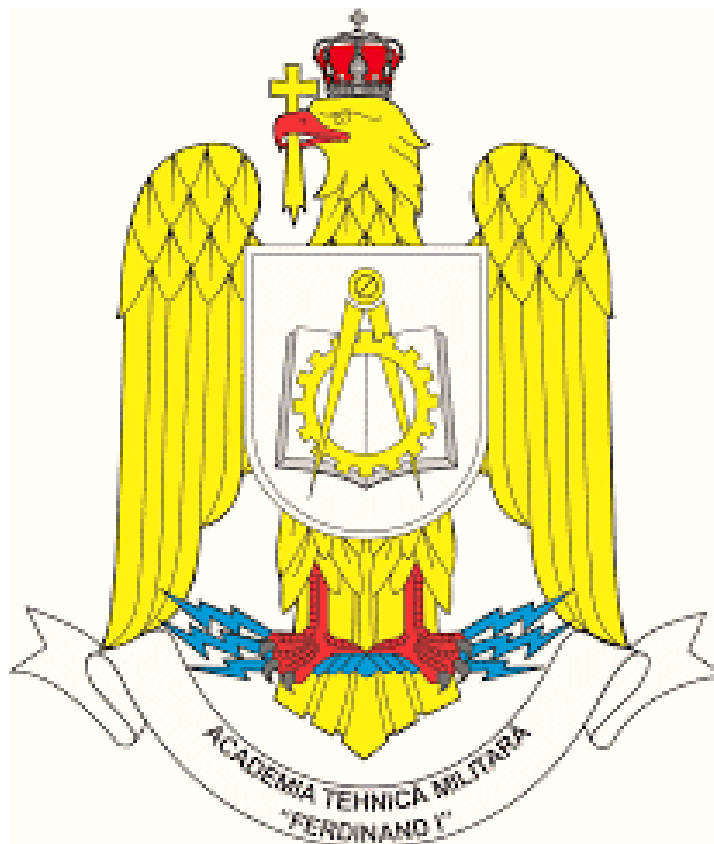


Mini WEB Server



Indrumator:

Slt. Adina VAMAN

Realizat de:

COLTAN Cristian

SOFRONIE Matei

Grupa C 113 D

Academia Tehnică Militară “Ferdinand I”

Proiect – Proiectarea Sistemelor de Operare

Cuprins

Mini WEB Server	1
Introducere	3
<i>Scopul proiectului</i>	3
<i>Lista definitiilor</i>	3
<i>Structura documentului</i>	3
Arhitectura si componente SW	4
<i>Descrierea produsului software</i>	4
<i>Detalierea platformei SW/HW</i>	4
<i>Actori</i>	4
<i>Arhitectura interna</i>	4
<i>Descriere functionalitati</i>	5
Testare functionalitati	6
<i>Test 1: Configurare port, IP, socket, bind</i>	6
<i>Test2: Verificare conexiune deja existenta pe port-ul dorit</i>	6
<i>Test3: Autentificare client</i>	7
<i>Test4: Interpretare Request GET text/html sau text/plain</i>	7
<i>Test5: Interpretare Request GET image/jpeg sau image/png</i>	7
<i>Test6: Interpretare Request GET php</i>	8
<i>Test7: Interpretare Request GET js</i>	8
<i>Test8: Interpretare Request GET fisier neinterpretabil</i>	9
<i>Test9: Interpretare Request GET fisier inexistent</i>	9
Snippets	10
Flux utilizare	10

Academia Tehnică Militară “Ferdinand I”

Proiect – Proiectarea Sistemelor de Operare

Introducere

Scopul proiectului

Crearea unui produs software care primește cereri de la clienți, cum ar fi browser-ele web, prin intermediul protocolului HTTP. Apoi, interpretează aceste cereri și furnizează conținutul corespunzător în funcție de cerințele specificate în cerere. Acest conținut poate fi, de exemplu, fișiere HTML, imagini, fișiere CSS, JavaScript sau orice alt tip de resursă web.

