de coduri de stare HTTP.

CODURI DE STARE HTTP

Cod	Categorie	Descriere
1xx	Informațional	Cererea a fost primită și este în procesare.
100	Continue	Serverul a primit prima parte a cererii și clientul poate continua trimiterea.
101	Switching Protocols	Serverul schimbă protocolul de comunicare conform cererii clientului.
102	Processing	Serverul a primit cererea și o procesează, dar răspunsul final nu este încă disponibil.
2xx	Succes	Cererea a fost primită, interpretată și acceptată.
200	OK	Cererea a fost procesată cu succes și serverul a returnat răspunsul așteptat.
201	Created	Resursa a fost creată cu succes pe server.
202	Accepted	Cererea a fost acceptată, dar încă nu a fost procesată complet.
204	No Content	Cererea a fost procesată cu succes, dar serverul nu returnează conținut.
3xx	Redirecționare	Pentru a completa cererea trebuie efectuate operații suplimentare.
301	Moved Permanently	Resursa a fost mutată permanent la o altă adresă URL.
302	Found	Resursa este disponibilă temporar la o altă adresă URL.
304	Not Modified	Resursa nu a fost modificată; clientul poate folosi versiunea din cache.
4xx	Eroare Client	Cererea conține sintaxă incorectă ce nu poate fi procesată.
400	Bad Request	Cererea conține sintaxă incorectă sau parametri invalidi.
401	Unauthorized	Accesul la resursă necesită autentificare.
403	Forbidden	Clientul nu are permisiunea de a accesa resursa.
404	Not Found	Resursa solicitată nu a fost găsită pe server.
405	Method Not Allowed	Metoda HTTP folosită nu este permisă pentru această resursă.
5xx	Eroare Server	Serverul nu poate îndeplini cerințele clientului, chiar dacă cererea pare validă.
500	Internal Server Error	O eroare generică a serverului, fără detalii specifice.
502	Bad Gateway	Serverul a primit un răspuns invalid de la un alt server.
503	Service Unavailable	Serverul nu este disponibil temporar (de exemplu, din cauza mentenanței).
504	Gateway Timeout	Serverul nu a primit un răspuns în timp util de la alt server.



SERVERUL HTTP

- Serverul Web găzduiește pagini Web și le pune la dispoziția clienților prin intermediul protocolului HTTP. Relația server-client se bazează, așadar, pe o aplicație care este instalată pe server și este programată să transfere paginile web găzduite.
- Utilizatorul, aflat în dreptul unui computer pe care are instalată o aplicație de tip browser solicită serverului prin intermediul unui URL o anumită pagină web; serverul rulează solicitarea și returnează un rezultat. Site-urile pot fi statice sau dinamice.
- Site-urile statice sunt bazate pe limbajele html și css.
- Site-urile dinamice au în compunere, alături de limbajul rudimentar de afișare a paginii web, și un limbaj de comunicare între serverul web și o bază de date și nu numai.

Despre proiectul nostru

Biblioteca Virtuală este o aplicație web care permite gestionarea digitală a cărților într-un mod interactiv și intuitiv. Utilizatorii pot adăuga, șterge, modifica vizualiza și descărca cărți în diferite formate. Sistemul folosește un server personalizat scris în C, care gestionează cererile HTTP și permite interacțiunea cu fișierele de stocare.

Funcționalități principale:

📚 Gestionarea colecției de cărți

- Cărțile sunt stocate într-un fișier text, oferind persistență între sesiuni.

🗑️ Ștergerea cărților

- Se poate introduce numele unei cărți pentru a o elimina definitiv.
- Dacă cartea nu este găsită, utilizatorul primește un mesaj de eroare.

📂 Descărcarea fișierelor asociate cărților

- Fiecare carte poate avea atașat un fișier PDF și o imagine de copertă.
- Utilizatorii pot descărca fișierele pentru a citi cărțile offline.

