Relatório

1. Primeiro nós precisamos importar as bibliotecas necessárias:

```
from socket import *
import os.path
from os import path
```

2. Determinaremos o endereço do nosso servidor Host que será o próprio endereço local do computador. Para isso, passamos uma string vazia.

```
myHost = ''
```

3. Agora vamos declarar qual porta o nosso servidor ira usar. Vamos escolher a porta 50004.

```
portNumber = 50004
```

5. Também precisamos definir um formato para decodificar e codificar nossas mensagens. Usaremos o formato UTF-8.

```
format = ‘utf-8'
```

6. Vamos agora declarar o tamanho do nosso Header em bytes. Ou seja, o tamanho da mensagem esperada. Para os nossos testes, utilizaremos um tamanho de 1024 bytes.

```
headerSize = 1024
```

7. Na linha seguinte, nós criamos o nosso objeto socket. Passamos dois parâmetros. O primeiro, AF_INET indica que vamos usar um endereço da família ipv4. O segundo, SOCK_STREAM, indica que queremos usar o protocolo TCP para trocas de mensagens.

```
socketObject = socket(AF_INET, SOCK_STREAM)
```

8. Em seguida, usamos o método bind para indicar qual endereço de host e porta o nosso objeto socket vai usar. (Localhost e porta 50004).

```
socketObject.bind((myHost, portNumber))
```

9. Agora usaremos o método listen, passando 1 como parâmetro, para indiciar que o nosso socket receberá apenas uma conexão por vez.

```
socketObject.listen(1)
```

10. Dentro de um laço Enquanto (que fica sempre executando para esperar novas conexões com clientes), chamamos o método accept para aceitar uma nova conexão. Esse método retorna um socket conexão e um endereço que identifica o cliente.

```
connection, address = socketObject.accept()
```

11. Em seguida, vamos salvar os dados recebidos desse cliente em uma variável. Caso nenhum dado seja recebido, podemos cancelar a conexão

```
data = connection.recv(headerSize)
if not data: break
```

12. Se nós conseguirmos recuperar os dados, nós vamos decodificar essa string em UTF-8 e faremos um split a cada espaço “ ” para termos um array com cada palavra da requisição. Depois, pegaremos a segunda palavra, que é o que foi passado após o endereço Home no navegador e por fim, removeremos a “/“ inicial para termos o nome do arquivo que o cliente quer solicitar do servidor.

```
fileURL = data.decode(format).split(" ")[1][1:]
```

13. Se o servidor conseguir localizar o arquivo, vamos abrir esse arquivo, ler ele, e então usar a função splitlines para transforma a string lida em um array da linhas originais do arquivo. Depois, vamos mandar uma mensagem header 200 OK e iterar esse array para mandar mensagens com cada linha para o cliente. Em seguida, vamos fechar o arquivo aberto.

```
if(path.exists(fileURL)):
      f = open(fileURL, "r")
      outputdata = f.read().splitlines()
      connection.send("HTTP/1.1 200 OK\r\n\r\n".encode(format))
for i in outputdata:
      connection.send(f"<html><head></head><body><p>{i}</p></
body></ html>\r\n".encode(format))
f.close()
```

