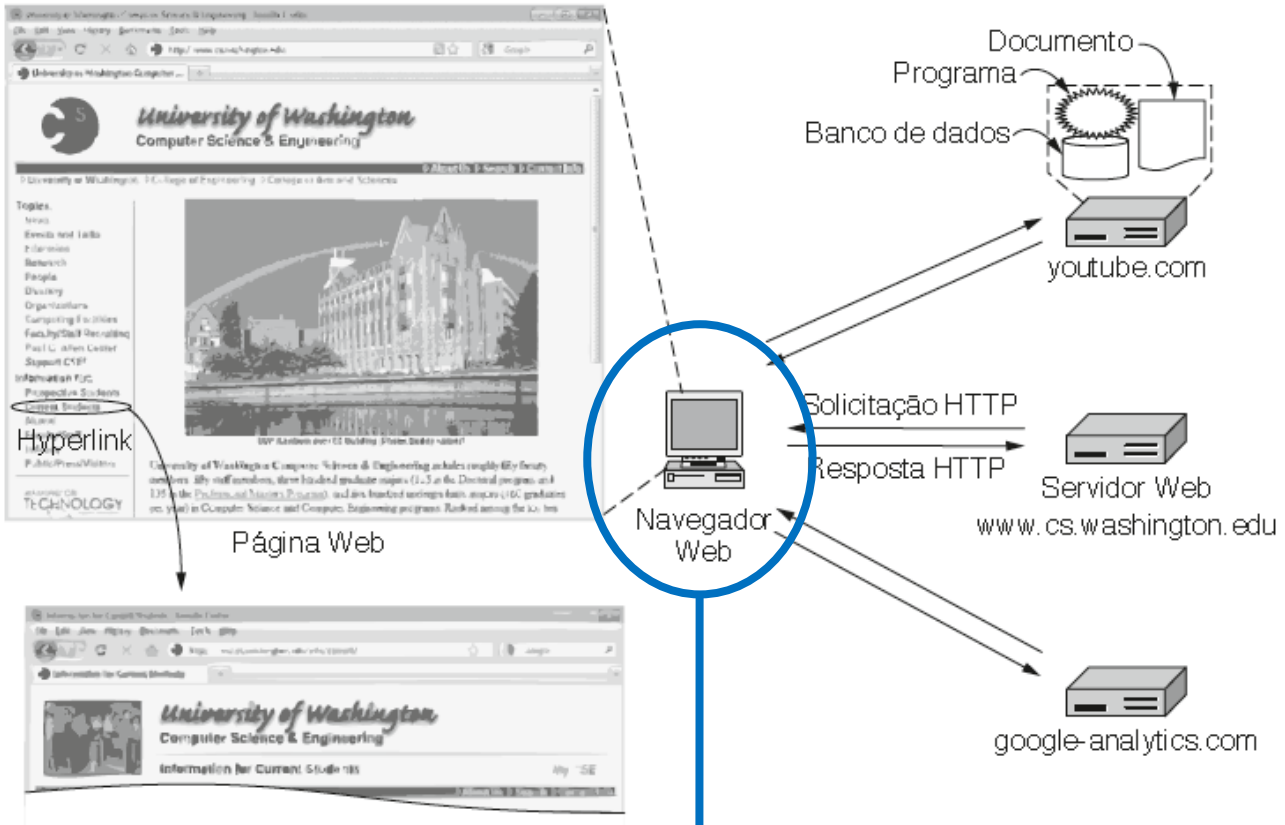
Computação Distribuída

Odorico Machado Mendizabal

Universidade Federal de Santa Catarina – UFSC
Departamento de Informática e Estatística – INE

Aplicações para Web

Visão Geral – Organização da Web



Múltiplos servidores, com propósitos diversos

Servidores mantém conteúdo e atendem requisições dos clientes

Uma única requisição normalmente retorna múltiplos conteúdos (ex. arquivo HTML, JPEG, fluxo de áudio, etc.)

Programa que executa no lado cliente.
Gerencia o conteúdo obtido dos servidores de aplicação

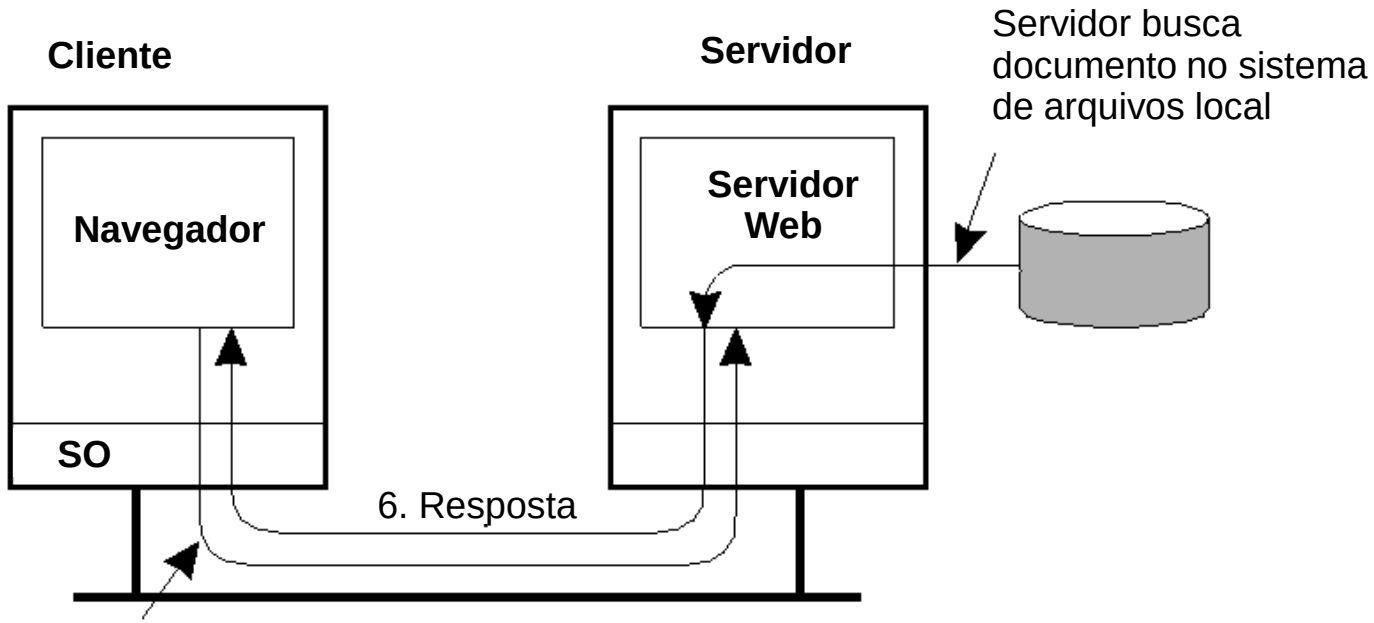
Clientes acessam conteúdo Web através do protocolo HTTP

Recuperação do conteúdo via Web

- Algumas questões sobre como “abrir” uma página Web:

1) Como buscar uma página?
2) Onde a página está localizada?
3) Como acessar a página?

