

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FEELT – FACULDADE DE ENGENHARIA ELÉTRICA
ENGENHARIA DE COMPUTAÇÃO

LUCAS ALBINO MARTINS
12011ECP022

**REDES(RC e RC-1): PRIMEIRO TRABALHO: IMPLEMENTAÇÃO DE UM
SERVIDOR WEB MULTITHREADED.**

UBERLÂNDIA
2021

Tarefa 1: Servidor Web

Nesta tarefa, você desenvolverá um servidor Web simples em Python, capaz de processar apenas uma requisição. Seu servidor Web (i) **criará um socket de conexão quando contatado por um cliente (navegador)**; (ii) **receberá a requisição HTTP dessa conexão**; (iii) **analisará a requisição para determinar o arquivo específico sendo requisitado**; (iv) **obterá o arquivo requisitado do sistema de arquivo do servidor**; (v) **criará uma mensagem de resposta HTTP consistindo no arquivo requisitado precedido por linhas de cabeçalho**; e (vi) **enviará a resposta pela conexão TCP ao navegador requisitante. Se um navegador requisitar um arquivo que não está presente no seu servidor, seu servidor deverá retornar uma mensagem de erro “404 Not Found”**. No site de apoio, oferecemos o código estrutural para o seu servidor. Sua tarefa é concluir o código, rodar seu servidor e depois testá-lo enviando requisições de navegadores rodando em hospedeiros diferentes. Se você rodar seu servidor em um hospedeiro que já tem um servidor Web rodando nele, então deverá usar uma porta diferente da porta 80 para o seu servidor.

(i) Criação de um socket de conexão quando contatado por um cliente (navegador).

```
def obtem_ip_externo():
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect(('8.8.8.8', 80))
        ip_externo = s.getsockname()[0]
        s.close()

        return ip_externo
    except:
        return 'localhost'

ip_servidor = 'localhost'
if USAR_IP_EXTERNO:
    ip_servidor = obtem_ip_externo()
```

(ii) Receberá a requisição HTTP dessa conexão;

```
def processa_requisicao(con):
    requisicao = con.recv(2048).decode()

    # Processa os campos da requisição HTTP
    inicio_requisicao = False
    campos_requisicao = {}

    for l in requisicao.split('\n'):
        if not inicio_requisicao:
            inicio_requisicao = l
            continue

        campo = l.strip('\r').split(':')[0]
        valor = ':'.join(l.strip('\r').split(':')[1:]).strip()

        campos_requisicao[campo] = valor
```

(iii) Analisara a requisição para determinar o arquivo específico sendo requisitado.

```
try:
    metodo_http, recurso, versao_http = inicio_requisicao.split(' ')
except ValueError:
    metodo_http = 'GET'
    recurso = '/'
    versao_http = 'HTTP/1.1'

with open(ARQUIVO_LOG, 'a') as f:
    f.write(inicio_requisicao + ';')
    if 'User-Agent' in campos_requisicao:
        f.write(campos_requisicao['User-Agent'])
    f.write('\n')
```

(iv) analisara a requisição para determinar o arquivo específico sendo requisitado;

```
# Cabeçalho de resposta HTTP
cabecalho_resp = 'HTTP/1.1 {0} {1}\nContent-Type: {2}\n\n'.format(
    codigo_resposta,
    txt_resposta,
    content_type
)

# Envia o cabeçalho para o cliente
con.send(cabecalho_resp.encode())

# Envia o conteúdo para o cliente
TAM_BUFFER = 1024

# Codifica o conteúdo caso o mesmo não esteja em formato binário
if isinstance(conteudo_resposta, str):
    conteudo_resposta = conteudo_resposta.encode()

for i in range(0, len(conteudo_resposta), TAM_BUFFER):
    try:
        con.send(conteudo_resposta[i:i + TAM_BUFFER])
    except BrokenPipeError:
        pass

# Fecha a conexão
con.close()
```

(v) criara uma mensagem de resposta HTTP consistindo no arquivo requisitado precedido por linhas de cabeçalho. Se um navegador requisitar um arquivo que não está presente no seu servidor, seu servidor deverá retornar uma mensagem de erro “404 Not Found”

```
codigo_resposta = '200'
txt_resposta = 'OK'
conteudo_resposta = '<h1>Pagina encontrada</h1>'
content_type = 'text/html'
```

```
codigo_resposta = '404'  
txt_resposta = 'Not Found'  
conteudo_resposta = HTML_NOT_FOUND
```

Teste 1. Teste de resposta do servidor.

