EXERCÍCIO PROGRAMA DE REDES

SERVIDOR WEB MULTITHREADED – ENTREGA A

BRUNO SALERNO ROCHA – 9004525

CAIO TAVARES CRUZ – 8921840

TURMA 94

SÃO PAULO

2016

Objetivo

Nessa primeira etapa, o objetivo do trabalho foi criar um servidor web *multithreaded* responsável apenas por receber solicitações HTTP e exibí-las na saída padrão (texto). Isso significa que as requisições não foram tratadas nem respondidas.

Execução

A execução restringiu-se à codificação e aos testes. Para a codificação, foram utilizados como materiais de estudo a especificação do trabalho, que continha a estrutura das classes e um passo-a-passo sobre como construir o código, o material disponibilizado para a disciplina no e-tidia, que continha uma explicação sobre a utilização de sockets e um modelo de utilização de sockets em Java, e a API Java 7, que continha uma breve explicação sobre implementação de *multithreading* em Java.

A construção utilizada foi a seguinte:

```
Classe ServidorWeb (thread principal)

     1) Abre um canal de comunicação na porta 6789
     2) Fica "esperando" requisições HTTP
          a. Para cada requisição HTTP que receber, abre
             uma nova thread e põe a classe HttpRequest
             para tratar a requisição na nova thread.


Classe HttpRequest

     1) "Lê" os dados da requisição em aberto
     2) Imprime a linha de requisição e o cabeçalho da
        mensagem de solicitação
     3) Fecha o canal de comunicação (socket)
```

O texto acima é apenas um pseudocódigo do programa codificado, que encontra-se documentado na pasta "Codigo".

