

Projeto Server HTTP Simples

Jhonatas Santana dos Anjos, Luiz Sacramento²

Departamento de Ciências Exatas e da Terra
Universidade do Estado da Bahia (UNEB) – Salvador, BA – Brasil

jhsantana11@hotmail.com

Abstract. *Python and own libraries is the main technology use to build it. Its low learning curve contributes significantly to learning, but there are also many solutions on the internet, and it is widely used worldwide. Through bibliography fonts was possibly made this work about HTTP (Hypertext Transfer Protocol). It used to find inside too many technologies in daily life.*

Resumo. *A linguagem de programação escolhida para a construção deste projeto foi o Python, utilizando-se algumas bibliotecas feitas para tal. Sua baixa curva de aprendizagem contribui significativamente para esse processo. Além disso, também há muitas soluções na internet amplamente utilizadas mundialmente. Baseando-se nas fontes bibliográficas, foram encontrados scripts prontos e linhas de comando que permitem a criação de servidores HTTP(Hipertext Transfer Protocol).*

1. Informações gerais

O trabalho consiste em montar um Servidor HTTP que simplesmente faz uma requisição e resposta ao cliente. “A HTTP or web server processes requests via HTTP, a network protocol used to exchange information on the World Wide Web (WWW). The main function of a HTTP server is to store, process and deliver web pages to clients. Pages delivered are usually HTML documents, which may include images, style sheets and scripts in addition to text content. Using HTML, you describe what a page must look like, what types of fonts to use, what color the text should be, where paragraph marks must come, and many more aspects of the document.”(ARM KEIL, 2021)

Essa citação explica o funcionamento da tecnologia em questão. Em linhas gerais, opera como um protocolo de transferência de hipertexto e usa uma linguagem de marcação em consonância à essa estrutura. Assim como, descreve a natureza dos arquivos que transitam nesse processo. Se trata da base para inúmeras aplicações, sejam elas sites, aplicativos web e etc.

2. Primeira página

“Segundo, um servidor web fornece suporte para HTTP (protocolo de transferência de hipertexto). Como o próprio nome indica, o HTTP especifica como transferir arquivos de hipertexto (ou seja, documentos vinculados da web) entre dois computadores. Um *protocolo* é um conjunto de regras para comunicação entre dois computadores. HTTP é um protocolo textual sem estado.”(Mozillla, 2021)

Essa explicação será pauta dos próximos pontos mais técnicos sobre o objeto de estudo, descrições que utilizaram códigos, imagens e fluxogramas ilustrativos. A primeira vista, parece ser algo totalmente distante, mas ao decorrer da explicação fica evidente que a grande maioria das aplicações do dia a dia utilizam um sistema semelhante.

3. Endereçando processos

Na web comunicação acontece entre pares de processos denominados de processo cliente e servidor. Essa comunicação acontece graças aos endereços. O hospedeiro (processo cliente e servidor) possui um endereço IP exclusivo. Para enviar uma mensagem é necessário o endereço de destino, também é necessário identificar o processo receptor. Um número de porta atende a esta finalidade. O endereço IP e o número para porta forma um socket, um direcionamento para envio da mensagem.

4. Serviços de Transporte disponíveis para aplicações

A escolha do protocolo da camada de transporte é essencial para aplicação devido a troca de mensagens entre cliente-servidor. Sua escolha tem que levar em consideração as necessidades da aplicação. O envio de mensagens é feito pelo remetente através de um socket. Assim, é responsabilidade da camada de transporte é transportar a mensagem até o socket do destinatário. Ao escolher a camada de transporte e dependendo da natureza da aplicação deve-se levar em consideração a segurança, transferência confiável de dados, vazão e temporização entre outras coisas.

5. A Web e o HTTP

A princípio a Internet era somente usada por acadêmicos, pesquisadores e estudantes universitários. Então, com a implementação World Wide Web, a partir da década de 1990, a Internet começou a se formar para o que ela é hoje.

5.1 HTTP (HyperText Transfer Protocol)

O HTTP é descrito e definido pelas RFCs 1945 e 2616, é executado nos processos cliente e servidor, e estes, por sua vez, são executados em sistemas finais diferentes que trocam mensagens HTTP entre si.