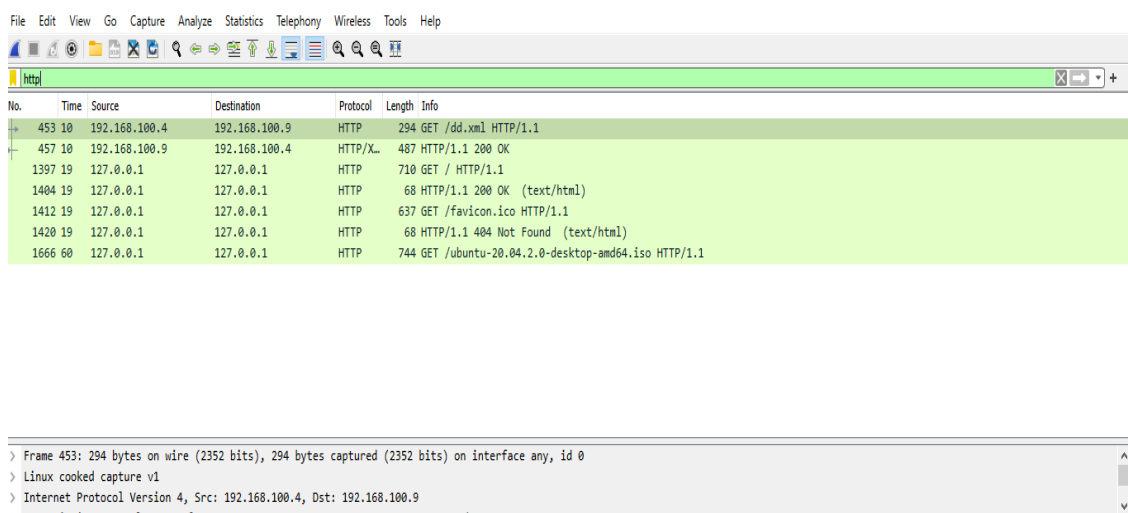
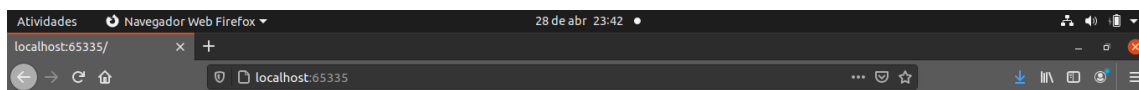


Figura 1. – Print da captura de pacotes HTTP(WireShark).

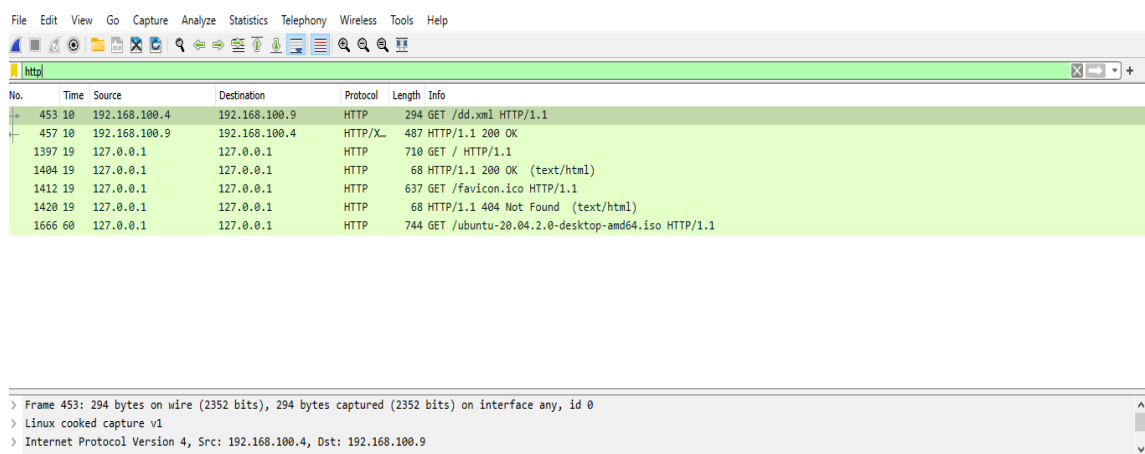


Pagina encontrada



Figura 2. – Teste utilizando o Firefox.