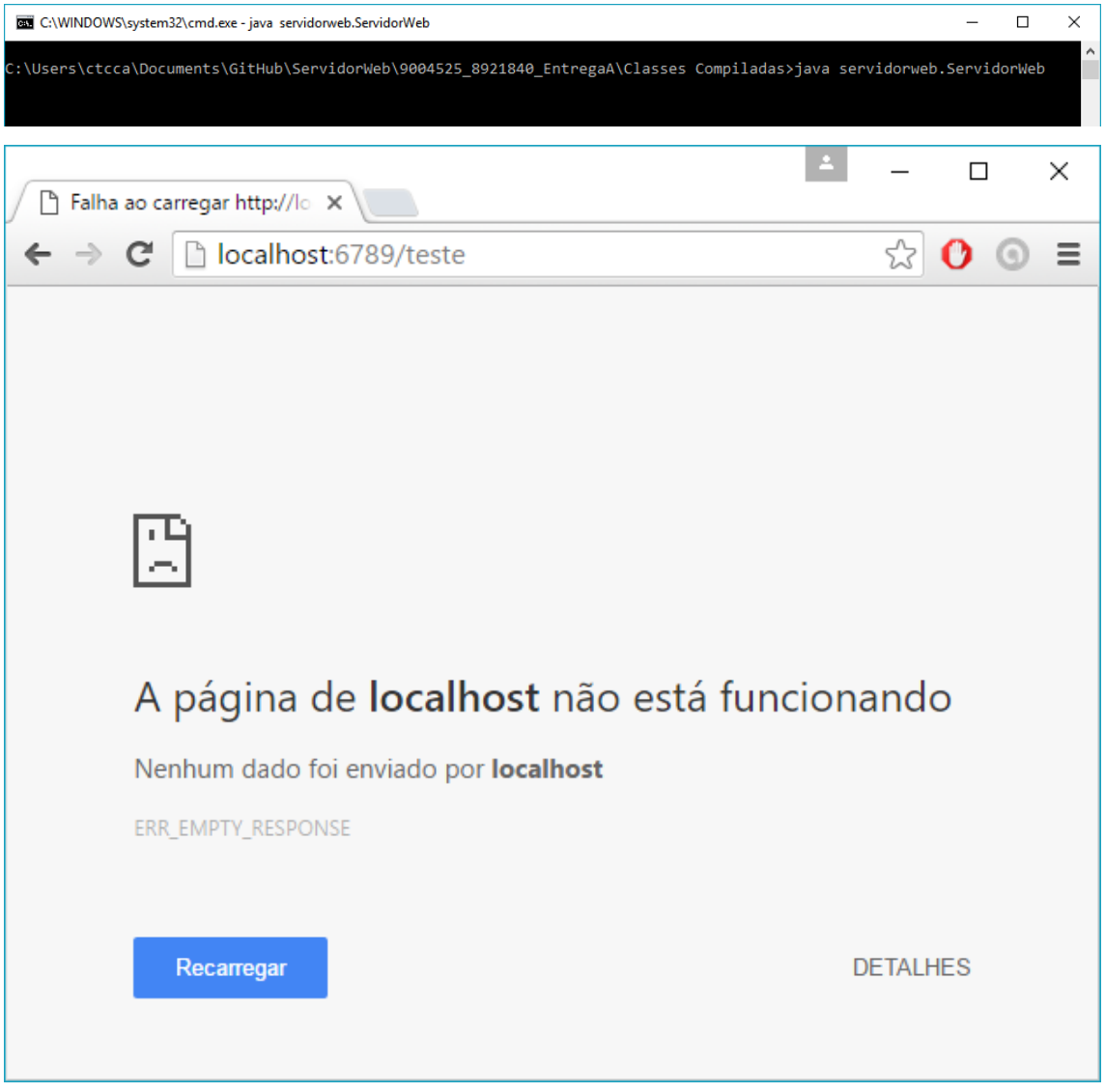
Resultados

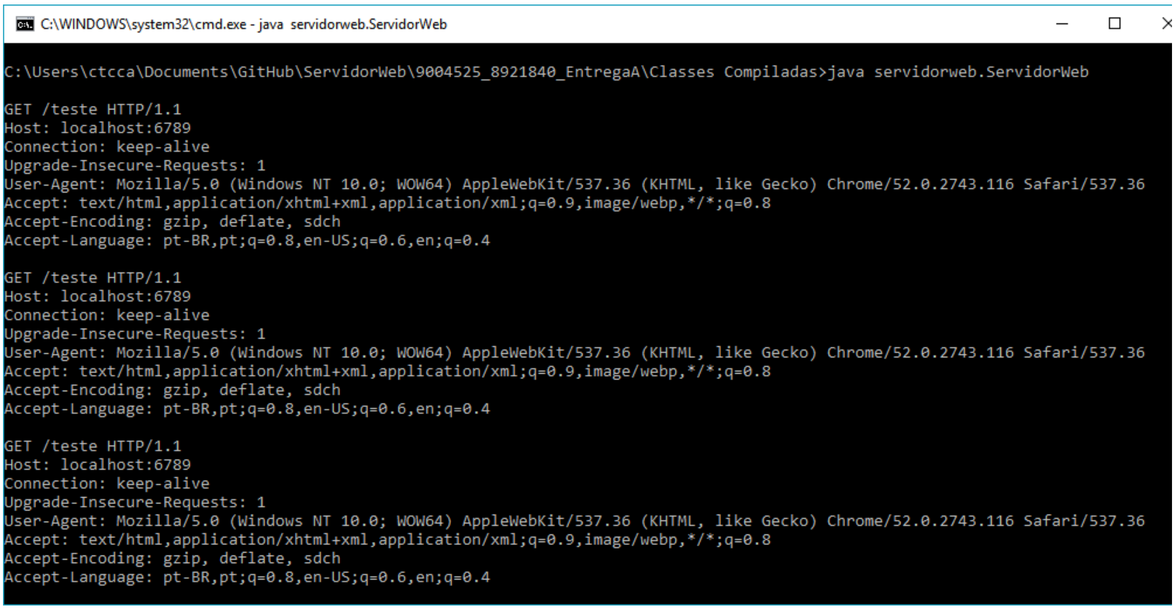
Depois da codificação e da compilação, foram feitos testes de funcionamento, para verificar se o programa estava de acordo com a especificação.

O teste segue o seguinte protocolo:

1) Iniciar a classe ServidorWeb
2) Fazer uma requisição (via browser) para a máquina local, porta 6789
3) Verificar o que aparece na saída padrão

Os testes apontaram a corretude dos resultados em detrimento da especificação. Um exemplo de caso de teste é especificado a seguir:



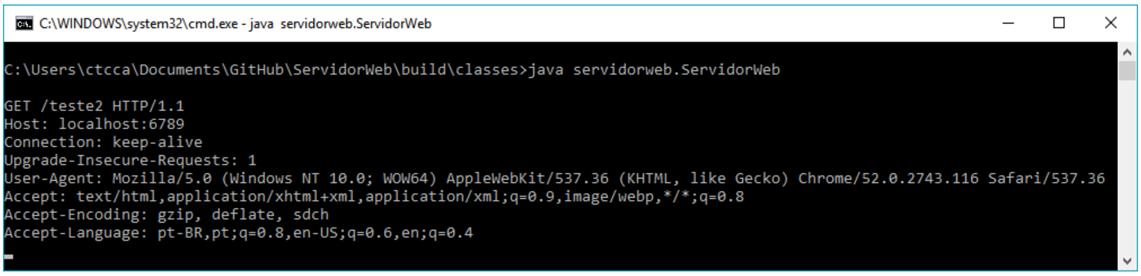


```
C:\WINDOWS\system32\cmd.exe - java  servidorweb.ServidorWeb

C:\Users\ctcca\Documents\GitHub\ServidorWeb\9004525_8921840_EntregaA\Classes Compiladas>java servidorweb.ServidorWeb

GET /teste HTTP/1.1
Host: localhost:6789
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4

GET /teste HTTP/1.1
Host: localhost:6789
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4

GET /teste HTTP/1.1
Host: localhost:6789
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4
```