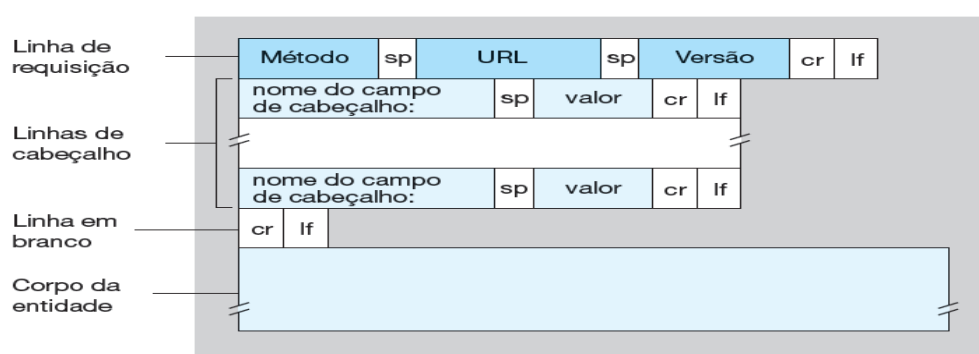
Como ilustrado na Figura 1 quando o processo cliente solicita uma página Web, o Browser instalado no computador envia ao processo servidor mensagem de requisição HTTP para os objetos (arquivos) da página. Então, por sua vez, o processo servidor recebe as requisições e envia a resposta contendo os objetos solicitados.

Para que as requisições e respostas aconteçam, primeiramente, o processo cliente HTTP estabelece uma conexão TCP com o processo servidor. Por meio de suas interfaces socket, os processos cliente e servidor acessam o TCP. A mensagem é enviada através da interface socket e é recebida pela mesma. A mensagem chegará ao destino intacta, pois o TCP oferece transferência de dados confiável.

O HTTP não precisa se preocupar com dados perdidos ou com detalhes de como o TCP se recupera da perda de dados ou os reordena dentro da rede. Essa é a tarefa do TCP e dos protocolos das camadas mais inferiores da pilha de protocolos. Além disso, como o servidor HTTP não mantém informação alguma sobre clientes, o HTTP é denominado um protocolo sem estado. (Kurose, 2013).

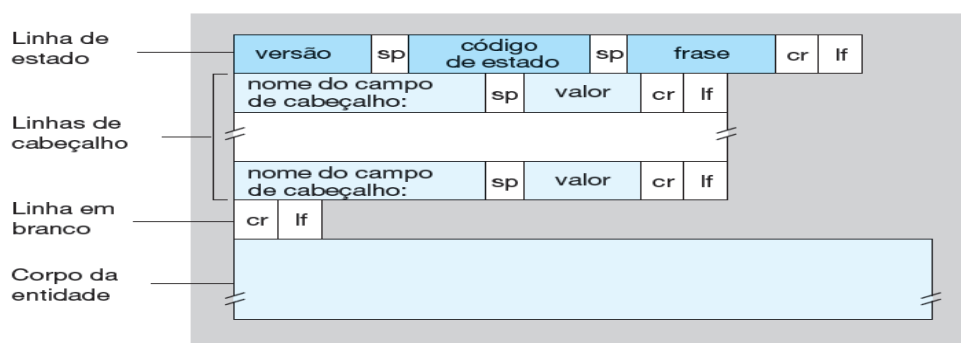
5.1.2 Formato de mensagem HTTP

Figura 1. Formato Geral da mensagem de requisição HTTP.



Fonte: Kurose, 2013

Figura 2. Formato Geral da mensagem de resposta HTTP.



Fonte: Kurose, 2013

6. Funcionamento

Servidor web estático: é a maneira mais simplória de se construir um *web service* (*servidor web*), tendo uma estrutura que possui apenas um cliente e um servidor para troca de mensagens. Em contrapartida aos Servidores web dinâmicos, se diferenciam por atualizar os dados dos arquivos antes de enviá-los, muito comum em aplicações de grande porte.

Pode acontecer de uma solução ter um pouco dos dois recursos citados acima. Nos estáticos a vantagem é a facilidade de se configurar, por sua vez são excelentes para fins didáticos. Os dinâmicos, por via de regra baseiam-se em processamento de conteúdo ou os gera através de um banco de dados (algo rotineiro em websites).