Teste 2. Teste para requisar um arquivo que não está no servidor.



The image shows a WireShark packet capture window. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons. The main display area shows a list of captured packets. The selected packet is number 453, which is an HTTP GET request for /dd.xml. The status bar at the bottom shows details for the selected packet: Frame 453: 294 bytes on wire (2352 bits), 294 bytes captured (2352 bits) on interface any, id 0. The packet details pane shows the following information: Linux cooked capture v1, Internet Protocol Version 4, Src: 192.168.100.4, Dst: 192.168.100.9, and Hypertext Transfer Protocol.

No.	Time	Source	Destination	Protocol	Length	Info
453	10	192.168.100.4	192.168.100.9	HTTP	294	GET /dd.xml HTTP/1.1
457	10	192.168.100.9	192.168.100.4	HTTP/X-	487	HTTP/1.1 200 OK
1397	19	127.0.0.1	127.0.0.1	HTTP	710	GET / HTTP/1.1
1404	19	127.0.0.1	127.0.0.1	HTTP	68	HTTP/1.1 200 OK (text/html)
1412	19	127.0.0.1	127.0.0.1	HTTP	637	GET /favicon.ico HTTP/1.1
1420	19	127.0.0.1	127.0.0.1	HTTP	68	HTTP/1.1 404 Not Found (text/html)
1666	60	127.0.0.1	127.0.0.1	HTTP	744	GET /ubuntu-20.04.2.0-desktop-amd64.iso HTTP/1.1

Figura 3. – Print da captura de pacotes HTTP(WireShark).

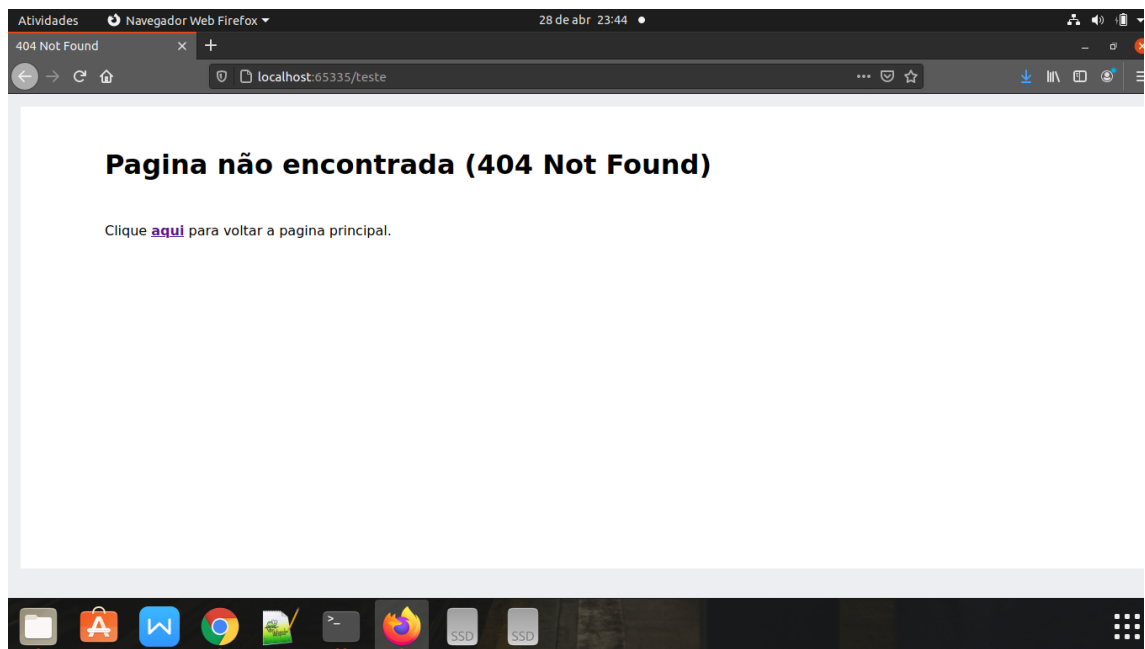
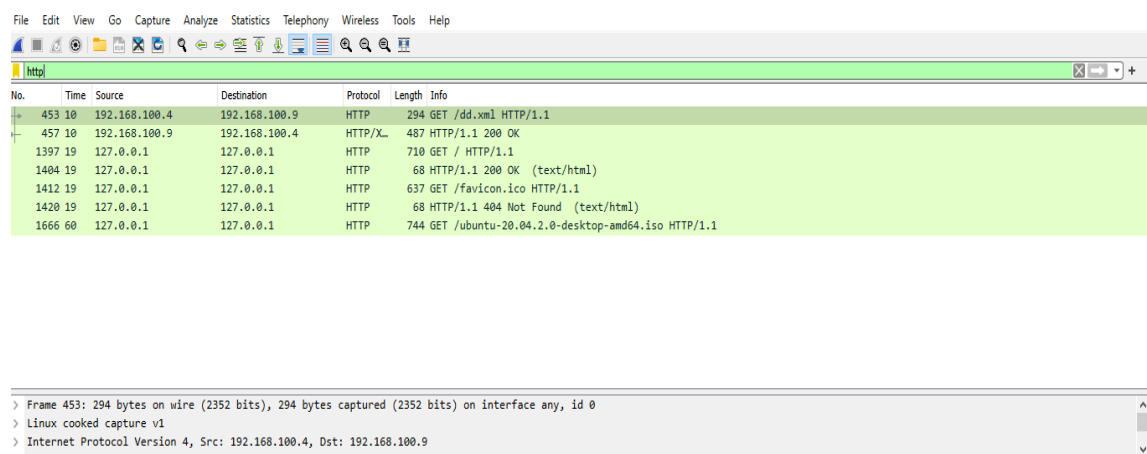