📜 Executarea de scripturi și automatizare

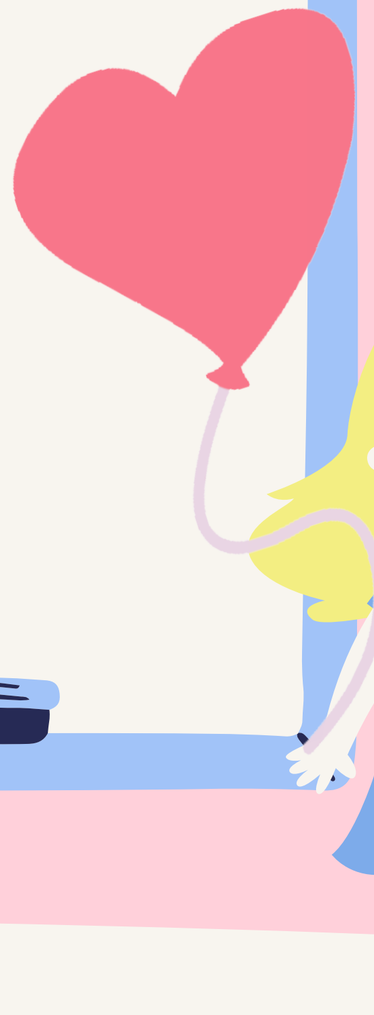
- Sistemul poate permite rularea unor scripturi asociate pentru fiecare formular de feedback.

🌍 Interacțiune web dinamică

- Utilizatorii pot modifica și șterge cărți fără reîncărcarea paginii datorită utilizării JavaScript (AJAX).
- Comunicarea dintre frontend și backend se face prin cereri HTTP.

💻 Backend robust în C

- Serverul gestionează cererile GET, DELETE, PUT, POST.
- Stocarea și procesarea datelor sunt optimizate pentru performanță ridicată.



THREAD POOL

Ce este un Thread Pool și de ce îl folosim?

📌 Un Thread Pool este un mecanism care gestionează un număr fix de fire de execuție (în cazul nostru, 4) care preiau și execută sarcini pe măsură ce sunt disponibile.

📌 În loc să creăm și să distrugem un thread pentru fiecare cerere HTTP primită, folosim un Thread Pool pentru a gestiona eficient resursele și a îmbunătăți performanța serverului.

🔗 Thread Pool în C

✅Cum am implementat Thread Pool în proiect?

- La pornirea serverului, sunt create 4 fire de execuție pentru a gestiona cererile.
- Când un client se conectează, cererea este adăugată în coada de sarcini.
- Un thread liber preia sarcina și o procesează, fără a bloca alte cereri.

✅ Eficiență și scalabilitate

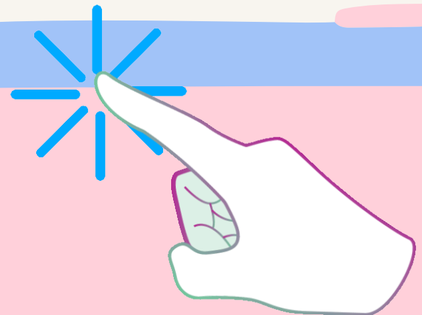
- Serverul poate răspunde mai rapid la multiple cereri simultane.
- Evităm crearea și distrugerea excesivă a firelor de execuție, economisind resurse.

INIȚIALIZAREA ȘI PORNIREA SERVERULUI (START_SERVER)

📌 Această funcție inițializează serverul și gestionează conexiunile.

🔹 Pași principali:

1. Se creează un socket TCP/IP.
2. Se setează opțiunea SO_REUSEADDR pentru reutilizarea rapidă a portului.
3. Se leagă socket-ul la portul 8080.
4. Se setează socket-ul ca non-blocking pentru a evita blocarea execuției.
5. Se creează un Thread Pool de 4 fire de execuție.
6. Acceptă conexiuni noi și distribuie sarcinile către handle_client().
7. Când serverul este oprit (SIGINT sau SIGTERM), toate firele sunt închise.



FUNCȚIA HANDLE_CLIENT() – GESTIONAREA CERERILOR HTTP

📌 Această funcție procesează cererea primită de la client.