7. Configurações

O código do Server HTTP é escrito em python, foi executado no jupyter notebook. A biblioteca os foi importada para capturar o caminho do arquivo html em forma de string, para montar o lista de string foi chamada a biblioteca sys que é usada para passar de forma automática uma lista de string que, por sua vez, representação argumentos em linha de comando. Para processar a entrada do usuário foi implementado a biblioteca cgi e para marcar a data e hora das operações do servidor é usada a módulo time. BaseHTTPRequestHandler é a classe para solicitações HTTP que chegam ao servidor. Já a classe HTTPServer se baseia na TCPServer classe em que armazena o endereço do servidor como variável de instância. O módulo http.server contém as classes BaseHTTPRequestHandler e HTTPServer.

8. Especificações técnicas

A Figura 3 representa o estabelecimento de comunicação entre Browser e Web Server. A figura 4 representa os serviços de um Server HTTP. Na subseção 5.1 é apresentado o código fonte do servidor em python. As funções `def do_GET` e `do_POST` seu nome são padrões da classe `BaseHTTPRequestHandler` presente na biblioteca `Http.server`. A função `do_GET` é usada para renderização de qualquer arquivo. Por exemplo, temos uma `cwd.html` em nossa pasta `htdocs`, ele vai trabalhar pegando o arquivo `cwd.html` e renderizando o html dele.

Em seguida é criada a função `def main(NameVirtualHost)`, a variável `NameVirtualHost` é onde passa o IP e porta para o Server HTTP.

Na biblioteca `http.server` tem uma função onde o mesmo server, para rodar o servidor, recebe dois parâmetros: o primeiro é IP/PORTA, e o segundo é a classe `http`, onde estão as funções (GET, POST etc).

É chamada a função `def main`. É definido a varável `DocumentRoot` que captura a pasta local e concatena com `“/htdocs/”`. É definida a porta padrão `“8000”` e o host `“localhost”` que é o host padrão.

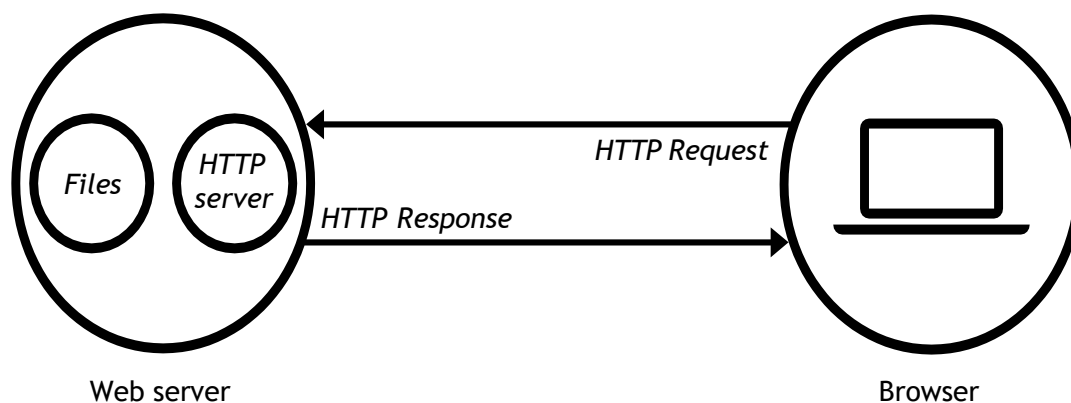
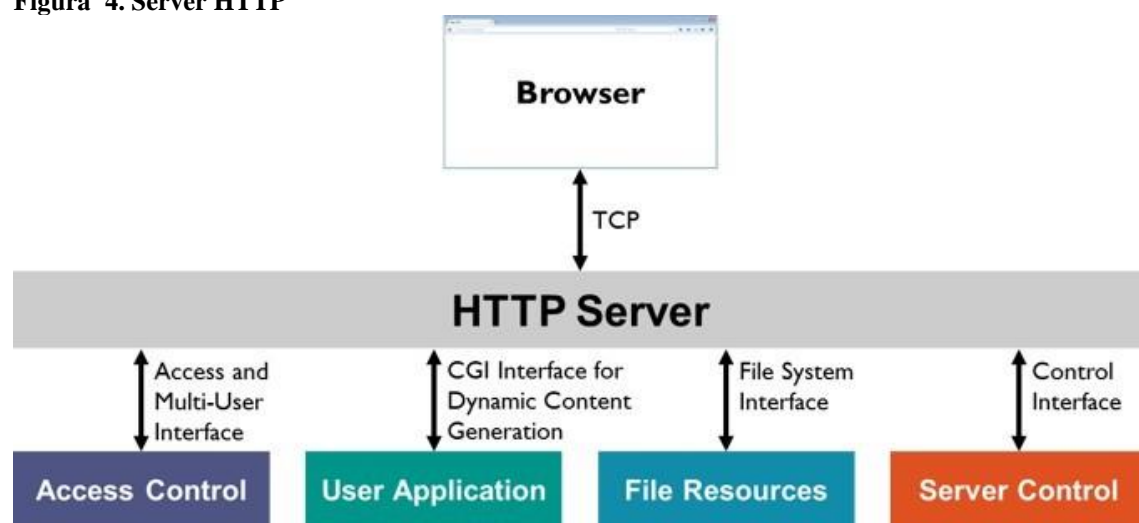