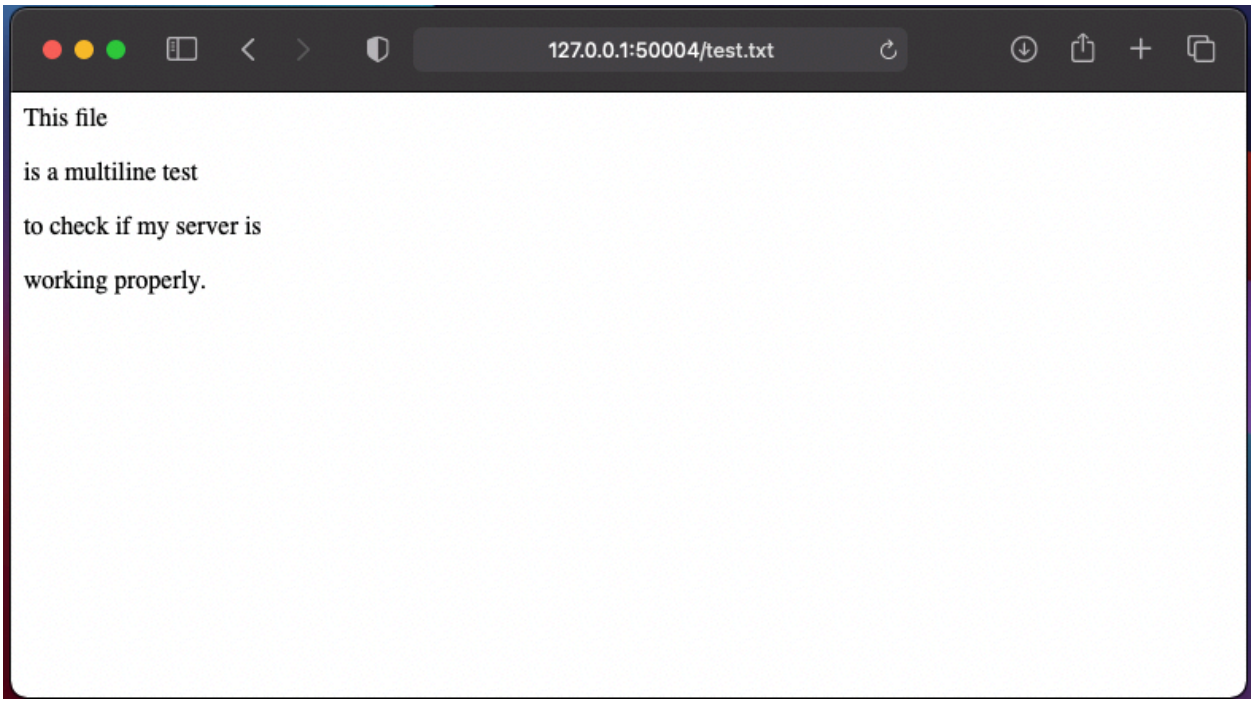
14. Caso o arquivo não seja localizado, mandaremos uma mensagem de error 404 para o cliente.

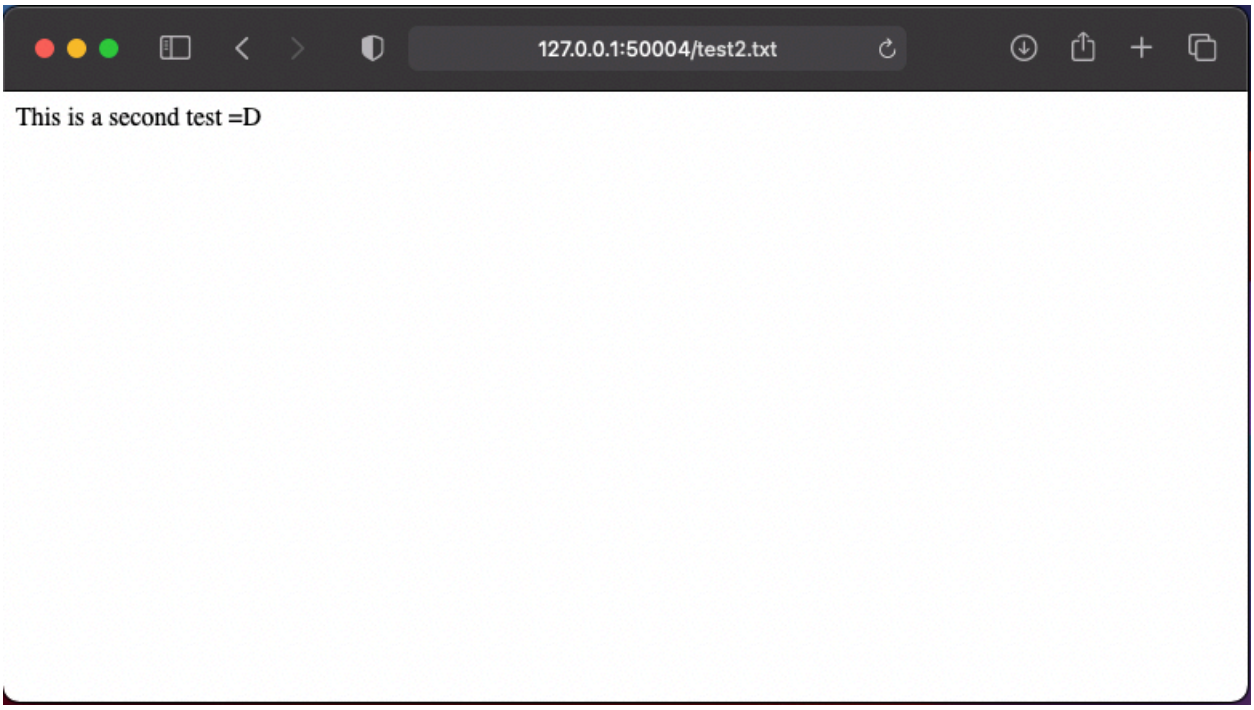
```
else:
        connection.send("HTTP/1.1 404 Not 
Found\r\n\r\n".encode(format))
        connection.send("<html><head></head><body><h1>404 Not 
Found</h1></body></html>\r\n".encode(format))
```

