Roteiro de Aula Prática – Criação de Servidor HTTP	
DISCIPLINA:	DCA0130 – Redes de Computadores
PROFESSOR:	Carlos Manuel Dias Viegas

Esta prática consiste em uma introdução ao desenvolvimento de um servidor HTTP utilizando Sockets. O objetivo deste trabalho é colocar em prática os conhecimentos apresentados em videoaula a respeito da camada de aplicação e o protocolo HTTP. Vocês irão criar um servidor HTTP que deverá responder ao comando `GET` enviado por um cliente. O servidor deverá responder pedidos via terminal `telnet` e também via navegador.

- Os requisitos para a realização desta prática são a instalação do Python na versão 3 e ter assistido às videoaulas sobre o desenvolvimento de aplicações de rede e protocolo HTTP disponibilizadas no SIGAA;
- É importante lembrar que esta prática deve ser gravada em vídeo, apresentando o funcionamento do que foi desenvolvido. Como sugestão, assistam às instruções no seguinte vídeo: https://youtu.be/0zzt2QWjedY (outros softwares de captura de vídeo podem ser utilizados);
- Durante a gravação do vídeo, à medida que as tarefas forem sendo realizadas, vocês devem narrar como estão fazendo e os resultados encontrados. Nesta prática em específico, pretende-se que seja apresentado o funcionamento do servidor. **O vídeo NÃO PODE ter duração superior a 5:00 minutos;**
- O vídeo deverá ser submetido até o dia 06/07/2020 na plataforma indicada pelo professor. O link para o vídeo deverá ser informado em uma tarefa específica no SIGAA.
- O código fonte desenvolvido também deverá ser enviado na tarefa criada no SIGAA, juntamente com o link.

Nesta prática, deverá ser utilizado um navegador e também um terminal `telnet` (como clientes) para acessar o servidor HTTP por vocês desenvolvido. A execução pode ser feita em um mesmo computador (ou seja, rodando o cliente e o servidor no mesmo) ou em computadores/dispositivos diferentes (o cliente rodando em um computador e o servidor em outro). Esta escolha fica a critério de vocês.

O código fonte (em Python) necessário para iniciar esta prática está disponível na seguinte página:

`https://www.dca.ufrn.br/~viegas/disciplinas/DCA0130/files/Sockets/HTTPserver/`

Lembrando que, obrigatoriamente, deverá ser utilizado como base o código fonte disponibilizado acima!

VERIFICAÇÃO DOS REQUISITOS PARA A PRÁTICA

Abrir um *prompt* de comando (`cmd` ou outro terminal) e verificar se o comando `telnet` funciona. Caso não funcione, apresento como sugestão (para Windows) o uso do terminal *MobaXterm* que já contém o `telnet`. Neste caso, pode-se baixar o *MobaXterm* no seguinte endereço: `https://goo.gl/MeJDA9`

Em seguida, deve-se extrair o .zip e executar o '`terminal.exe`' e verificar se o comando `telnet` funciona.

TAREFAS

1. Execute o servidor HTTP fornecido e entenda o seu funcionamento:
   a. Abra um navegador e acesse o endereço do servidor:
      `http://127.0.0.1:8080`
      Repare que a página exibirá sempre a mensagem "Hello, World!", independentemente do que for solicitado pelo cliente.

   b. Abra um terminal e acesse por meio de `telnet` acesse o endereço do servidor:
      ```
      telnet 127.0.0.1 8080
      GET / HTTP 1.1
      ```
      Após o GET, dê dois "Enter". Será exibido o cabeçalho da mensagem (`HTTP/1.1 200 OK`) e também "Hello, World!". Repare que independentemente do comando digitado, o servidor retornará sempre a mesma resposta.

