Mini Server Web

~ PROIECTAREA SISTEMELOR DE OPERARE ~



**Grupa C112B**

**Sd. Sg. Florea Cristian Gabriel**

**Sd. Sg. Milea Alexandru-Nicolae**

**CUPRINS**

[1.Introducere 2](#_Toc717278804)

[1.1 Scopul Proiectului 2](#_Toc575863193)

[2.Funcționalități principale 2](#_Toc800014157)

[2.1 Gestionarea conexiunilor multiple 3](#_Toc1088688657)

[2.2 Procese si thread-uri 3](#_Toc1180300104)

[2.3 Requesturi HTTP: GET, POST, PUT 3](#_Toc895613792)

[2.4 CGI (Common Gateway Interface) 3](#_Toc130343925)

[2.5 Header-e HTTP și coduri de stare 3](#_Toc477992729)

[2.6 Pagini de eroare predefinite 4](#_Toc1561774273)

[2.7 Compresie 4](#_Toc1412766054)

[3. Tehnologii utilizate 4](#_Toc810919274)

[4. Proiectarea Sistemului 4](#_Toc1277405530)

[4.1 Arhitectura 4](#_Toc148690761)

[4.2 Organizarea fișierelor 5](#_Toc1503374843)

[5. Planificare și Etape 5](#_Toc1798711662)

[6. Testare și Validare 5](#_Toc2129150104)

# **1.Introducere**

În acest proiect, voi dezvolta un mini server web care va suporta conexiuni multiple și va gestiona cereri HTTP de tip GET, POST, și PUT. Serverul va folosi thread pooling pentru a optimiza gestionarea resurselor și va fi capabil să trateze cererile de la clienți prin procese noi sau thread-uri. De asemenea, va suporta CGI (Common Gateway Interface), coduri de stare HTTP și compresie a răspunsurilor.

Serverul va fi implementat folosind limbajul C și va fi proiectat să fie portabil între diferite sisteme de operare, precum macOS, Linux și Windows, prin utilizarea de standarde POSIX și biblioteci cross-platform.

## **1.1 Scopul Proiectului**

Scopul acestui proiect este de a dezvolta un server web minimalist, dar funcțional, care să respecte principiile de proiectare a unui sistem de operare. Serverul va putea:

* Să răspundă la multiple cereri simultane folosind thread-uri.
* Să proceseze cereri HTTP standard (GET, POST, PUT).
* Să ruleze scripturi CGI pentru generarea de conținut dinamic.
* Să gestioneze erorile prin coduri de stare și pagini de eroare.
* Să ofere suport pentru compresia răspunsurilor pentru optimizarea traficului de date.

# **2.Funcționalități principale**

## **2.1 Gestionarea conexiunilor multiple**

* Utilizarea **thread pooling** pentru a gestiona un număr mare de conexiuni simultane.
* Fiecare thread va procesa cererile venite de la clienți și va returna răspunsul corespunzător.

## **2.2 Procese si thread-uri**

Serverul va putea trata clienții prin:

* **Procese noi**: Fiecare client va fi tratat prin crearea unui nou proces.
* **Thread-uri**: Fiecare cerere de la client va fi gestionată prin intermediul unui thread dedicat dintr-un pool de thread-uri.

### **2.3 Requesturi HTTP: GET, POST, PUT**

* **GET**: Returnează resursele solicitate de client (de ex. fișiere HTML).
* **POST**: Permite trimiterea de date la server (de ex. prin completarea unui formular).
* **PUT**: Permite înlocuirea unei resurse pe server cu datele noi furnizate de client.

### **2.4 CGI (Common Gateway Interface)**

* Suport pentru scripturi CGI (de exemplu, rularea unor scripturi Perl sau Python care generează răspunsuri dinamice).
* Serverul va invoca scripturile CGI și va returna rezultatele către client.

### **2.5 Header-e HTTP și coduri de stare**

* Implementarea headerelor HTTP standard, cum ar fi Content-Type, Content-Length și Content-Encoding.
* Gestionarea codurilor de stare HTTP precum:
  + 200 OK pentru cereri valide.
  + 404 Not Found pentru resurse care nu există.
  + 500 Internal Server Error pentru erori de server.

### **2.6 Pagini de eroare predefinite**

* Implementarea unor pagini predefinite (de exemplu, o pagină simplă HTML pentru eroarea 404) care vor fi afișate atunci când cererea clientului nu poate fi îndeplinită.

### **2.7 Compresie**

* Suport pentru compresia răspunsurilor (ex. gzip) pentru a reduce dimensiunea datelor transmise.
* Adăugarea header-ului Content-Encoding: gzip în cazul în care clientul acceptă compresia.

# **3. Tehnologii utilizate**

* **Limbaj de programare**: C (cu POSIX API).
* **Multithreading**: pthread pentru crearea și gestionarea thread-urilor.
* **Networking**: Socket-uri POSIX pentru gestionarea conexiunilor de rețea.
* **Compresie**: Utilizarea bibliotecii **zlib** pentru compresie.
* **CGI**: Rularea de scripturi externe pentru generarea dinamică de conținut.
* **Managementul platformei**: Proiectul va fi compatibil cu mai multe platforme (macOS, Linux, Windows).

# **4. Proiectarea Sistemului**

### **4.1 Arhitectura**

* **Server multithread**: Serverul va folosi thread pooling pentru a trata cererile multiple. Un pool fix de thread-uri va trata cererile pe măsură ce acestea vin, reducând costurile asociate cu crearea thread-urilor noi.
* **Comunicarea prin socket-uri**: Se va folosi modelul server-client bazat pe socket-uri pentru a comunica cu clienții.
* **Suport CGI**: Serverul va suporta execuția scripturilor externe (CGI) prin intermediul apelurilor de sistem.

### **4.2 Organizarea fișierelor**

* **main.c**: Fișierul principal care inițializează serverul, gestionează conexiunile și invocă thread-urile pentru fiecare client.
* **cgi\_handler.c**: Gestionarea execuției scripturilor CGI.
* **http\_parser.c**: Modulul care parsează cererile HTTP și generează răspunsuri.
* **compression.c**: Modulul care se ocupă de compresia răspunsurilor, dacă este necesar.
* **thread\_pool.c**: Implementarea unui pool de thread-uri pentru gestionarea cererilor simultane.

# **5. Planificare și Etape**

1. **Inițializarea serverului**: Crearea unui server care poate deschide un socket și asculta conexiuni.
2. **Thread pooling**: Implementarea unui pool de thread-uri care va gestiona cererile clienților.
3. **Requesturi HTTP**: Implementarea suportului pentru GET, POST și PUT.
4. **Suport CGI**: Adăugarea suportului pentru scripturi CGI.
5. **Coduri de stare și pagini de eroare**: Implementarea codurilor de stare și paginilor de eroare.
6. **Compresie**: Adăugarea suportului pentru compresia răspunsurilor.
7. **Testare pe multiple platforme**: Asigurarea compatibilității pe macOS, Linux și Windows.

# **6. Testare și Validare**

* Testarea serverului pe multiple platforme pentru a verifica portabilitatea și performanța.
* Testarea cu cereri HTTP valide și invalide pentru a verifica corectitudinea codurilor de stare și a răspunsurilor.
* Testarea compresiei și a suportului CGI pentru a asigura o performanță optimă.