🔹 Tipuri de cereri gestionate:

- GET / → Returnează pagina principală.
- GET /books → Afișează lista de cărți.
- GET /download/{nume_fisier} → Permite descărcarea unui fișier PDF sau a unei imagini.
- GET /feedback → Returnează pagina de feedback.
- GET /delete → Returnează pagina de ștergere a cărților.
- DELETE /books/{titlu} → Șterge cartea specificată.
- POST /feedback → Procesează feedback-ul utilizatorului și rulează un script shell.

🔹 Cum funcționează?

✅ Primește cererea HTTP și o analizează.

✅ Determină acțiunea potrivită și apelează funcțiile corespunzătoare.

✅ Trimite răspunsul HTTP către client.

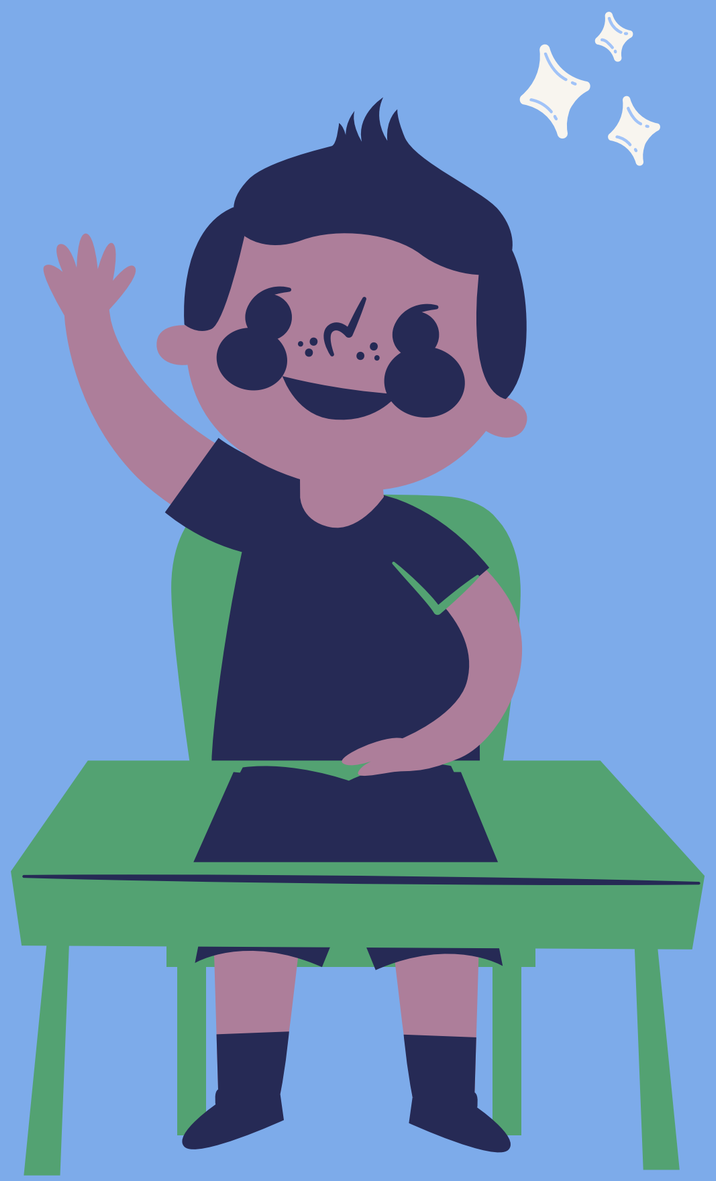
FUNCȚIA GET_MIME_TYPE() – DETECTAREA TIPULUI DE FIȘIER

📌 Această funcție determină tipul MIME al unui fișier.

🔹 Exemple de tipuri MIME gestionate:

- .html, .htm → text/html
- .jpg, .jpeg → image/jpeg
- .png → image/png
- .pdf → application/pdf

✅ Este folosită pentru a trimite corect antetul HTTP în build_http_response().