Recuperação do conteúdo via Web



1) O navegador identifica a URL
2) O navegador solicita ao serviço de DNS o endereço IP do servidor
3) O navegador obtém resposta DNS
4) O navegador estabelece uma conexão TCP
5) O navegador envia uma solicitação HTTP para a página URL
6) O servidor envia a página como resposta HTTP
7) O navegador apresenta a página
8) As conexões TCP são encerradas

Uniform Resource Location (URL)

URLs são compostas por 3 partes:
- Protocolo
- O nome DNS
- O caminho que localiza o recurso no servidor destino

Algumas URLs comuns:

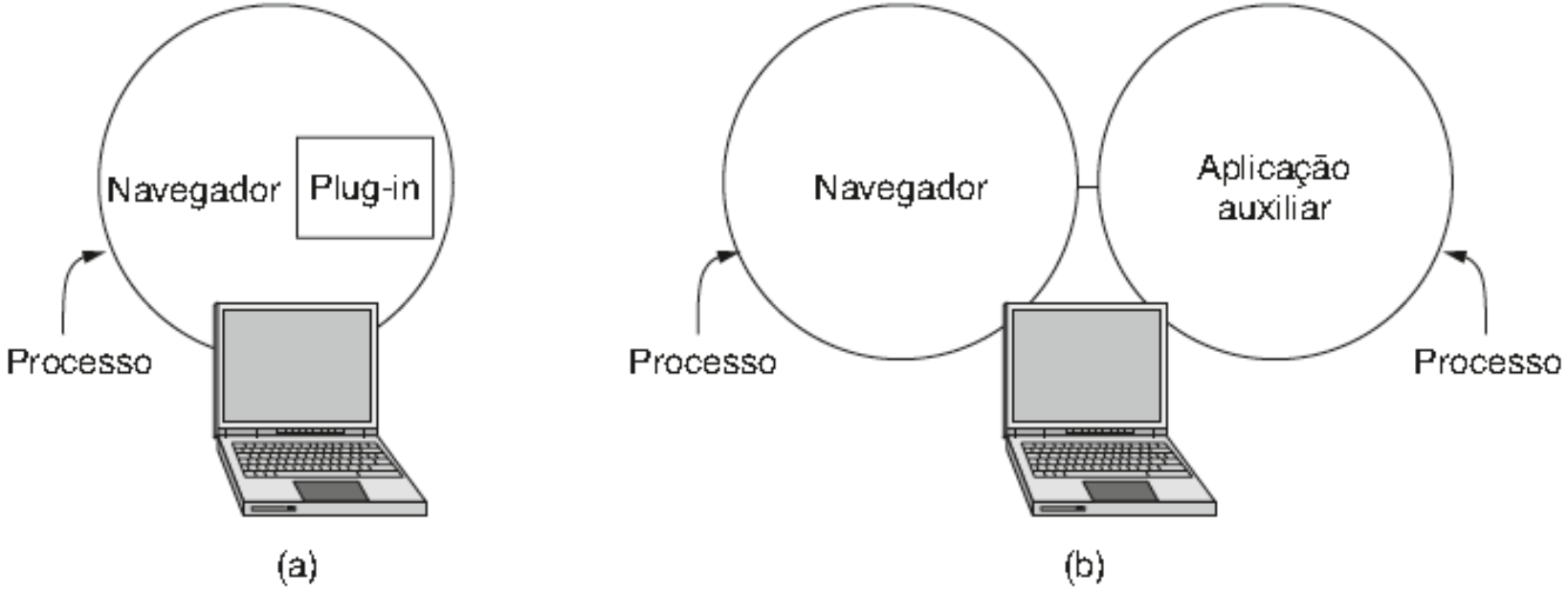
Nome	Usado para	Exemplo
http	Hipertexto (HTML)	http://www.ee.uwa.edu/~rob/
https	Hipertexto com segurança	https://www.bank.com/accounts/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Arquivo local	file:///usr/suzana/prog.c
mailto	Envio de e-mail	mailto:JoaoSilva@acm.org
rtsp	Streaming de mídia	rtsp://youtube.com/montypython.mpg
sip	Chamadas de multimídia	sip:eve@adversary.com
about	Informação do navegador	about:plugins

Configurações do navegador: about:config

O navegador

O navegador é um aplicativo que permite acessar o conteúdo de URLs

Em algumas situações, o navegador não dispõe de recursos necessários para executar o conteúdo recuperado de um servidor (ex.: algum formato de vídeo ou áudio, um jogo, etc.)



O servidor de aplicações

O servidor de aplicação atende e gerencia requisições de múltiplos clientes

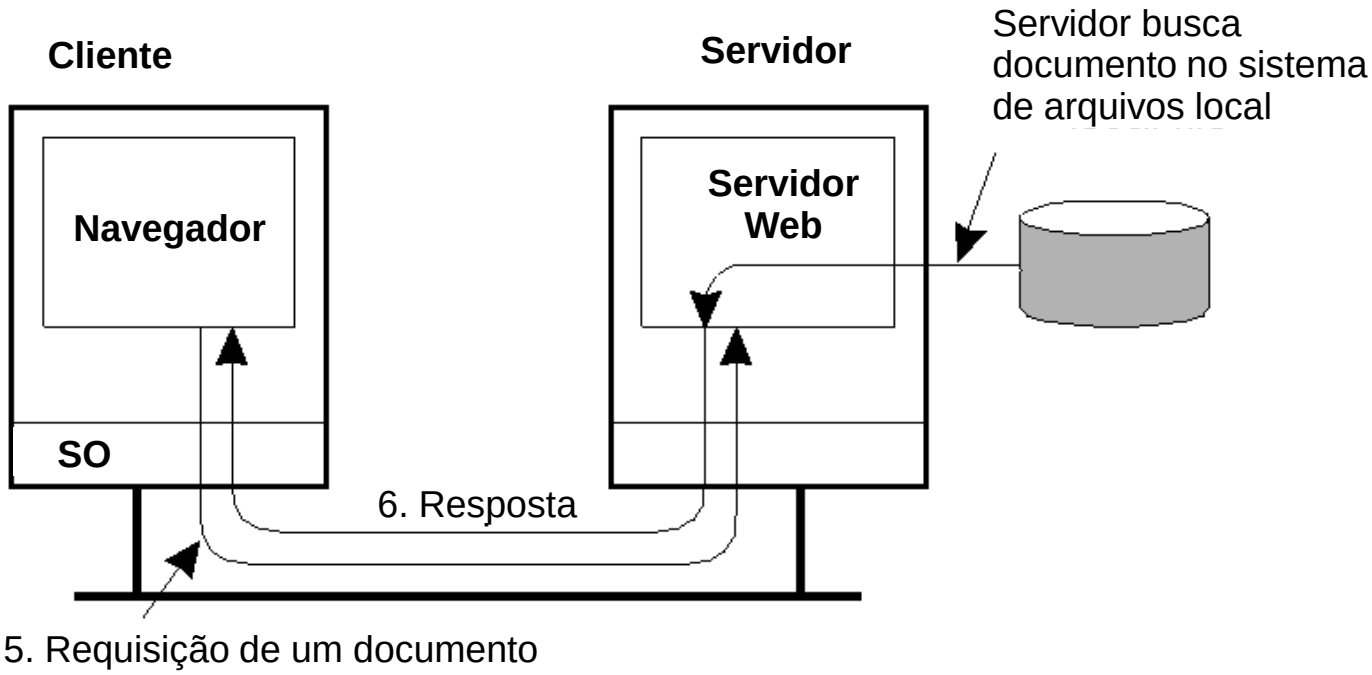
As requisições normalmente seguem formatos bem definidos, de acordo com protocolos:

Exemplos:

Servidor HTTP (protocolo HTTP)
Apache, IIS, Nginx, GlassFish,

Servidor de email (protocolos POP, POP3, SMTP, IMAP, ..)
Gmail, Hotmail, Mailjet, Amazon SES

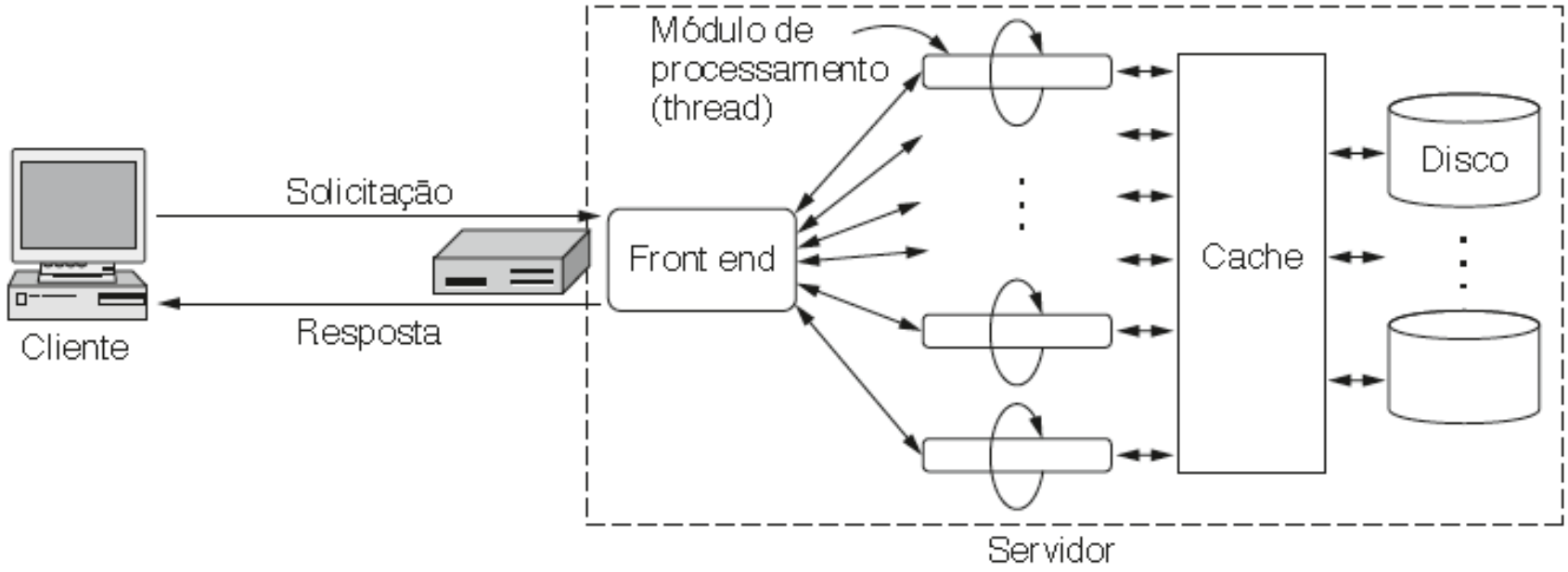
Servidor FTP (protocolo FTP)
ColoradoFTP, Wing FTP server, Apache Mina FTP