Lista definițiilor

Porturi de rețea: valoare numerică asociată cu un protocol care facilitează comunicarea pentru un serviciu/funcție.

IP: este un protocol de comunicație care oferă identificare și adresare pentru dispozitivele conectate la o rețea de internet sau la o altă rețea de comunicații.

Socket: oferă o metodă de a stabili și gestiona legături între două puncte terminale (de exemplu, două dispozitive sau două procesoare) pentru a permite schimbul de date. Este o pereche formată din IP și port.

Thread: (fir de execuție) este o unitate fundamentală de execuție a programului într-un proces. Un proces este un program în desfășurare pe un sistem de operare, iar un thread este o subunitate a unui proces care poate executa cod independent.

Structura documentului

Documentul este împărțit în cinci capitole. Capitolul 1 reprezintă introducerea. Capitolul 2 prezintă actorii (tipurile de utilizatori ai aplicației) și arborele de fișiere al aplicației. Capitolul 3 prezintă funcționalitățile pe care aceasta le-o oferă utilizatorului. Capitolul 4 cuprinde exemple de testare a funcționalităților aplicației. Capitolul 5 prezintă snippet-uri.

Arhitectura si componente SW

Descrierea produsului software

Aplicatia va fi dezvoltata în limbajul de programare C; transformă un dispozitiv într-un server web simplu și ușor de configurat. Aceasta oferă un mediu de dezvoltare sau de testare pentru aplicații web și permite accesul la resursele locale prin intermediul unei interfețe web.

Detalierea platformei SW/HW

Produsul software este dezvoltat pentru dispozitivele care rulează pe sistemul de operare Linux. Software-ul include un server HTTP personalizat scris în C, care gestionează cererile HTTP primite de la clienți și oferă răspunsuri corespunzătoare. Produsul include un sistem de configurare pentru a permite utilizatorilor să specifice parametrii serverului, cum ar fi portul, rădăcina site-ului, fișierele de configurare etc. Software-ul include mecanisme pentru gestionarea erorilor, generarea de coduri de stare HTTP și furnizarea de răspunsuri adecvate în caz de erori.

Actori

Utilizatorii finali: entitățile care accesează conținutul furnizat de serverul web.

Dezvoltatori: persoanele responsabile pentru crearea, dezvoltarea și menținerea platformei software/hardware a serverului web.

Clienți: dispozitivele care fac cereri HTTP către mini serverul web pentru a accesa conținutul sau pentru a comunica cu serviciile web furnizate.

Arhitectura interna

1. Componenta de gestionare a cererilor HTTP:

Analiză a cererilor: Serverul web primește cererile HTTP de la clienți și le analizează pentru a extrage informații precum URL-ul solicitat, metoda HTTP, anteturile și altele.

Tratarea cererilor: Serverul web decide cum să răspundă cererii, cum ar fi returnarea unui fișier static, executarea unui script sau rutează cererea către o aplicație web sau serviciu specific.

Academia Tehnică Militară “Ferdinand I”

Proiect – Proiectarea Sistemelor de Operare

Răspunsul la cereri: Serverul web generează răspunsul HTTP corespunzător, inclusiv codul de stare, anteturile și conținutul cerut.

2. Componenta de gestionare a conținutului:

Gestionarea fișierelor: Serverul web trebuie să poată citi și furniza fișierele de pe sistemul de fișiere al gazdei, fie ele statice (HTML, CSS, JavaScript) sau dinamice (script-uri PHP etc.).

Procesare a script-urilor: Serverul web suportă limbaje de programare server-side (cum ar fi PHP sau JavaScript), acesta poate include o componentă de procesare a script-urilor pentru a genera conținut dinamic.

3. Componenta de configurare:

Citirea fișierelor de configurare: Serverul web poate citi și interpreta fișierele de configurare pentru a stabili parametri precum portul și numărul thread-urilor.