FUNCȚIA BUILD_HTTP_RESPONSE() – SERVIREA FIȘIERELOR

📌 Această funcție trimite un fișier către client (HTML, PDF, imagini).

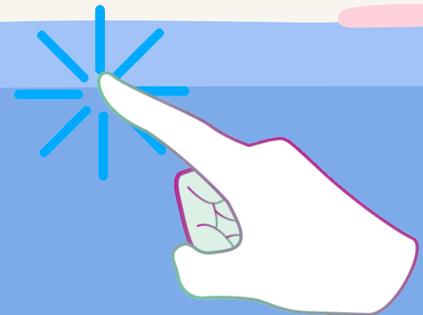
🔹 Pași principali:

✅ Determină tipul MIME al fișierului (HTML, PNG, PDF etc.).

✅ Dacă este un fișier descărcabil, adaugă antetul Content-Disposition.

✅ Deschide fișierul și îl trimite în pachete către client.

✅ În cazul unui fișier inexistent, trimite un mesaj 404 Not Found.



FUNCȚIA HANDLE_DELETE_REQUEST() – ȘTERGEREA UNEI CĂRȚI

📌 Această funcție permite ștergerea unei cărți din fișier și din memorie.

🔹 Cum funcționează?

1. Primește cererea DELETE /books/{titlu} și extrage numele cărții.
2. Decodează titlul (pentru a interpreta caractere speciale precum %20).
3. Caută cartea în lista carti[].
4. Dacă este găsită, o elimină și actualizează fișierul carti.txt.
5. Trimite un răspuns de succes sau eroare în funcție de rezultat.

Funcția handle_feedback_submission() – Procesarea Feedback-ului

📌 Funcționalitatea de feedback permite utilizatorilor să trimită un mesaj către server, care rulează un script shell (script.sh) pentru a procesa aceste date și a returna un răspuns.

🔹 Rolul acestei funcționalități

✅ Permite execuția unor scripturi personalizate pentru procesarea datelor trimise de utilizatori.

✅ Demonstrează cum serverul poate rula procese externe în mod controlat.

Cum are loc rularea scriptului?

1 Extrage datele din cererea POST: Când un utilizator completează formularul de feedback și îl trimite, browser-ul face o cerere POST către server.Serverul citește corpul cererii (body), care conține numele și prenumele introduse de utilizator. Datele sunt extrase folosind funcția extract_form_data().

2 Înlocuiește caracterele speciale: + este înlocuit cu un spațiu (deoarece în URL encoding, + reprezintă un spațiu).Se elimină spațiile goale de la început și sfârșit.

3 Se creează un proces copil pentru a rula scriptul: Serverul creează un proces nou folosind fork(). Procesul copil redirecționează ieșirea standard (stdout) și eroarea standard (stderr) către un pipe. Aceasta permite capturarea rezultatului execuției scriptului.

4 Rularea efectivă a scriptului cu execvp(): Scriptul script.sh este executat folosind execvp(). execvp() înlocuiește procesul copil cu scriptul shell și îi pasează argumentele (nume și prenume). Dacă execuția are succes, serverul nu mai controlează procesul – acesta rulează ca și cum ar fi lansat direct din terminal.

5 Așteptarea terminării scriptului și capturarea răspunsului: Procesul părinte (serverul) așteaptă terminarea scriptului folosind wait(NULL). Între timp, citește datele din pipe și le adaugă în răspunsul HTTP.

6 Trimite răspunsul clientului

MULȚUMIM!

Bibliografie

[1]. Introduction to the server side, https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction

[2]. How the web works: HTTP and CGI explained, https://www.garshol.priv.no/download/text/http-tut.html

[3]. How I Built a Simple HTTP Server from Scratch using C, https://dev.to/jeffreythecoder/how-i-built-a-simple-http-server-from-scratch-using-c-739

[4]. Thread Pool in C, https://nachtimwald.com/2019/04/12/thread-pool-in-c/

[5]. Analiza protocolului HHTP, https://wiki.mta.ro/c/3/rc/lab/11/