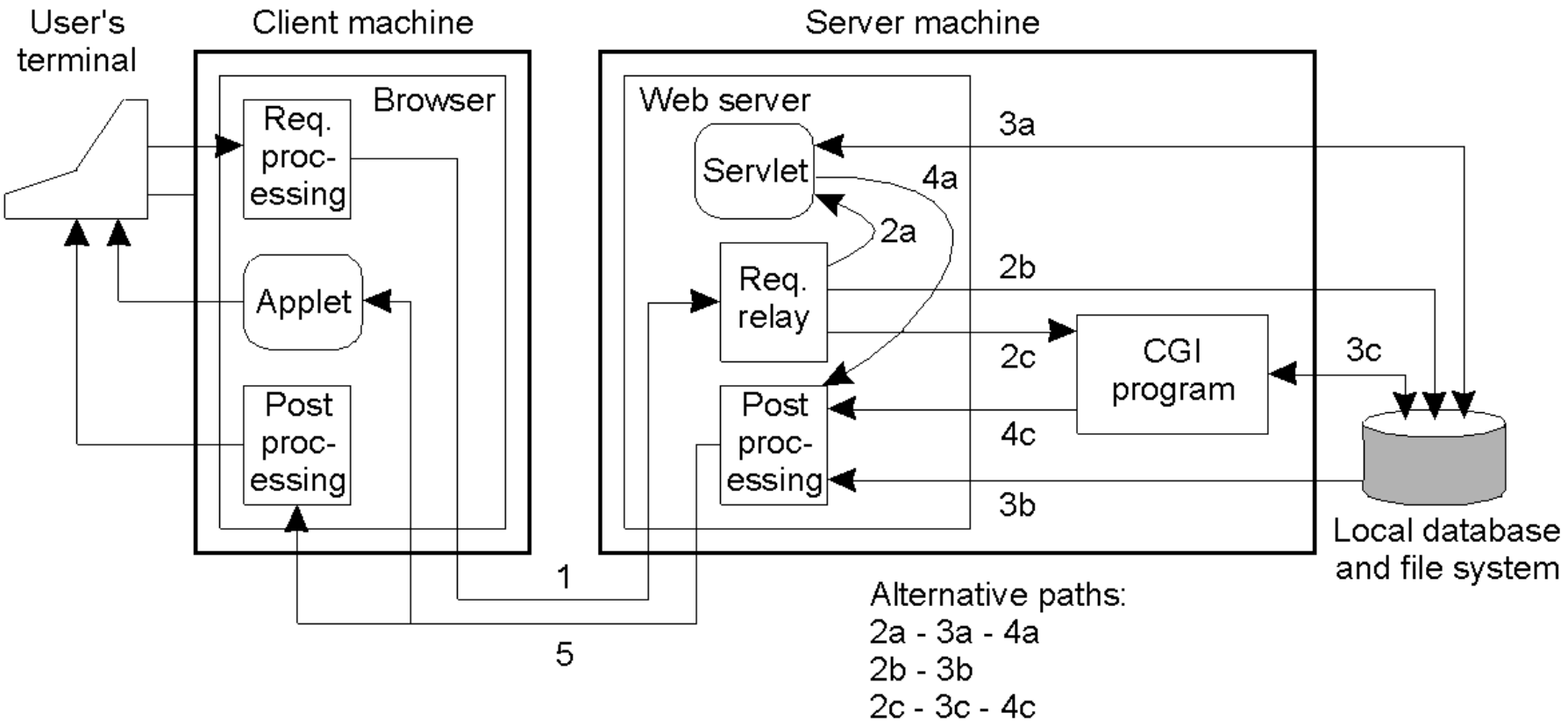
O servidor de aplicações

Visão simplificada dos passos realizados pelo servidor em um laço principal:

1) Aceitar conexão TCP do cliente
2) Obter caminho para página, nome do arquivo requisitado
3) Obter arquivo (do disco)
4) Enviar conteúdo do arquivo ao cliente
5) Encerrar conexão TCP



Na prática: arquitetura Cliente / Servidor multi-camadas



Páginas Web Estáticas – HTML

- **H**yper **T**ext **M**arkup **L**anguage
- Conjunto de marcações (*tags)*
- Não é uma linguagem de programação, e sim de marcação

Alguns *tags*:

Cabeçalho

De **<h1>** Cabeçalho 1 **</h1>** até **<h6>** Cabeçalho 6 **</h6>**

Parágrafo

** ** Isso é um parágrafo. ** **

Link

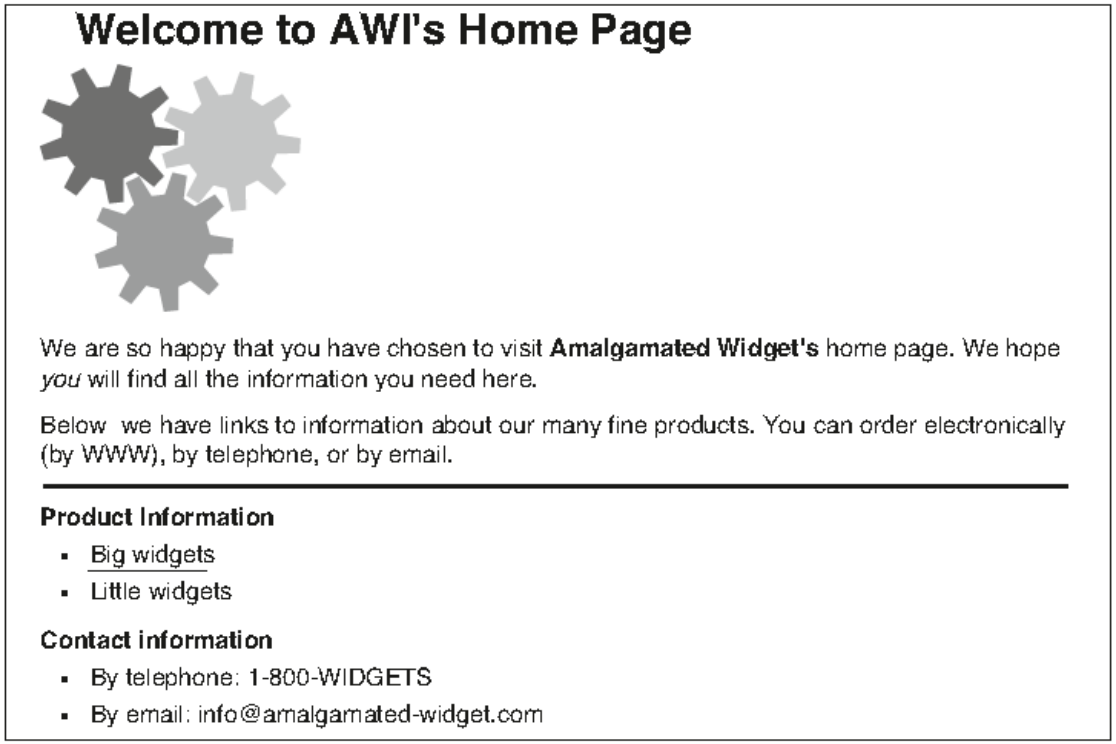
**** Tudo sobre filmes****

Imagem

****

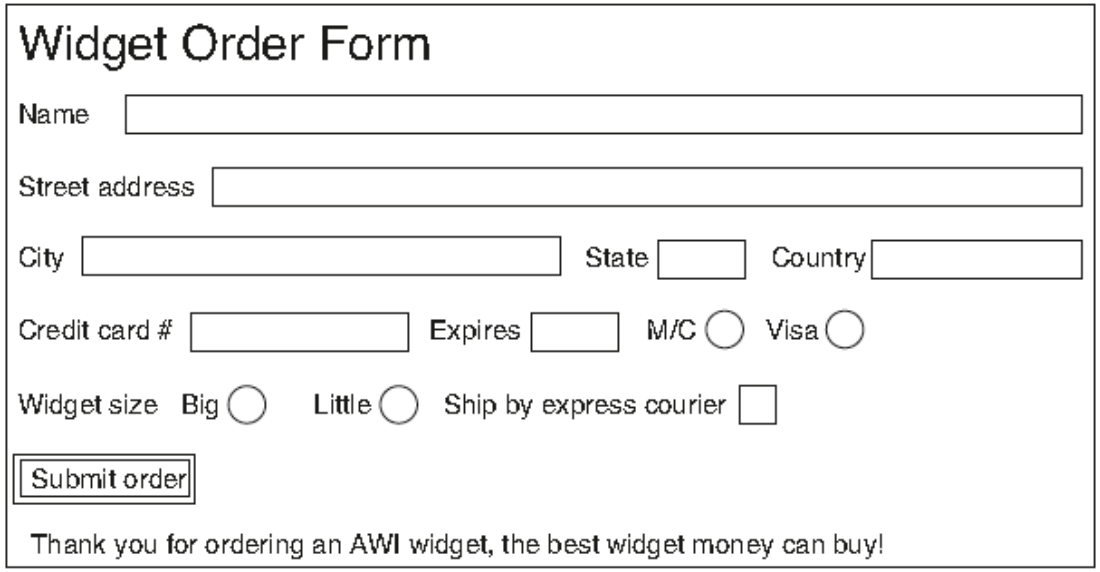
Páginas Web Estáticas – HTML: Exemplos

```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI's Home Page </h1>
<img src = "http://www.widget.com/images/logo.gif = "ALT"AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by email. </p>
<hr>
<h2> Product information </h2>
<ul>
    <li> <a href = "http://widget.com/products/big"> Big widgets </a> </li>
    <li> <a href = "http://widget.com/products/little"> Little widgets </a> </li>
</ul>
<h2> Contact information </h2>
<ul>
    <li> By telephone: 1-800-WIDGETS </li>
    <li> By email: info@amalgamated-widget.com </li>
</ul>
</body>
</html>
```