Figura 3. Browser e Web Server

Nesta figura, é enfatizada, a representação da comunicação entre o Browser (processo cliente) e Server HTTP (processo servidor). Através de uma conexão orientada o processo cliente estabelece, primeiramente, uma comunicação com o processo servidor por meio de protocolo e endereços. O processo cliente envia uma requisição (HTTP Request) ao processo servidor através de um dos protocolos de transporte escolhido para aplicação. Faz isso por endereços. O processo servidor, por sua vez, responde (HTTP Response) a requisição enviada com o envio do arquivo armazenado ou com uma mensagem de erro e a comunicação é encerrada.

Figura 4. Server HTTP



Fonte: Criação Keil, 2021

A imagem nos mostra que o Server HTTP estabelece comunicação com o Browser através do Protocolo TCP. O Server HTTP disponibiliza seus serviços ao processo cliente. Nesta figura, o Server HTTP representado tem filtro de hosts que não tem permissão para se conectar ao Server HTTP, adição de contas de usuários adicionais e gerencia de direito de acesso para cada usuário. Interface de controle para explicar como start / stop o Server HTTP e gerenciar contas de usuário integradas. Interface do sistema de arquivos que detalha as funções que são usadas para ler / gravar dados no dispositivo de armazenamento do Server HTTP.

8.1 Especificação do código

```
#Bibliotecas importadas
import os
import sys,os
import string,cgi,time
from http.server import BaseHTTPRequestHandler, HTTPServer

# Captura uma string com o diretório atual
```

```

cwd = os.getcwd()
#c = os.ctermid()
#print(cwd)

#Classe http com a função do_GET e do_POST
class http(BaseHTTPRequestHandler):
    def do_GET(self):
        try:
            if self.path.endswith(".html"):
                f = open(DocumentRoot + self.path)
                self.send_response(200)
                self.send_header('Content-type','text/html')
                self.end_headers()
                self.wfile.write(f.read())
                f.close()
            return
            if self.path.endswith(".esp"):
                self.send_response(200)
                self.send_header('Content-type','text/html')
                self.end_headers()
                self.wfile.write("hey, today is the" + str(time.localtime()[7]))
                self.wfile.write(" day in the year " + str(time.localtime()[0]))
                return
            return
        except IOError:
            self.send_error(404,'File Not Found: %s' % self.path)

    def do_POST(self):
        global rootnode
        try:
            ctype, pdict = cgi.parse_header(self.headers.getheader('content-type'))
            if ctype == 'multipart/form-data':
                query=cgi.parse_multipart(self.rfile, pdict)
                self.send_response(301)
                self.end_headers()
                upfilecontent = query.get('upfile')
                print("filecontent", upfilecontent[0])
                self.wfile.write("<html>Post OK. <br /><br />");
                self.wfile.write(upfilecontent[0]);
                self.wfile.write("</html>")
            except:
                pass

# Função criada para chamar as declarações da classe.
def main(NameVirtualHost):
    try:
        virtualhost = str.split(NameVirtualHost,":")
        if virtualhost[0] == "*":
            virtualhost[0] = ""
        server = HTTPServer((virtualhost[0], int(virtualhost[1])), http)