Observações

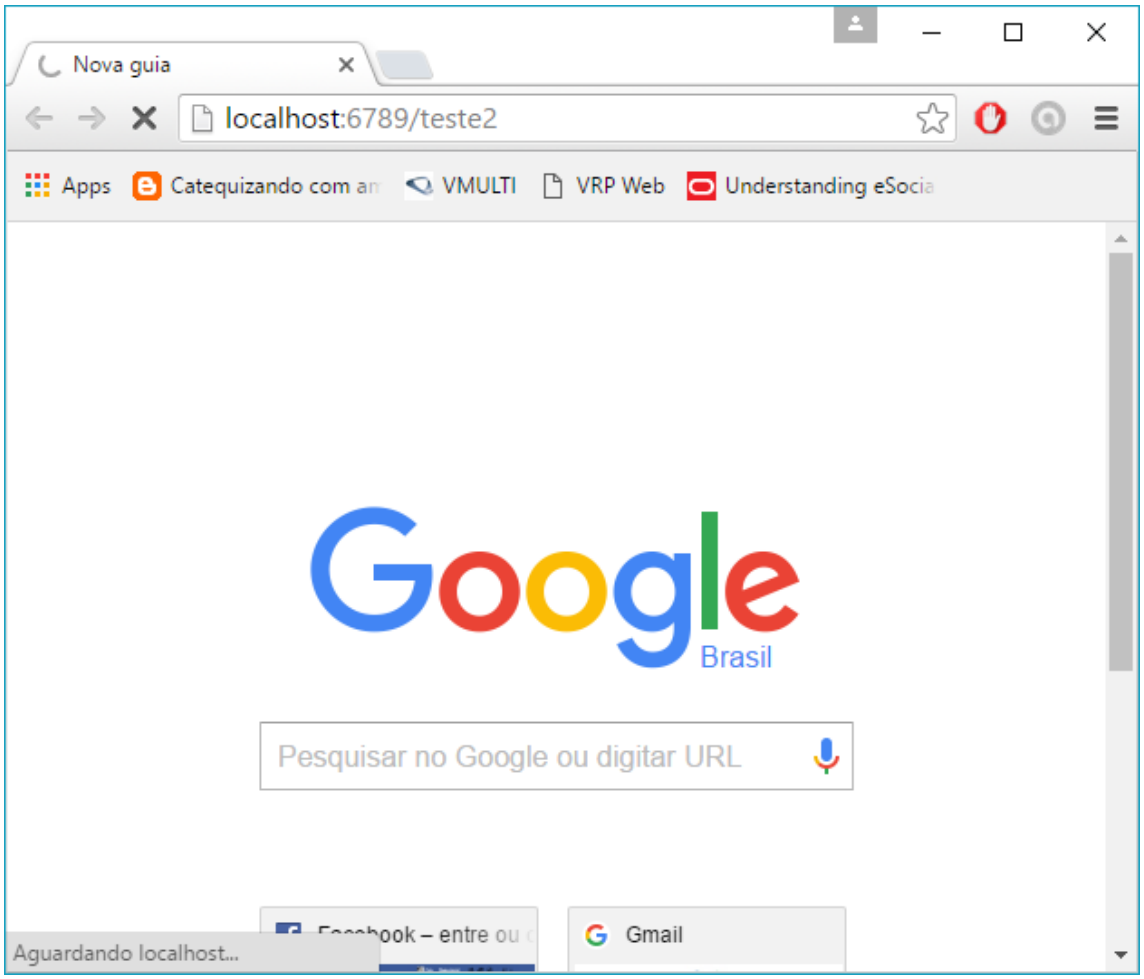
Observamos que embora o funcionamento estivesse correto, para cada requisição que fazíamos no *browser*, apareciam cerca de três requisições na saída do servidor. Levantamos a hipótese de que, como o servidor fecha o *socket* sem retornar nada, o *browser* reenvia as requisições (tenta mais algumas vezes) antes de mostrar para o usuário a resposta de que nenhum dado foi recebido.

Para testar essa hipótese, removemos do código a linha que encerra o *socket* e as streams de entrada e saída refizemos esse protocolo de testes. Obtendo o seguinte resultado:



```
C:\WINDOWS\system32\cmd.exe - java  servidorweb.ServidorWeb

C:\Users\ctcca\Documents\GitHub\ServidorWeb\build\classes>java servidorweb.ServidorWeb

GET /teste2 HTTP/1.1
Host: localhost:6789
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4
```

Apenas uma requisição foi exibida, e a página, ao invés de mostrar mensagem de erro, ficou esperando indefinidamente por uma resposta: “aguardando localhost”:



Os resultados desse teste são um indício que nos leva a crer que nossa hipótese estava correta.

Manual de Execução

A pasta do presente relatório (9004525_8921840_EntregaA), contém, além deste arquivo e do manual em *txt* dois diretórios:

- Codigo: Contém os arquivos *.java* (arquivos de codigo) referentes à implementação dessa entrega;
- Classes Compiladas: Contém os arquivos *.class* (compilados) referentes à implementação dessa entrega.

As duas classes pertencem ao pacote servidorweb. Para executar a aplicação via terminal:

- Direcionar até o diretório das classes compiladas, por exemplo: C:\Users\ctcca\Documents\GitHub\ServidorWeb\9004525_8921840_EntregaA\Classes Compiladas
- Escrever: java servidorweb.ServidorWeb
- A partir daí a aplicação já estará rodando, basta fazer os testes, como na evidência de testes acima.