Figura 4. – Teste 404-error utilizando o Firefox

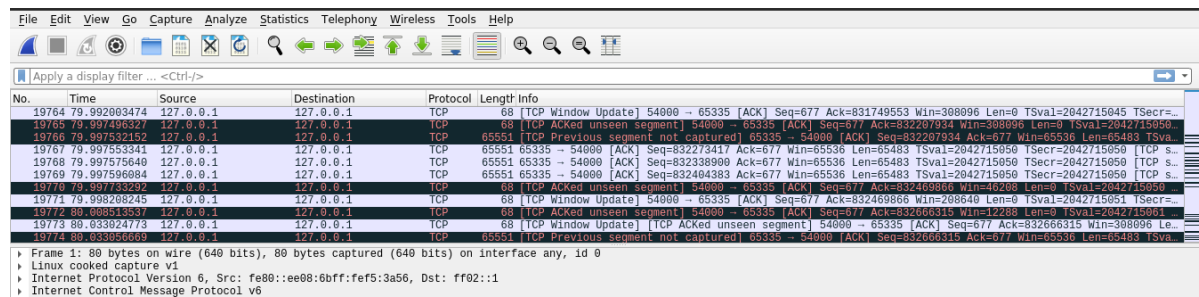
Teste 3. Teste de download de um arquivo pelo navegador.



No.	Time	Source	Destination	Protocol	Length	Info
453	10	192.168.100.4	192.168.100.9	HTTP	294	GET /dd.xml HTTP/1.1
457	10	192.168.100.9	192.168.100.4	HTTP/X	487	HTTP/1.1 200 OK
1397	19	127.0.0.1	127.0.0.1	HTTP	710	GET / HTTP/1.1
1404	19	127.0.0.1	127.0.0.1	HTTP	68	HTTP/1.1 200 OK (text/html)
1412	19	127.0.0.1	127.0.0.1	HTTP	637	GET /favicon.ico HTTP/1.1
1420	19	127.0.0.1	127.0.0.1	HTTP	68	HTTP/1.1 404 Not Found (text/html)
1666	60	127.0.0.1	127.0.0.1	HTTP	744	GET /ubuntu-20.04.2.0-desktop-amd64.iso HTTP/1.1

> Frame 453: 294 bytes on wire (2352 bits), 294 bytes captured (2352 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 192.168.100.4, Dst: 192.168.100.9
> Internet Protocol Version 4, Src: 192.168.100.4, Dst: 192.168.100.9

Figura 5. – Print da captura de pacotes HTTP(WireShark).