Páginas Web Estáticas – HTML: Exemplos

```
<html>
<head> <title> AWI CUSTOMER ORDERING FORM </title> </head>
<body>
<h1> Widget Order Form </h1>
<form ACTION = "http://widget.com/cgi-bin/order.cgi" method = POST>
<p> Name <input name = "customer" size = 46 </p>
<p> Street address <input name = "address" size = 40> </p>
<p> City <input name = "city" size = 20> State <input name = "state" size = 4>
Country <input name = "country" size = 10> </p>
<p> Credit card # <input name = "cardno" size = 10>
Expires <input name = "expires" size = 4>
M/C <input name = "cc" type = radio value = "mastercard">
VISA <input name = "cc" type = radio value = "visacard"> </p>
<p> Widget size Big <input name = "product" type = radio value = "expensive">
Little <input name = "product" type = radio value = "cheap">
Ship by express courier <input name = "express" type = checkbox> </p>
<p> <input type = submit value = "Submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>
```

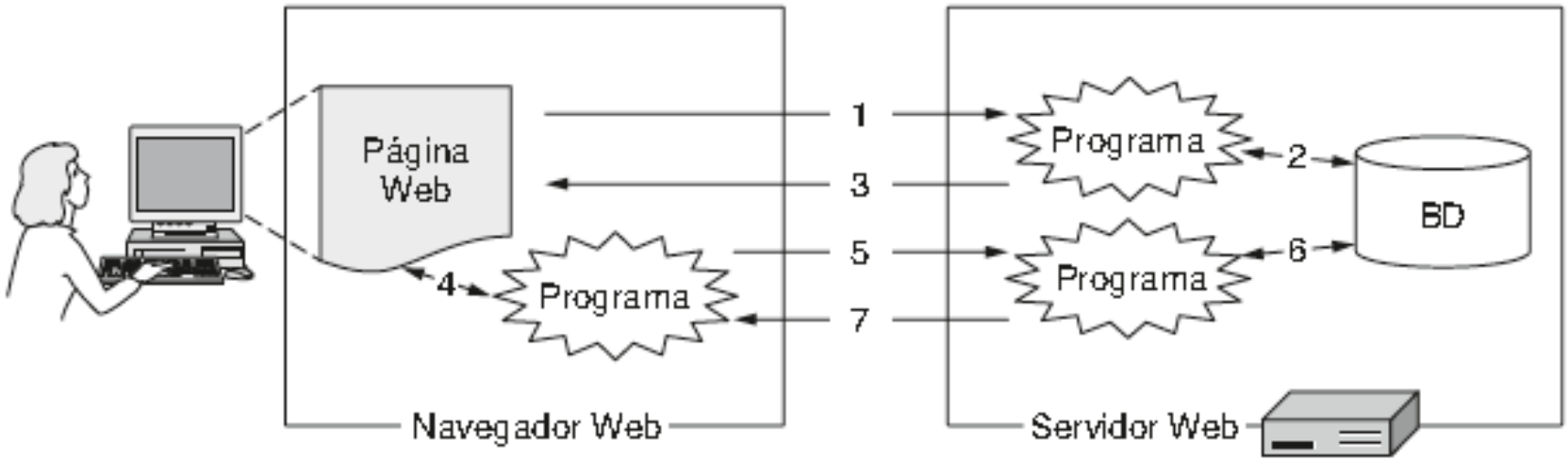


Páginas Web Dinâmicas

Páginas dinâmicas têm o seu conteúdo modificado em tempo de execução

Dependendo dos atributos informados nas requisições, um **programa** é capaz de adaptar o conteúdo ou buscar conteúdos específicos para atender às requisições

Os programas capazes de adaptar o conteúdo dinâmico podem executar no lado do cliente ou servidor



Páginas Web Dinâmicas – Exemplo PHP

```
<html>
<body>
<form action = "action.php" method = "post">
<p> Please enter your name: <input type = "text" name = "name"> </p>
<p> Please enter your age: <input type = "text" name = "age"> </p> <input type = "submit">
</form>
</body>
</html>
```

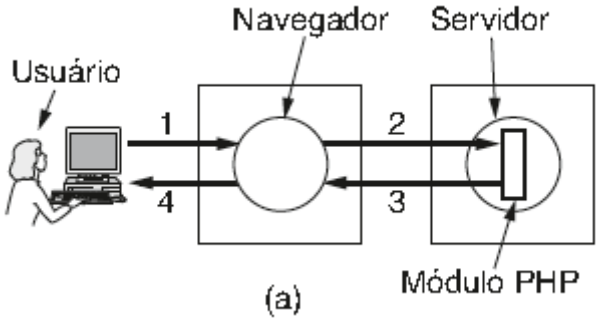
(a)

```
<html>
<body>
<h1> Reply: </h1>
Hello <?php echo $name; ?>.
Prediction: next year you will be <?php echo $age + 1; ?>
</body>
</html>
```

(b)

```
<html>
<body>
<h1> Reply: </h1>
Hello Barbara.
Prediction: next year you will be 33
</body>
</html>
```

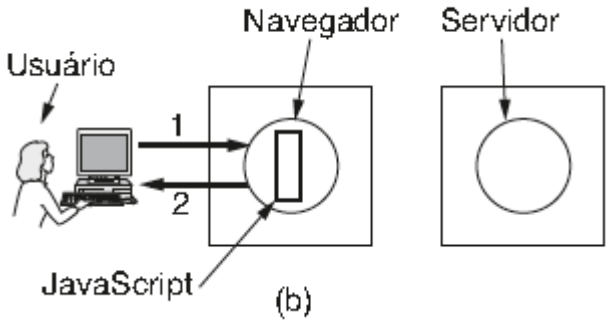
(c)