Descriere functionalitati

1. Pornire și Oprire a Serverului:

Permite utilizatorilor să pornească și să oprească serverul web.

2. Configurare a Serverului:

Permite utilizatorilor să configureze parametrii serverului web, cum ar fi portul de ascultare, autentificare, număr clienți, și altele.

3. Autentificare:

Permite conectarea de către utilizator la server.

4. Interpretare GET text/html sau text/plain:

5. Interpretare GET image/jpeg sau image/png:

6. Interpretare GET php:

7. Interpretare GET JavaScript:

8. Interpretare GET fisier neinterpretabil:

9. Interpretare GET fisier inexistent:

Testare functionalitati

Test 1: Configurare port, IP, socket, bind

INPUT:

IP: localhost, Port: 8080

OUTPUT:

Socket configured!

Socket bind successful!

Listening on port 8080

```
cristian@Coltan:~/VSCode$ ./main
Socket configured!
Socket bind successful!
Listening on port 8080
|
```

Test2: Verificare conexiune deja existenta pe port-ul dorit

INPUT:

IP: localhost, Port: 8080

OUTPUT:

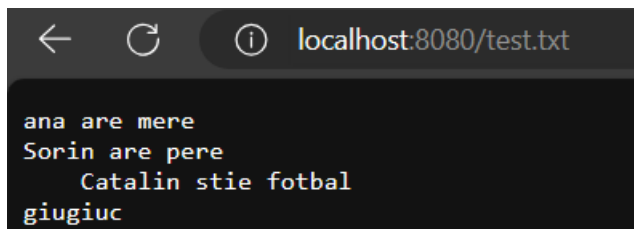
Bind fail!: Address already in use

```
cristian@Coltan:~/VSCode$ ./main
Bind fail!: Address already in use
```

Academia Tehnică Militară “Ferdinand I”
Proiect – Proiectarea Sistemelor de Operare
Test3: Autentificare client

```
○ cristian@Coltan:~/VSCode$ ./main
Welcome!
App configured successful!
Username: admin
Password: admin
Login succesful!
Listening on port 8080
```

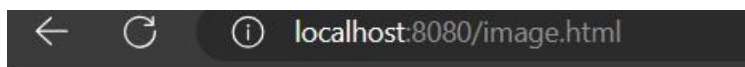
Test4: Interpretare Request GET text/html sau text/plain



← ↻ ⓘ localhost:8080/test.txt

ana are mere
Sorin are pere
Catalin stie fotbal
giugiuc

Test5: Interpretare Request GET image/jpeg sau image/png



← ↻ ⓘ localhost:8080/image.html

Checking if it's working!

Yeah, it's working!



Academia Tehnică Militară "Ferdinand I"
Proiect – Proiectarea Sistemelor de Operare
Test6: Interpretare Request GET php

```
script.php
1  Generated message:
2
3  <?php |
4  |     echo "Hello from PHP!";
5  ?>
6
7  Salut!
```

← ↻ ⓘ localhost:8080/script.php
Generated message: Hello from PHP! Salut!

Test7: Interpretare Request GET js

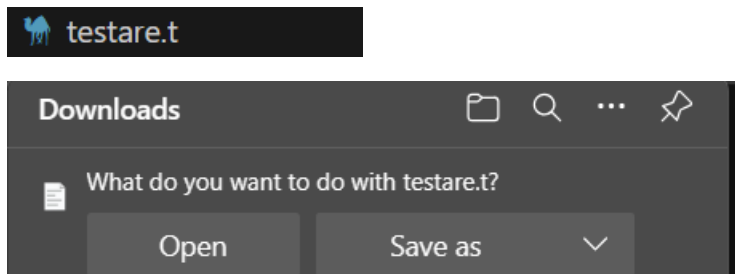
```
JS scriptjava.js > ...
1  function greet(name) {
2  |     return 'Hello, ' + name + '!';
3  | }
4
5  let message = greet('JavaScript!!!');
6  console.log(message);
7
```

← ↻ ⓘ localhost:8080/scriptjava.js
Hello, JavaScript!!!!