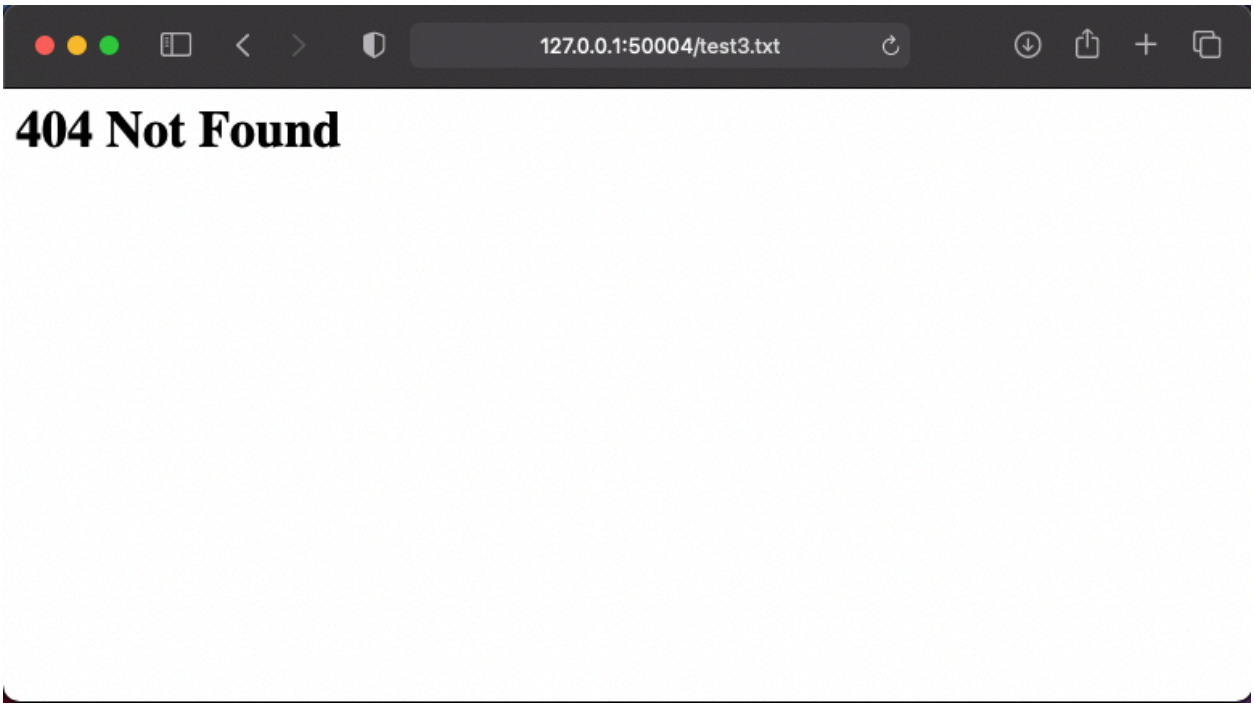
15. Sempre ao final, encerraremos a conexão com o cliente.