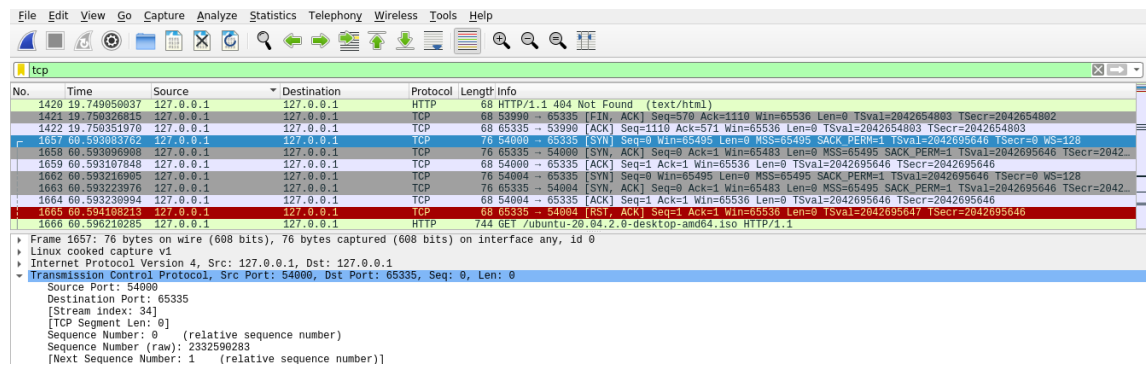


No.	Time	Source	Destination	Protocol	Length	Info
19764	79.992003474	127.0.0.1	127.0.0.1	TCP	68	[TCP Window Update] 54000 -> 65335 [ACK] Seq=677 Ack=831749553 Win=308096 Len=0 TSval=2042715045 TSecr=...
19765	79.997496327	127.0.0.1	127.0.0.1	TCP	68	[TCP ACKed unseen segment] 54000 -> 65335 [ACK] Seq=677 Ack=832207934 Win=308096 Len=0 TSval=2042715050 TSecr=...
19766	79.997527152	127.0.0.1	127.0.0.1	TCP	65551	65335 -> 54000 [ACK] Seq=832273417 Ack=677 Win=65536 Len=65483 TSval=2042715050 TSecr=2042715050 [TCP s...
19767	79.997553341	127.0.0.1	127.0.0.1	TCP	65551	65335 -> 54000 [ACK] Seq=832273417 Ack=677 Win=65536 Len=65483 TSval=2042715050 TSecr=2042715050 [TCP s...
19768	79.997575640	127.0.0.1	127.0.0.1	TCP	65551	65335 -> 54000 [ACK] Seq=832338900 Ack=677 Win=65536 Len=65483 TSval=2042715050 TSecr=2042715050 [TCP s...
19769	79.997596084	127.0.0.1	127.0.0.1	TCP	65551	65335 -> 54000 [ACK] Seq=832404383 Ack=677 Win=65536 Len=65483 TSval=2042715050 TSecr=2042715050 [TCP s...
19770	79.997618527	127.0.0.1	127.0.0.1	TCP	68	[TCP ACKed unseen segment] 54000 -> 65335 [ACK] Seq=677 Ack=832404383 Win=12288 Len=0 TSval=2042715050 TSecr=...
19771	79.998208245	127.0.0.1	127.0.0.1	TCP	68	[TCP Window Update] 54000 -> 65335 [ACK] Seq=677 Ack=832404383 Win=12288 Len=0 TSval=2042715051 TSecr=...
19772	80.008513537	127.0.0.1	127.0.0.1	TCP	68	[TCP ACKed unseen segment] 54000 -> 65335 [ACK] Seq=677 Ack=832606315 Win=12288 Len=0 TSval=2042715061 TSecr=...
19773	80.033024773	127.0.0.1	127.0.0.1	TCP	68	[TCP Window Update] [TCP ACKed unseen segment] 54000 -> 65335 [ACK] Seq=677 Ack=832666315 Win=308096 Le...
19774	80.033050669	127.0.0.1	127.0.0.1	TCP	65551	[TCP Previous segment not captured] 65335 -> 54000 [ACK] Seq=832666315 Ack=677 Win=65536 Len=65483 TSva...

> Frame 1: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 6, Src: fe80::ee08:6bff:fe5:3a56, Dst: ff02::1
> Internet Control Message Protocol v6

Figura 6. – Print da captura de pacotes TCP (WireShark).

Teste 4. Teste Three-way Handshake.