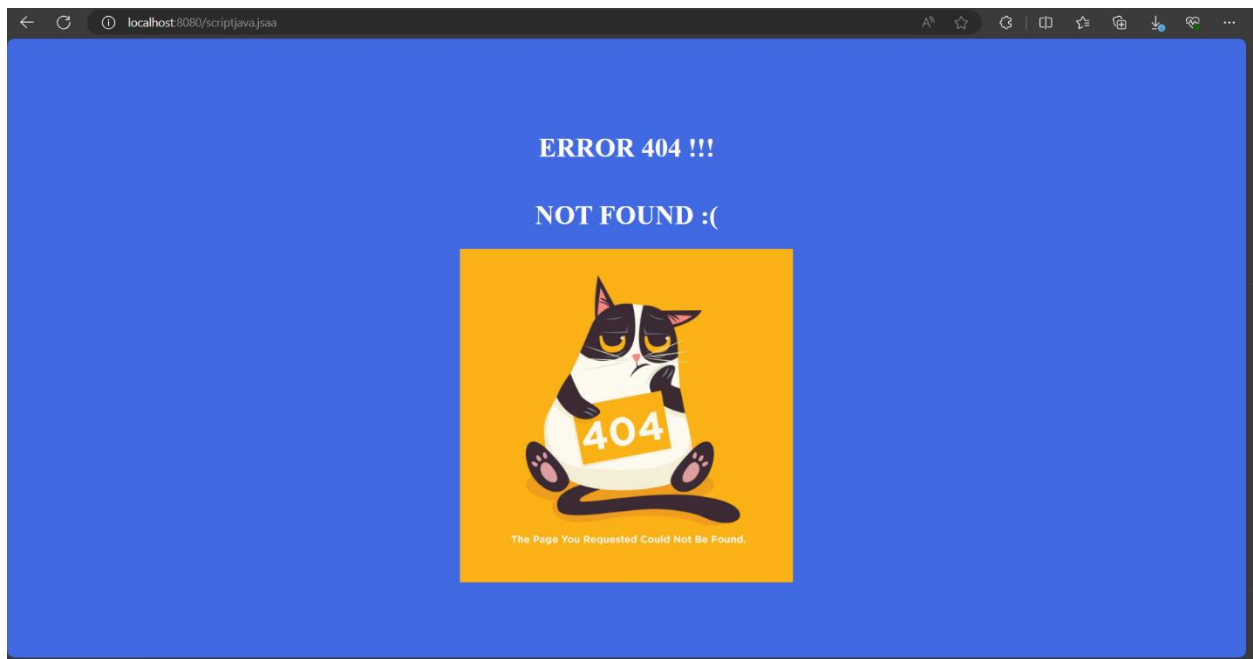
Academia Tehnică Militară “Ferdinand I”

Proiect – Proiectarea Sistemelor de Operare

Test8: Interpretare Request GET fisier neinterpretabil



Test9: Interpretare Request GET fisier inexistent



Snippets

```
// SERVER
const char *get_file_extension(const char *filename);
//aflam extenisa fisierului

const char *get_mime_type(const char *file_ext);
//aflam mime_type

char *url_decode(const char *src);
//decodam caracterele URL

void build_http_response(const char *file_name, const char *file_ext,
                        char *response, size_t *response_len);
//creem header-ul de raspuns

void *handle_client(void *arg);
//functia pentru comportamentul thread - ului

int login(const char *username, const char *password);
//functia de logare

void interpretPHP(const char *file_name);
//interpretare php

void interpretJAVA(const char *file_name);
//interpretare js
```

Flux utilizare

- make main
- ./main
- (intram in browser pe <http://localhost:8080/>)
- Alegem ce fisier dorim sa fie parsat (ex: <http://localhost:8080/image.html>)
-