(a)

(a) Uma página Web contendo um formulário. (b) Um script PHP para o controle de saída de formulário. (c) Saída do script PHP para as entradas ‘‘Barbara’’ e ‘‘32’’

Páginas Web Dinâmicas – Exemplo Javascript



Uso do JavaScript no processamento de um formulário

```
<html>
<head>
<script language = "javascript" type = "text/javascript">
function response(test#form) {
      var person  test#form.name.value;
      var years  eval(test#form.age.value) + 1;
      document.open();
      document.writeln("<html> <body>");
      document.writeln("Hello " + person + ".br");
      document.writeln("Prediction: next year you will be " + years + ".");
      document.writeln("</body> </html>");
      document.close();
}
</script>
</head>

<body>
<form>
Please enter your name: <input type = "text" name = "name">
<p>
Please enter your age: <input type = "text" name = "age">
<p>
<input type = "button" value = "submit" onclick = "response(this.form)">
</form>
</body>
/html
```

HTTP – *Hypertext Transfer Protocol*

Protocolo de comunicação amplamente utilizado para transferência de dados na *World Wide Web*

Implementado na camada de aplicação

- Utiliza TCP/IP
- Modelo cliente/servidor

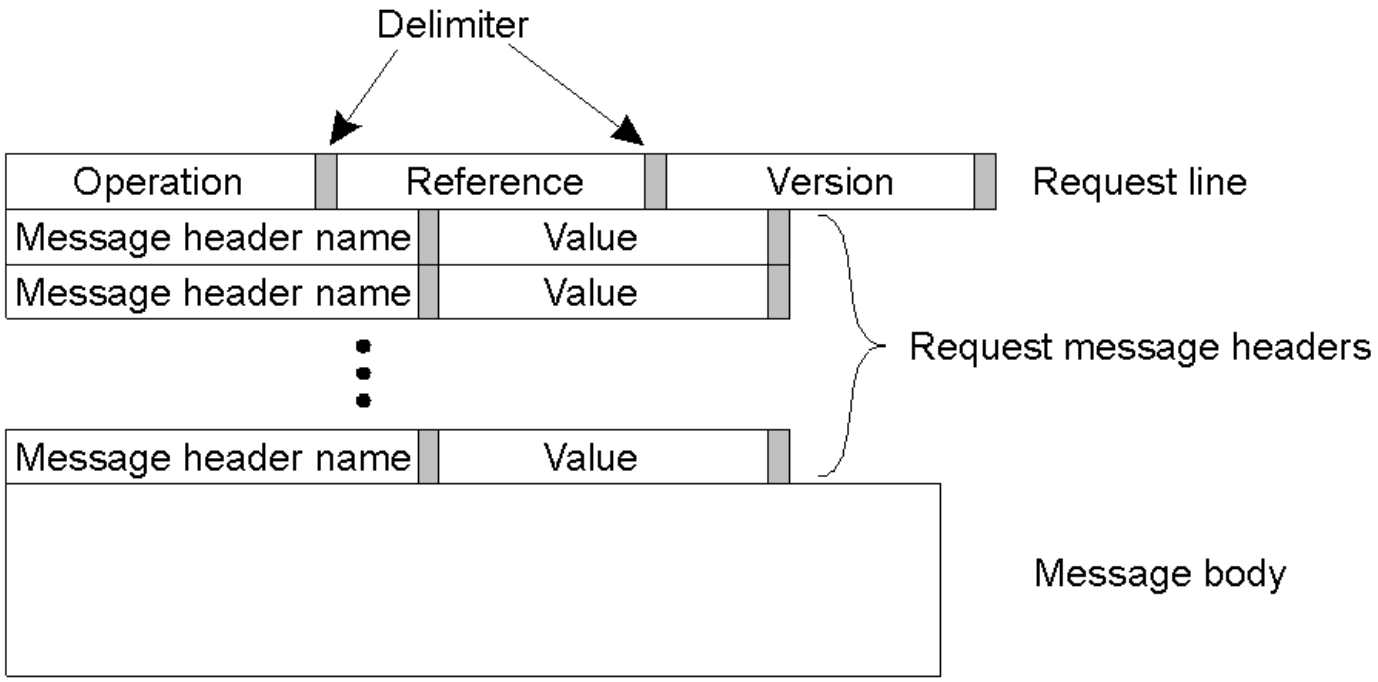
Principais operações suportadas pelo protocolo HTTP

Operação	Descrição
Head	Requisição para retornar o cabeçalho de um documento
Get	Requisição para retornar um documento para o cliente
Put	Requisição para armazenar um documento
Post	Fornece dados para serem adicionados a um documento (coleção)
Delete	Requisição para remover um documento

Protocolo HTTP – Requisições

Requisição:

```
GET / HTTP/1.1
Host: ufsc.br
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0)
Gecko/20100101 Firefox/86.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*
Accept-Language: en-US,en;
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: PHPSESSID=9i0n080oqu31eog9n9lph96o41
Upgrade-Insecure-Requests: 1
```