No.	Time	Source	Destination	Protocol	Length	Info
1420	19.749059037	127.0.0.1	127.0.0.1	HTTP	68	HTTP/1.1 404 Not Found (text/html)
1421	19.750320815	127.0.0.1	127.0.0.1	TCP	68	53990 -> 65335 [FIN, ACK] Seq=570 Ack=1110 Win=65536 Len=0 TSval=2042654803 TSecr=2042654802
1422	19.750351970	127.0.0.1	127.0.0.1	TCP	68	65335 -> 53990 [ACK] Seq=1110 Ack=571 Win=65536 Len=0 TSval=2042654803 TSecr=2042654803
1657	0.3650035702	127.0.0.1	127.0.0.1	TCP	76	54000 -> 65335 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=2042695646 TSecr=0 WS=128
1658	0.593096998	127.0.0.1	127.0.0.1	TCP	76	65335 -> 54000 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=2042695646 TSecr=2042...
1659	0.593107048	127.0.0.1	127.0.0.1	TCP	68	54000 -> 65335 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2042695646 TSecr=2042695646
1662	0.593216905	127.0.0.1	127.0.0.1	TCP	76	54004 -> 65335 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=2042695646 TSecr=0 WS=128
1663	0.593223976	127.0.0.1	127.0.0.1	TCP	76	65335 -> 54004 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=2042695646 TSecr=2042...
1664	0.593239994	127.0.0.1	127.0.0.1	TCP	68	54004 -> 65335 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2042695646 TSecr=2042695646
1665	0.594108213	127.0.0.1	127.0.0.1	TCP	68	65335 -> 54004 [RST, ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=2042695647 TSecr=2042695646
1666	0.596219285	127.0.0.1	127.0.0.1	HTTP	744	GET /ubuntu-20.04.2.0-desktop-amd64.iso HTTP/1.1

> Frame 1657: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 54000, Dst Port: 65335, Seq: 0, Len: 0
Source Port: 54000
Destination Port: 65335
[Stream index: 34]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 2332590293
[Next Sequence Number: 1 (relative sequence number)]

Figura 7. – Print da captura de pacotes TCP(SYN,ACK) pelo WireShark.

Teste 5. Teste de acesso a um arquivo no servidor.