```

```

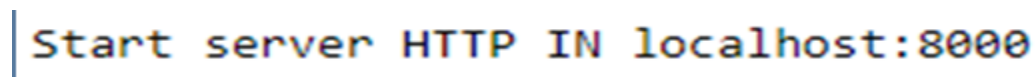
print('Start server HTTP IN %s' % NameVirtualHost)
server.serve_forever()
except KeyboardInterrupt:
print('Shutting down server HTTP')
server.socket.close()

#Chama a função def main
if __name__ == '__main__':
DocumentRoot = "%s/htdocs/" % os.path.realpath(os.path.dirname(cwd))
PORT = "8000"
HOST = "localhost"
try :
main(sys.argv[1])
except :
main("%s:%s" % (HOST,PORT))

```

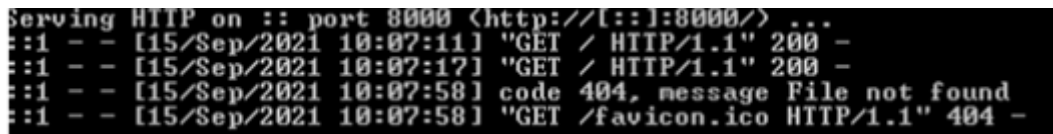
9. Resultados alcançados

Foi passado como argumento a string com o caminho do arquivo para depois ser passa como uma lista de string. Ao implementar o Server HTTP foi dado o Start Server HTTP in localhost: 8000 como definido previamente no código ao chamar a função def main. O servidor permanece em execução até dar um shutting down acionado pelo programador.



A terminal window with a black background and white text. The text reads: "Start server HTTP IN localhost:8000".

Figura 5. Start do Server HTTP



A terminal window with a black background and white text. The text shows the server's log of HTTP requests:

Serving HTTP on :: port 8000 (http://[::]:8000/) ...

:1 - - [15/Sep/2021 10:07:11] "GET / HTTP/1.1" 200 -

:1 - - [15/Sep/2021 10:07:17] "GET / HTTP/1.1" 200 -

:1 - - [15/Sep/2021 10:07:58] code 404, message File not found

:1 - - [15/Sep/2021 10:07:58] "GET /favicon.ico HTTP/1.1" 404 -

Figura 6. GET in Server HTTP

O Server HTTP foi requisitado através do localhost:8000, respondendo em seguida com a página do diretório. Também foi mostrado como retorno a recusa do servidor em estabelecer conexão.

10. Considerações finais

O trabalho em questão, foi para entender o processo comunicação cliente-servidor. Primeiramente, ao executar o código do Server HTTP deu erros de sintaxe, pois o jupyter notebook usa a versão python 3 e o código estava escrito com sintaxe de versões anteriores. Além disso, tivemos que importar o módulo http.server, não presente originalmente no código, pois chamando a biblioteca BaseHTTPServer dava erro por mesclada ao módulo em questão. Ao importar o http.server e passar como argumento a string com o caminho do arquivo o servidor foi implemetado.

11. Referências

- ALECRIM, Emerson. Conhecendo o Servidor Apache (HTTP Server Project). Disponível em: <<https://www.infowester.com/servapach.php>> Acesso em 13/09/2021
- ALVELINO, Thiago. Criando um servidor HTTP (Web) com Python. Disponível em: <<https://imasters.com.br/back-end/criando-um-servidor-http-web-com-python>> Acesso em 16/09/2021
- Arm keil. Network Component. Disponível em: <https://www.keil.com/pack/doc/mw/Network/html/group__net_h_t_t_pfunc.html> Acesso em 22/09/2021
- Crie um servidor HTTP simples em poucos minutos. Disponível em: <<https://sempreupdate.com.br/crie-um-servidor-http-simples-em-poucos-minutos/>> Acesso em 13/09/2021
- Exemplo Python: Código do Python(server.py). Disponível em: <https://docs.aws.amazon.com/pt_br/polly/latest/dg/example-Python-server-code.html> Acesso em 16/09/2021
- KRUNAL. Python3 SimpleHTTPServer: Como Usar SimpleHTTPServer. Disponível em: <<https://appdividend.com/2019/02/06/python-simplehttpserver-tutorial-with-example-http-request-handler/>> Acesso em 12/09/2021
- Kurose, James F. Redes de computadores e a Internet: uma abordagem top-down/ James F. Kurose, Keith W. Ross ; tradução Daniel Vieira; revisão técnica Wagner Luiz Zucchi. – 6. ed. – São Paulo: Pearson Education do Brasil, 2013.
- Mozilla .O que é um servidor web (web server)?. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/Common_questions/What_is_a_web_server> Acesso em 22/09/2021