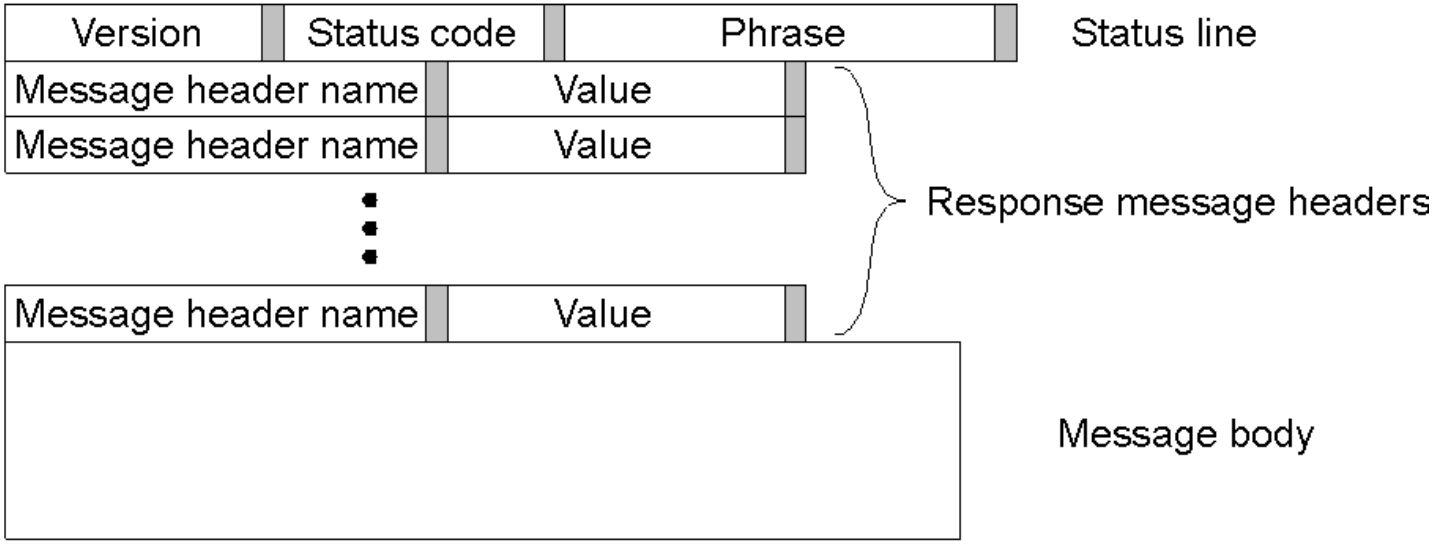


Protocolo HTTP – Respostas

Resposta:

```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 09 Mar 2021 01:18:17 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Keep-Alive: timeout=2
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Encoding: gzip

<!DOCTYPE html>
<head> ...
```



Protocolo HTTP – Atributos

Cabeçalho	Tipo	Conteúdo
User-Agent	Solicitação	Informações sobre o navegador e sua plataforma
Accept	Solicitação	O tipo de páginas que o cliente pode manipular
Accept-Charset	Solicitação	Os conjuntos de caracteres aceitáveis para o cliente
Accept-Encoding	Solicitação	As codificações de páginas que o cliente pode manipular
Accept-Language	Solicitação	Os idiomas com os quais o cliente pode lidar
If-Modified-Since	Solicitação	Data e hora para verificar atualização
If-None-Match	Solicitação	Tags enviadas anteriormente para verificar atualização
Host	Solicitação	O nome DNS do servidor
Authorization	Solicitação	Uma lista das credenciais do cliente
Referer	Solicitação	O URL anterior do qual a solicitação veio
Cookie	Solicitação	Cookie previamente definido, enviado de volta ao servidor
Set-Cookie	Resposta	Cookie para o cliente armazenar
Server	Resposta	Informações sobre o servidor
Content-Encoding	Resposta	Como o conteúdo está codificado (por exemplo, *gzip*)
Content-Language	Resposta	O idioma usado na página
Content-Length	Resposta	O tamanho da página em bytes
Content-Type	Resposta	O tipo MIME da página
Content-Range	Resposta	Identifica uma parte do conteúdo da página

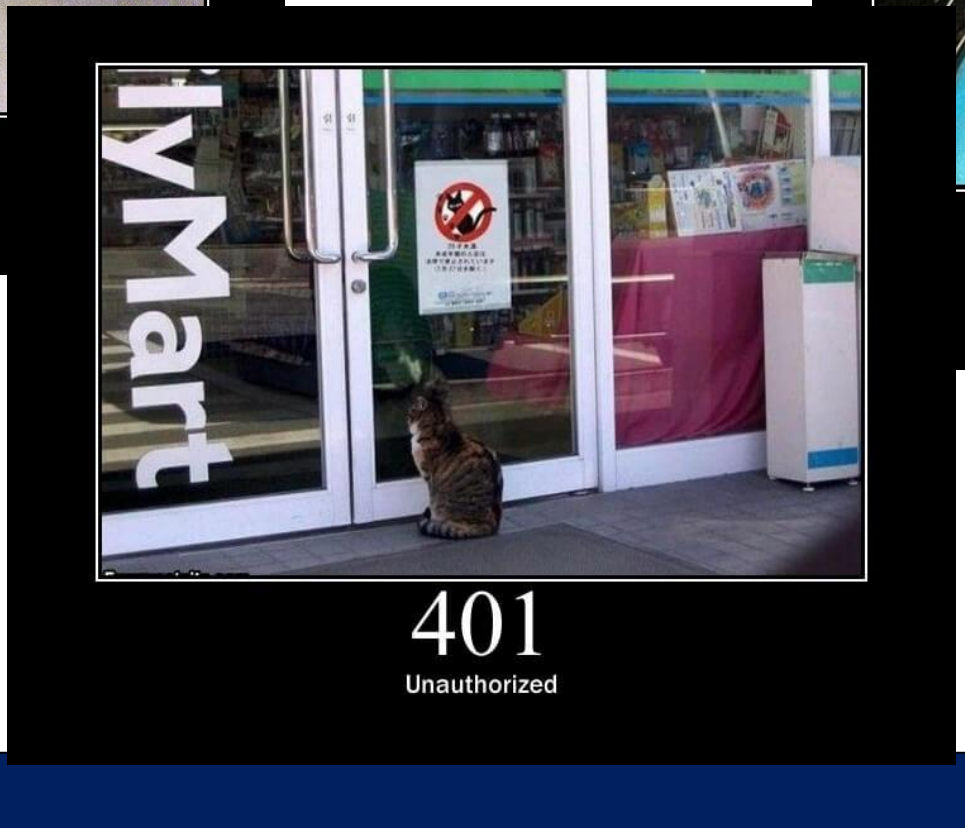
Protocolo HTTP – Status de respostas

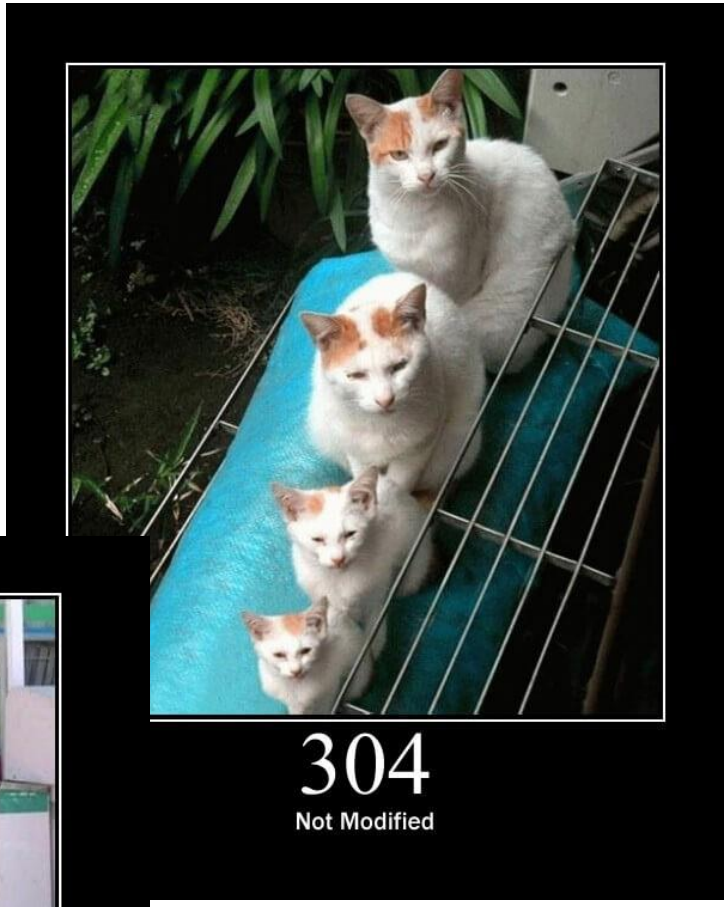
Grupos de respostas para uma requisição HTTP: código de *status*