2. Após realizar os testes acima, analise o código fonte da aplicação com o uso de um editor de texto ou IDE de programação;

3. Faça as alterações necessárias no código para que o servidor seja capaz de processar um pedido `GET` de um cliente e retornar um arquivo .html para o mesmo;
   a. Crie um arquivo `index.html` (na mesma pasta em que o servidor estiver sendo executado) com o código abaixo (como exemplo) para ser usado como resposta:

      ```
      <html>
         <head><title>Este é o meu servidor!</title></head>
         <body>
               <h1>Olá mundo!</h1>
               O meu servidor funciona!
         </body>
      </html>
      ```

   b. Quando um pedido é feito ao servidor, a variável `request` (linha 35) recebe os dados solicitados. Esta variável deve ser inspecionada para que se possa analisar o que está sendo pedido. É a partir dela que devem ser tratados os casos abaixo solicitados.
      DICA: Esta variável `request` pode ser segmentada em pedaços de informação menores e consultados por índice (vetor).

   c. A sintaxe do pedido `GET` deve seguir a especificada pelo protocolo HTTP:
      `GET /caminho HTTP/versão`
      Por exemplo:
      `GET /arquivo.html HTTP/1.1`
      Por exemplo (caso o arquivo esteja em uma pasta):
      `GET /pasta/arquivo.html HTTP/1.1`
      Outro exemplo:
      `GET / HTTP/1.1`
      Neste caso quando não é especificado o arquivo no pedido, o servidor deve procurar pelo `index.html`.

      IMPORTANTE: Ao fazer um pedido com `GET`, nunca se deve inserir o caminho do arquivo no sistema de arquivos do sistema operacional, pois o HTTP não sabe interpretar esse caminho (exemplo a **NÃO** usar:

`C:\Documentos\pasta\arquivo.html` ou `/home/usuario/arquivo.html`). Deve-se inserir apenas o caminho do arquivo em relação à pasta em que o servidor HTTP está rodando (caminho relativo).

d. Caso o caminho solicitado não exista, o servidor deve retornar o código 404 - página não encontrada. Este retorno deve seguir a especificação do protocolo HTTP, onde são enviados dois comandos: um para ser interpretado apenas pelo navegador (primeira linha) e outro para ser lido pelo cliente (segunda linha em diante), como o exemplo abaixo. A primeira linha JAMAIS pode ser salva em um arquivo .html. Deve ser retornada pelo servidor antes do conteúdo/corpo da página.

```
HTTP/1.1 404 Not Found\r\n\r\n
<html>
      <head></head>
      <body>
            <h1>404 Not Found</h1>
      </body>
</html>
\r\n
```

e. Caso algum outro comando diferente de `GET` seja digitado, o servidor deve tratar a exceção e retornar "comando desconhecido" e continuar aguardando por novos comandos. Neste caso, o erro deve ser:

```
HTTP/1.1 400 Bad Request\r\n\r\n
<html>
      <head></head>
      <body>
            <h1>400 Bad Resquest</h1>
      </body>
</html>
\r\n
```

f. Recorde-se que caso a página seja realmente encontrada, o servidor deve retornar (na primeira linha) o comando `HTTP/1.1 200 OK\r\n\r\n`. E só em seguida retornar o conteúdo da página html. A primeira linha JAMAIS pode ser salva em um arquivo .html. Deve ser retornada pelo servidor antes do conteúdo/corpo da página;

g. Sempre que um cliente enviar um comando `GET` para o servidor, este deve imprimir na tela (do terminal) todo o comando enviado pelo cliente. Ou seja, a variável `request` deve ser apresentada na tela do terminal que estiver executando o servidor;

h. É importante lembrar que, por omissão, quando se faz um pedido diretamente à raiz (isto é: `GET / HTTP/1.1`), sem especificar uma página, o servidor deve retornar a página padrão (geralmente o `index.html`).

Atenção: O servidor nunca deverá ser encerrado. Apenas os clientes terão as conexões abertas por mais tempo ou encerradas logo após um pedido.

4. Faça os testes do funcionamento do servidor usando o `telnet`, bem como o navegador. É importante notar que o navegador faz mais de um pedido por vez, pois solicita um arquivo `favicon.ico` (arquivo de ícone para a aba). É necessário tratar esta situação na implementação do servidor.

5. O objetivo ao final da implementação é que o servidor receba e responda às solicitações dos clientes, independentemente do arquivo solicitado e do software utilizado para o pedido.