Universidade Federal de Uberlândia

FEELT – Faculdade de Engenharia Elétrica

Redes 1

Trabalho 01:
Web Server Multi thread Java

Professor: Paulo Guardieiro
Engenharia da computação

Aluno: Rafael Lucas Pereira 12011ECP026

21/02/2022

Objetivos

O objetivo deste trabalho é a implementação de um servidor Web "multithreaded". Implementamos a versão 1.0 do HTTP, definida no RFC 1945, onde mensagens de pedido HTTP separadas são enviadas para cada objeto da página Web.Este servidor deverá manipular múltiplas requisições simultâneas de serviço em paralelo. Isso significa que o servidor Web é "multithreaded". Ele deverá ser capaz de atender pedidos de transferência de arquivos gerados a partir de um browser comum, como o Firefox. Os pedidos serão apresentados em uma porta específica e o servidor Web deverá aceitar conexões não persistentes e simultâneas de um número arbitrário de clientes Web.

Introdução

O servidor Web é um dos componentes da aplicação Web e se baseia num programa servidor. Quando esse programa é executado, cria-se o processo servidor que se mantém na “escuta” pelos clientes Web que queiram extrair documentos usando o protocolo HTTP. Por meio de conexões TCP, ele recebe pedidos e envia respostas até que o cliente Web ou o próprio servidor Webdecida encerrar a conexão.

Desenvolvimento

A partir dos passos enumerados no arquivo "Tarefa de Programação 1: Construindo um servidor Web multithreaded", o arquivo "WebServer.java" foi construído.

Este arquivo se trata de um código java que implementa um servidor web que opera a versão 1.0 do HTTP (RFC-1945). A operação do mesmo funciona de forma simples, implementamos uma classe principal que cria uma thread para ouvir uma porta fixa até que um cliente faça uma requisição TCP, nesse momento o sistema cria uma thread separada para aquela requisição e resolve esta conexão em uma porta distinta.

Como é um código simples, o servidor têm por função servir 4 rotas principais:

- http://localhost:3000/index.html
- http://localhost:3000/relatorio.pdf
- http://localhost:3000/matrix.gif
- http://localhost:3000/hackerman.jpg
- http://localhost:3000/notfound (ou qualquer outra rota que não esteja bem definida)

Testes

Para testar o servidor, como as chamadas são todas do tipo GET, podemos utilizar apenas um browser e nos conectarmos através de uma chamada local na porta 3000.

Para isso, primeiramente é necessário que executemos o servidor, para isso basta abrir um shell e executar o seguinte comando (é necessário ter uma versão do java instalada em sua máquina e executar este comando na pasta do projeto):

javac WebServer.java

Esse comando criará dois arquivos .class, o WebServer que é nossa classe principal e a classe de consumo chamada de HttpRequest. Com o procedimento realizado, basta executar o comando:

java WebServer

Isso fará com que nosso método main da classe WebServer seja executado e crie a thread principal, a partir dai, qualquer requisição feita na porta 3000 por um cliente será operada pelo servidor, que por sua vez "logga" todas as requisições no console.

Com isso feito podemos começar a realizar nossos testes. Para isso basta abrir o browser de sua preferência e chamar as rotas citadas acima.

Para testar a operação do sistema, enumeramos as respostas tanto do cliente, quanto do servidor para cada uma das rotas:

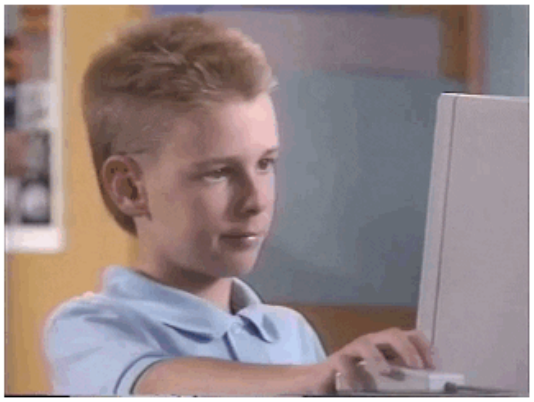
Index.html

O console apresenta o seguinte log no momento em que a rota é acessada no cliente:

```
GET /brant.gif HTTP/1.1
Host: localhost:3000
Connection: keep-alive
sec-ch-ua: " Not;A Brand";v="99", "Google Chrome";v="97", "Chromium";v=
"97"
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/97.0.4692.71 Safari/537.36
sec-ch-ua-platform: "Linux"
Accept: application/signed-exchange;v=b3;q=0.7,*/*;q=0.8
Purpose: prefetch
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: image
Referer: http://localhost:3000/index.html
Accept-Encoding: gzip, deflate, br
Accept-Language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: _hjSessionUser_1744038=eyJpZCI6IjdjYmE2NzUyLTNhYWItNTY5Ni1hZmM3
LTYxYmNmNzE2OWI0MyIsImNyZWF0ZWQiOjE2NDE5OTUzNDY2NzMsImV4aXN0aW5nIjp0cnV
lfQ==; bu=ALIMENTARGRANDESVAREJOS; _ga=GA1.1.2060422030.1642690470; _hj
SessionUser_1875841=eyJpZCI6IjdjOGI5NjkzLWY5YmMtNTA4Ny04M2ZjLWJkZjU1ZDE
2YTJjNCIsImNyZWF0ZWQiOjE2NDI2OTA0NzAzOTQsImV4aXN0aW5nIjpmYWxzZX0=
```

Após o servidor processar a solicitação, ele retorna a página index.html da seguinte forma:

Web Server Multithread em Java



Navegue por aqui

Visualizar uma imagem Visualizar um gif Visualizar pdf Not found

O resultado, portanto, é o esperado, a apresentação de uma página html simples.

hackerman.jpeg

Novamente, no momento em que acessamos a rota pelo cliente, o retorno do console mostra as informações da requisição http:

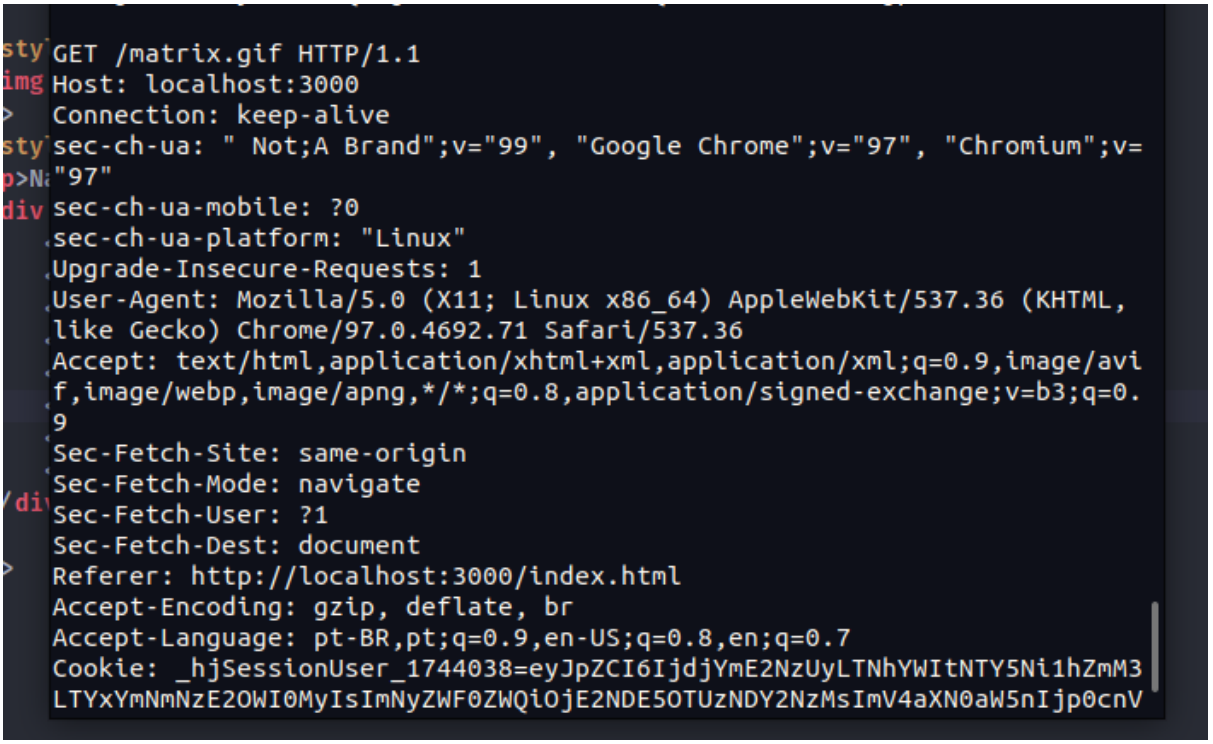
```
GET /hackerman.jpg HTTP/1.1
Host: localhost:3000
Connection: keep-alive
sec-ch-ua: " Not;A Brand";v="99", "Google Chrome";v="97", "Chromium";v=
"97"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/97.0.4692.71 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.
9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:3000/index.html
Accept-Encoding: gzip, deflate, br
Accept-Language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: _hjSessionUser_1744038=eyJpZCI6IjdjYmE2NzUyLTNhYWItNTY5Ni1hZmM3
LTYxYmNmNzE2OWI0MyIsImNyZWF0ZWQiOjE2NDE5OTUzNDY2NzMsImV4aXN0aW5nIjp0cnV
1f0--: hu=ALIMENTARGRANDESVAREJOS: _ga=GA1.1.2060422030.1642690470: _hj
```