Código	Significado	Exemplos
1xx	Informação	100 = servidor concorda em tratar da solicitação do cliente
2xx	Sucesso	200 = solicitação com sucesso; 204 = nenhum conteúdo presente
3xx	Redirecionamento	301 = página movida; 304 = página em cache ainda válida
4xx	Erro do cliente	403 = página proibida; 404 = página não localizada
5xx	Erro do servidor	500 = erro interno do servidor; 503 = tente novamente mais tarde

Protocolo HTTP – Status de respostas

Algumas ilustrações para os códigos de status HTTP: https://http.cat/





Exemplos de Servidores Web

Para hospedar páginas ou recursos Web, é necessário implantar o serviço em um servidor Web:

- Apache
- Nginx (pronuncia “*enginhex*”)
- Glassfish
- IIIS (*Internet Information Server*)

Servidor Apache

- (Re)inicializando o serviço:
  /etc/init.d/apache2 restart

- Parando o serviço:
  /etc/init.d/apache2 stop

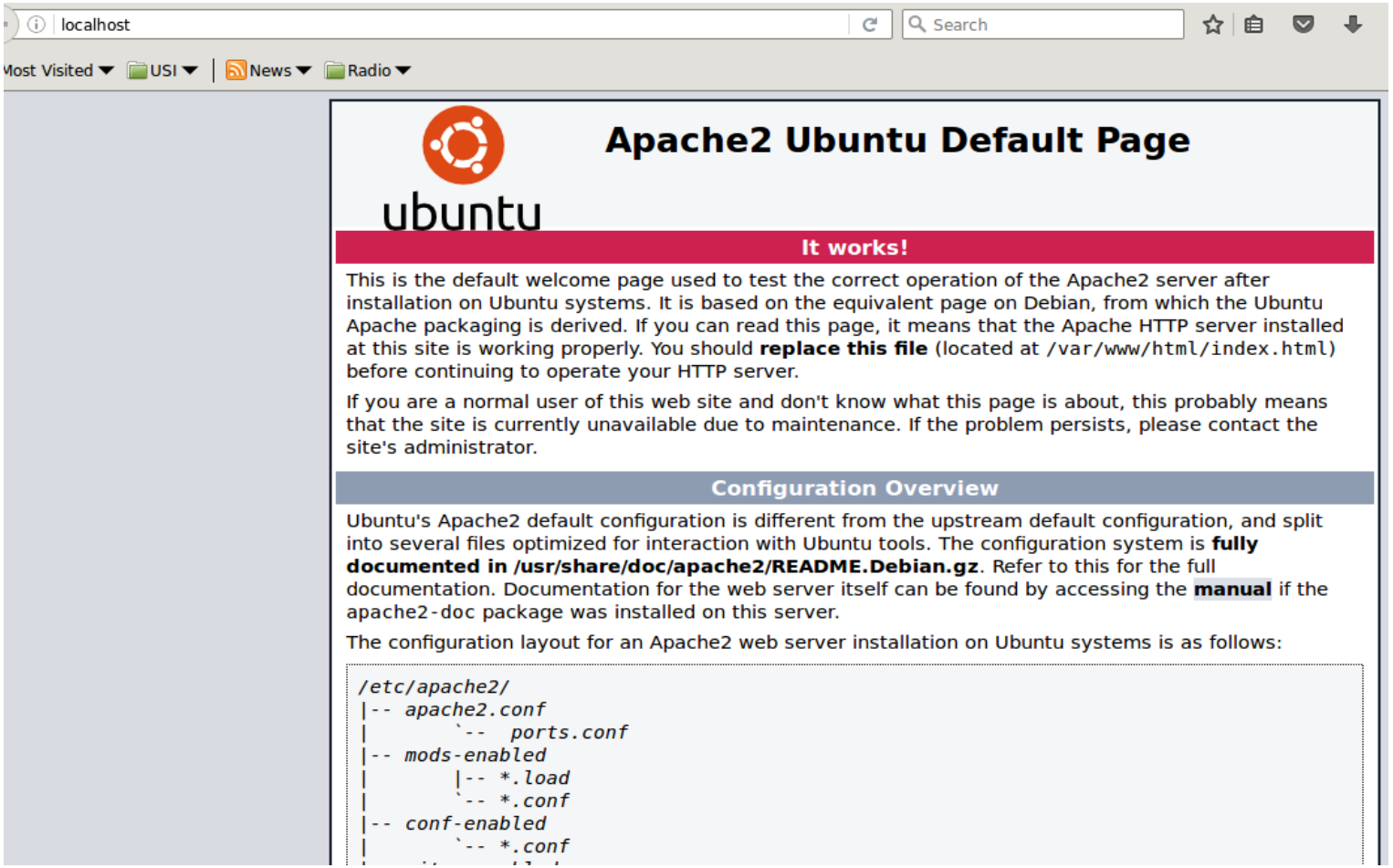
- Local onde recursos serão processados pelo serviço:
  /var/www/

Simples tutorial: https://www.youtube.com/watch?v=5XyPJiU_ZJs

Instalação do Nginx:
https://www.youtube.com/watch?v=FBl1gZzcESY

Servidor Apache

Para testar se serviço está executando. No navegador, digite: localhost



O conteúdo da página carregada está em: /var/www/html/index.html

Exercício

Procure observar algumas requisições e respostas HTTP em sites que costuma usar.

Para isso, utilize alguma ferramenta de depuração Web, por exemplo, fiddler, Yslow!, ...

Identifique requisições GET e POST

1) Que tipos de documentos comumente são requisitados?

2) Faça sucessivos acessos ao mesmo site e observe se todas as requisições são feitas novamente. Há alguma diferença no desempenho e na quantidade de documentos requisitados? Explique