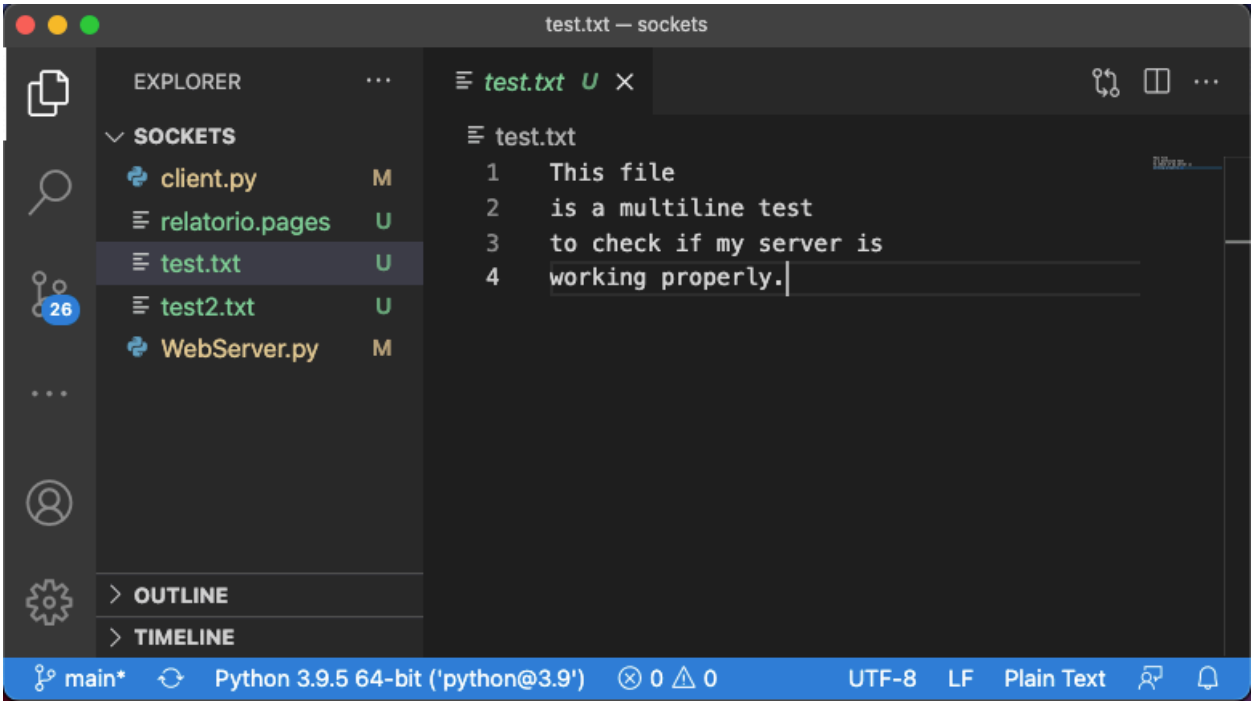
```
connection.close()
```

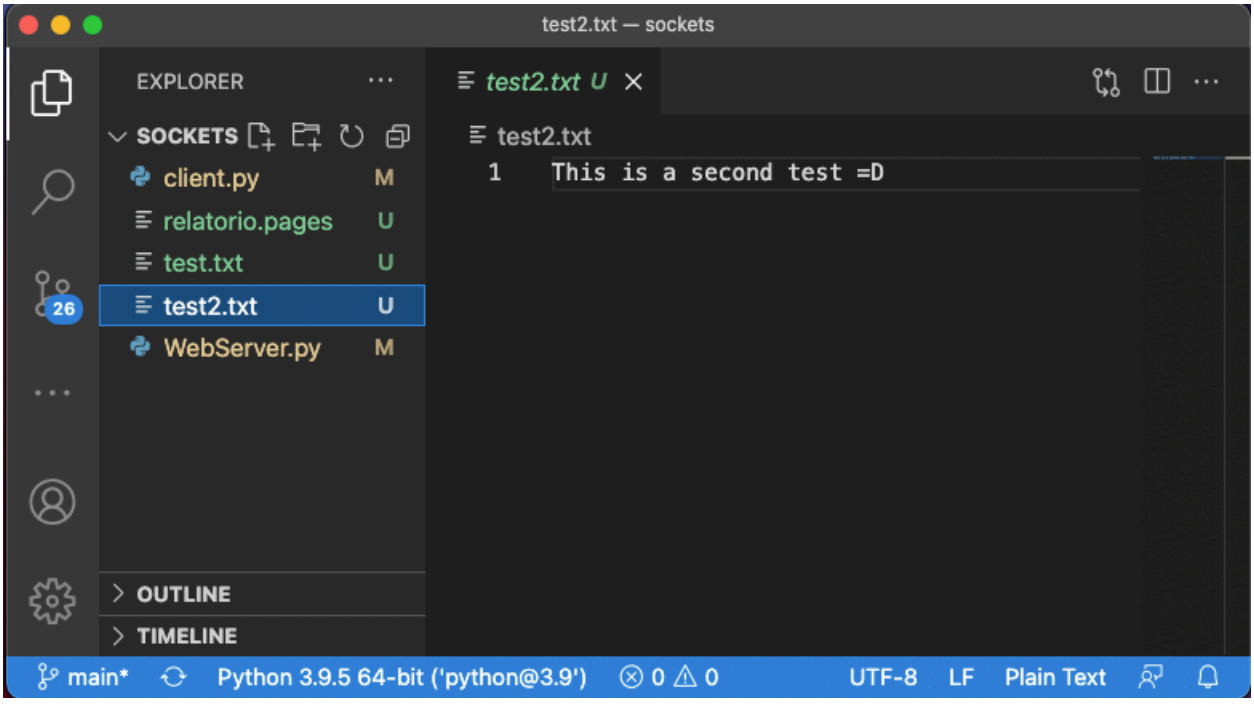
Testes




127.0.0.1:50004/test2.txt
This is a second test =D


127.0.0.1:50004/test3.txt
404 Not Found





Jose Douglas Gondim Soares, 485347