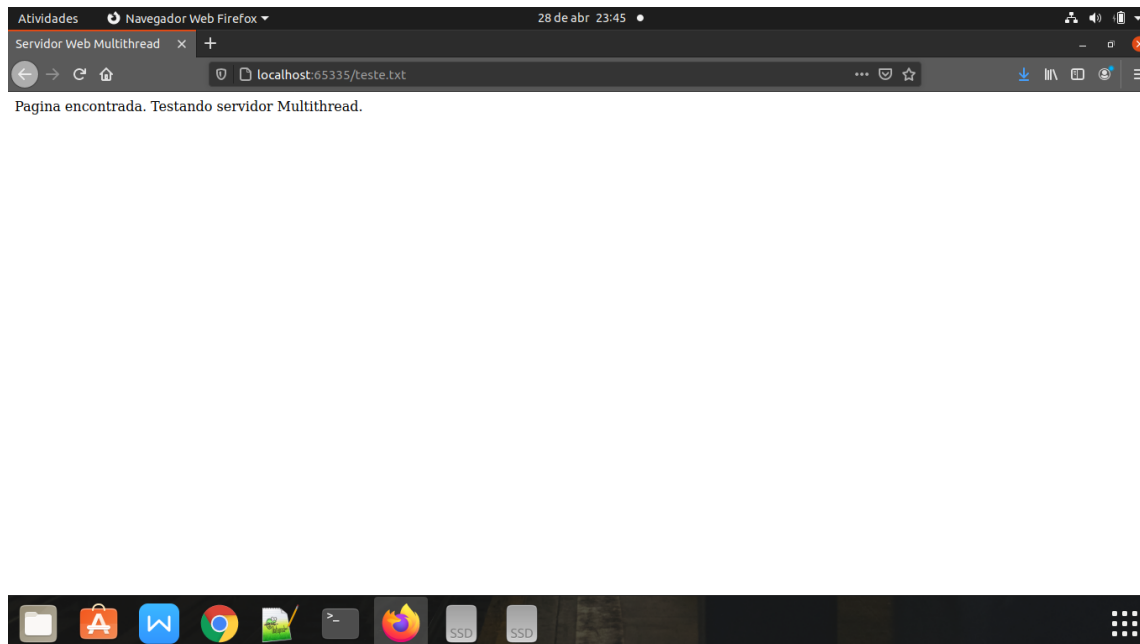


Figura 8. – Teste abrindo arquivo no servidor utilizando o Firefox

Código:

```
# Servidor Web Multithread
# Objetivo: Cria um servidor web para multi conexões.
# Testes: Index, Download, erro404
# Disciplinas: Redes de Computadores I
# Aluno: Lucas Albino Martins
```

```
import datetime
import os
import socket
import sys
import time
```

```
from threading import Thread
```

```
import magic
```

```
# Caso False, escuta em localhost
USAR_IP_EXTERNO = False
```

```
ARQUIVO_LOG = 'log_servidor.txt'
```

```
HTML_LISTAGEM = """
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>{0}</title>
    <style type="text/css">
      *{
```

```
margin: 0;
padding: 0;
font-family: sans-serif;
}}

body{{
background-color: #ecef1;
}}

main{{
display: block;
margin: 15px;
padding: 50px 100px;
background-color: #fff;
height: calc(100vh - 150px);
}}

h1{{
margin-bottom: 50px;
}}

a, span{{
display: inline-block;
}}

a{{
width: 30%;
padding-left: 15px;
}}

a{{
font-weight: bold
}}

.tam_bytes{{
width: 15%;
padding: 20px 50px;
text-align: center;
}}

.mod_date{{
width: 30%;
padding: 20px 50px;
text-align: center;
}}

.item_listagem{{
border-bottom: solid 1px #eee;
}}

.titulo_listagem{{
font-weight: bold;
text-transform: uppercase;
font-size: 13px;
}}

.t{{
text-align: left;
width: 30%;
padding-left: 15px;
```



```

    }}

    .voltar{{
        font-size: 20px;
    }}
</style>
</head>
<body>
    <main>
        <h1>{0}</h1>
        <p class="voltar">
            {2}
        </p>
        <div class="titulo_listagem">
            <span class="t">Nome</span>
            <span class="tam_bytes">Tamanho (bytes)</span>
            <span class="mod_date">Última modificação</span>
        </div>
        {1}
    </main>
</body>
</html>
'''

```

Criando erro 404

```

HTML_NOT_FOUND = '''
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>404 Not Found</title>
        <style type="text/css">
            *{
                margin: 0;
                padding: 0;
                font-family: sans-serif;
            }

            body{
                background-color: #ecef1;
            }

            main{
                display: block;
                margin: 15px;
                padding: 50px 100px;
                background-color: #fff;
                height: calc(100vh - 150px);
            }

            h1{
                margin-bottom: 50px;
            }

            a{
                font-weight: bold;
            }
        </style>
    </head>
    <body>

```

```

        <main>
        <h1>Pagina não encontrada (404 Not Found)</h1>
        <p>Clique <a href="/">aqui</a> para voltar a pagina principal.</p>
        </main>
    </body>
</html>
'''

```

```

def obtem_ip_externo():
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        s.connect(('8.8.8.8', 80))
        ip_externo = s.getsockname()[0]
        s.close()

        return ip_externo
    except:
        return 'localhost'

```

```

ip_servidor = 'localhost'
if USAR_IP_EXTERNO:
    ip_servidor = obtem_ip_externo()

```

```

def processa_requisicao(con):
    requisicao = con.recv(2048).decode()

    # Processa os campos da requisição HTTP
    inicio_requisicao = False
    campos_requisicao = {}

    for l in requisicao.split('\n'):
        if not inicio_requisicao:
            inicio_requisicao = l
            continue

        campo = l.strip('\r').split(':')[0]
        valor = ':'.join(l.strip('\r').split(':')[1:]).strip()

        campos_requisicao[campo] = valor

```

```

    try:
        metodo_http, recurso, versao_http = inicio_requisicao.split(' ')
    except ValueError:
        metodo_http = 'GET'
        recurso = '/'
        versao_http = 'HTTP/1.1'

```

```

    with open(ARQUIVO_LOG, 'a') as f:
        f.write(inicio_requisicao + ';')
        if 'User-Agent' in campos_requisicao:
            f.write(campos_requisicao['User-Agent'])
        f.write('\n')

```

```

# Evita o uso de .. para voltar diretórios
recurso = recurso.replace('..', '').replace('//', '/')