Em seguida, após o processamento, o servidor entrega uma imagem .jpg

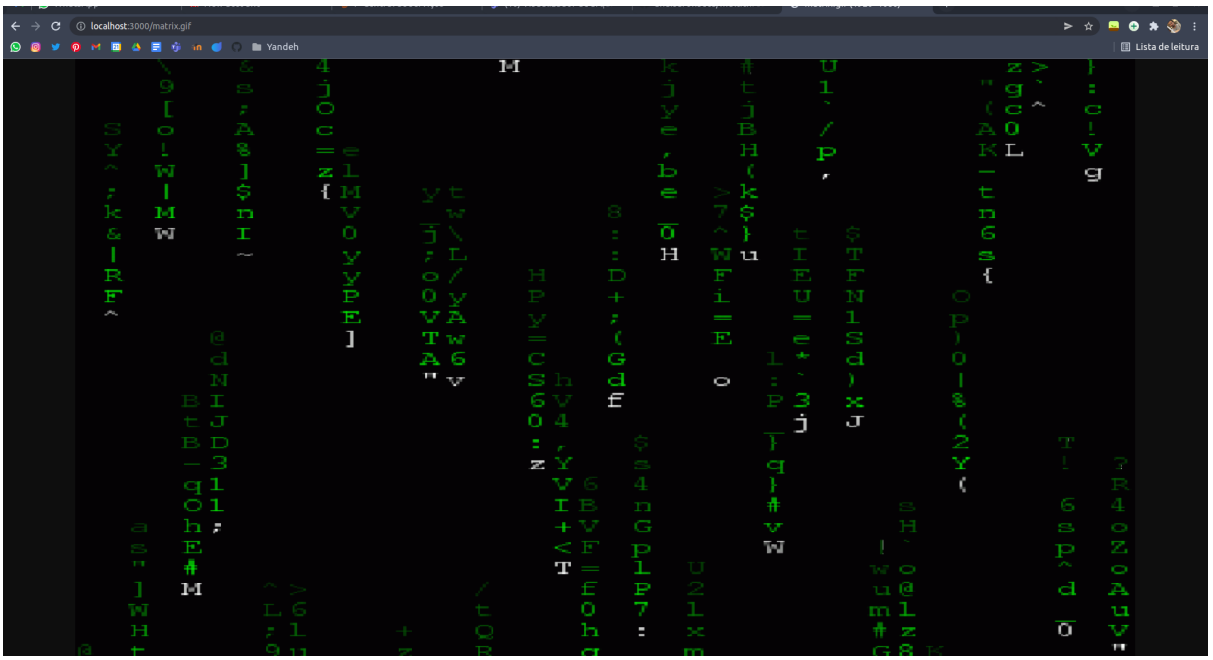
Novamente o resultado apresentado é como esperávamos, a apresentação de uma imagem JPG.

Matrix.gif

Analogamente as requisiçoẽs anteriores, temos o seguinte retorno do console:



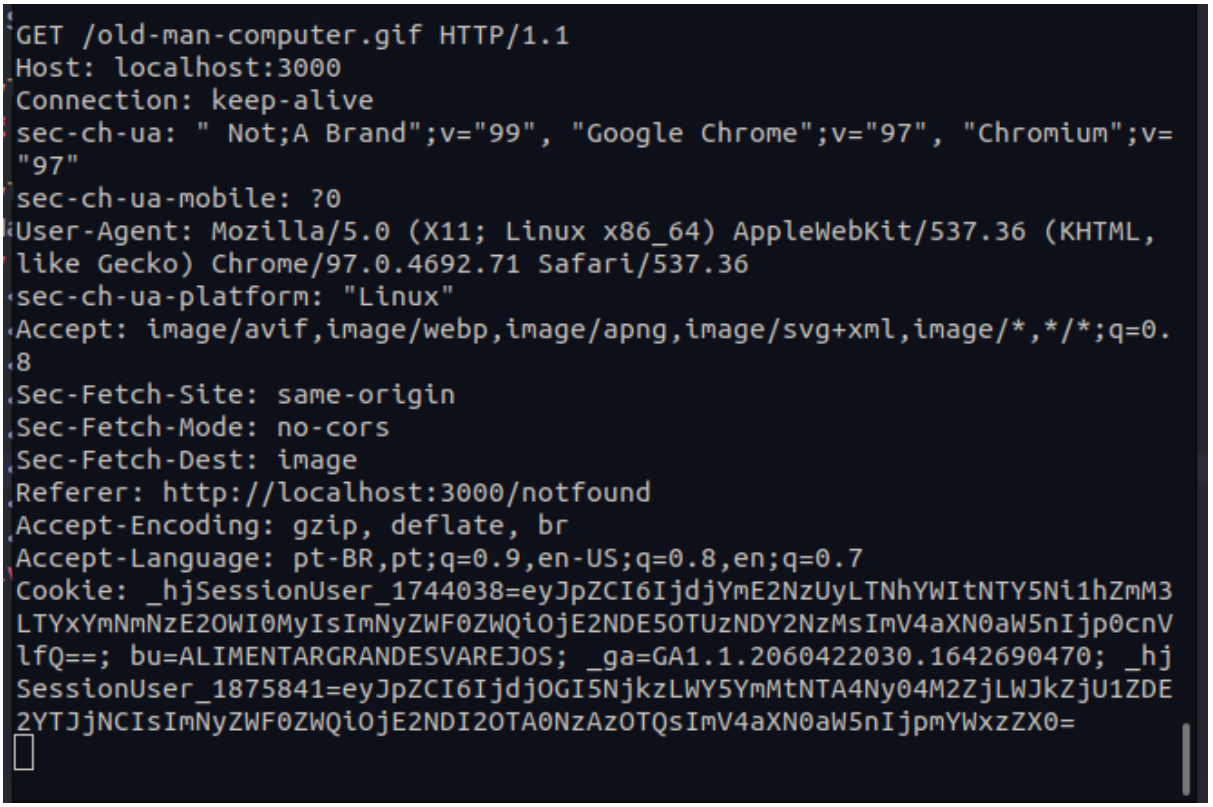
Sendo assim o server, entrega um gif após a operação:



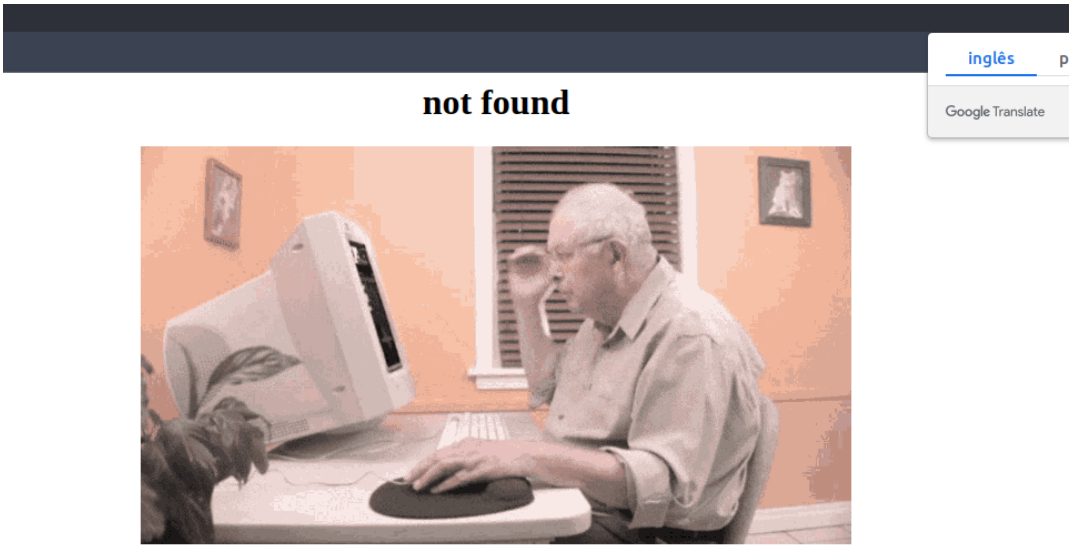
Mais uma vez temos o resultado que esperávamos.

Not Found

Por fim, temos a rota para erros de busca, então sempre que o server não encontrar o arquivo requisitado ele retorna uma mensagem 400 com o erro not found apresentando uma mensagem. No momento em que fazemos essa requisição obtemos:



O conteúdo apresentado é o seguinte:



Conclusão

O trabalho ajudou na compreensão dos conceitos de redes de maneira prática, implementando um sistema simples e divertido de ser construído.