```

```
codigo_resposta = '200'
txt_resposta = 'OK'
conteudo_resposta = '<h1>Pagina encontrada</h1>'
content_type = 'text/html'
```

```
# Checa se o recurso existe e se é um arquivo ou diretório
caminho_recurso = (diretorio + recurso).replace('//', '/')
```

```
if os.path.isfile(caminho_recurso):
    # Descobre o MIME type do arquivo
    mime = magic.Magic(mime=True)
    content_type = mime.from_file(caminho_recurso)
```

```
# Lê o conteúdo do arquivo
conteudo_resposta = open(caminho_recurso, 'rb').read()
```

```
elif os.path.isdir(caminho_recurso):
    listagem = ""
    for i in os.listdir(caminho_recurso):
```

```
        data_modificacao = time.ctime(os.path.getmtime(caminho_recurso.rstrip('/') + '/' +
```

i))

```
        if os.path.isfile(caminho_recurso.rstrip('/') + '/' + i):
            tamanho_bytes = str(os.path.getsize(caminho_recurso.rstrip('/') + '/' + i))
        else:
            tamanho_bytes = '-'
```

```
        listagem += '<div class="item_listagem">'
        listagem += '<a href="' + recurso.rstrip('/') + '/' + i + '">' + i + '</a>'
        listagem += '<span class="tam_bytes">' + tamanho_bytes + '</span>'
        listagem += '<span class="mod_date">' + data_modificacao + '</span>'
        listagem += '</div>'
```

```
else:
    codigo_resposta = '404'
    txt_resposta = 'Not Found'
    conteudo_resposta = HTML_NOT_FOUND
```

```
# Cabeçalho de resposta HTTP
cabecalho_resp = 'HTTP/1.1 {0} {1}\nContent-Type: {2}\n\n'.format(
    codigo_resposta,
    txt_resposta,
    content_type
)
```

```
# Envia o cabeçalho para o cliente
con.send(cabecalho_resp.encode())
```

```
# Envia o conteúdo para o cliente
TAM_BUFFER = 1024
```

```
# Codifica o conteúdo caso o mesmo não esteja em formato binário
if isinstance(conteudo_resposta, str):
    conteudo_resposta = conteudo_resposta.encode()
```

```

for i in range(0, len(conteudo_resposta), TAM_BUFFER):
    try:
        con.send(conteudo_resposta[i:i + TAM_BUFFER])
    except BrokenPipeError:
        pass

# Fecha a conexão
con.close()

if __name__ == '__main__':

    # Obtém a porta e o diretório por linha de comando
    try:
        porta = int(sys.argv[1])
        diretorio = sys.argv[2]

        # Checa se é um diretório válido
        if not os.path.isdir(diretorio):
            print('Forneça um diretório válido.')
            sys.exit()
    except:
        print('Modo de uso: python3 servidor.py porta diretorio')
        sys.exit()

    while True:
        # Instancia o socket TCP IPv4
        s = socket.socket(
            socket.AF_INET,
            socket.SOCK_STREAM
        )

        # Permite que a porta do servido seja utilizada sucessivas vezes,
        # sem necessitar de aguardar um tempo de espera
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

        try:
            s.bind((ip_servidor, porta))
        except PermissionError:
            print('Você não possui permissão para utilizar essa porta.')
            sys.exit()

        s.listen(1)

        print('Aguardando conexão em {0}:{1}'.format(ip_servidor, porta))

        con, info_cliente = s.accept()

        print('Conexão efetuada por', ':'.join([str(i) for i in info_cliente]), '\n')

        with open(ARQUIVO_LOG, 'a') as f:
            f.write(datetime.datetime.now().strftime('%d/%m/%Y %H:%M:%S') + ';')
            f.write(':'.join([str(i) for i in info_cliente]) + ';')

        # Processa a requisição em uma thread paralela
        Thread(target=processa_requisicao, args=(con,)).start()

```

Referências:

[1] KUROSE, James F. **Redes de computadores e a Internet**: uma abordagem top-down. São Paulo: Pearson Education do Brasil, 2013. xxii, 634 p., il. Inclui bibliografia e índice. ISBN 9788581436777.

[2] Python Magic Disponível em: <<https://pypi.org/project/python-magic/>>. Acessado em: 24-04-2021

[3] Socket – Low-level networking interface. Disponível em: <<https://doscs.python.org/3/library/socket.html>>. Aceesado em 24-04-2021.