Desenvolvimento de Software para Internet

Aula 01

Contato

Profª. Aparecida F. C. Rosa

Profª. Cida

E-mail: aparecida.rosa@docente.unip.br

Compartilhamento do material

- Link para baixar o material de apoio das aulas de DSWI

http://tiny.cc/7d1nsz

- O que é uma rede e como são computadores conectados?
- O que é a Internet?
- Como funcionam os navegadores (*browsers*)?
- O que acontece "nos bastidores" quando eu conecto a um servidor Web?
- O que é o ***Hypertext Markup Language*** (HTML)?
- Como funcionam os Web sites?

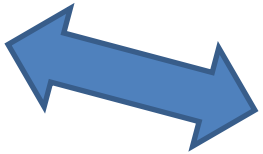
Objetivos

- Entender o ambiente em que uma aplicação Web é executado, que inclui:
  - a Internet,
  - *Web servers* (servidores Web),
  - *browsers* (navegadores),
  - comunicações de rede,
  - identificadores de recursos e
  - linguagem de marcação (HTML).

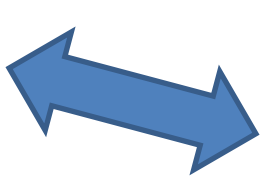
- Uma rede de computadores é qualquer sistema no qual dois ou mais dispositivos de computação estão ligados entre si.
- Redes consistem de computadores interconectados que podem trocar informações entre si.
- Duas partes de qualquer rede que permite aos computadores trocar informações:
  - Primeiro, eles precisam estar conectados por um *link* - esta é geralmente uma ligação física, como um cabo ou uma linha telefônica, mas também pode ser uma ligação sem fios.
  - Segundo, eles precisam ter uma linguagem comum em que cada um deles pode usar para decodificar os sinais que cada um deles envia ao longo de cada *link* para o outro.

Protocolo de comunicação

- Na rede, o termo de uma linguagem comum é um **protocolo.**
- ***Transmission Control Protocol/Internet Protocol***, ou **TCP/IP -** protocolo utilizado na Internet.
- Esse protocolo permite que os computadores troquem 'pacotes' de mensagens que contêm dados ou informações de controle, como confirmação de recebimento.
- Ao trocar dados e informações de controle, os dois computadores podem ter certeza de que as informações foram transmitidas com êxito ou um deles pode reenviá-las se ocorrer um problema.



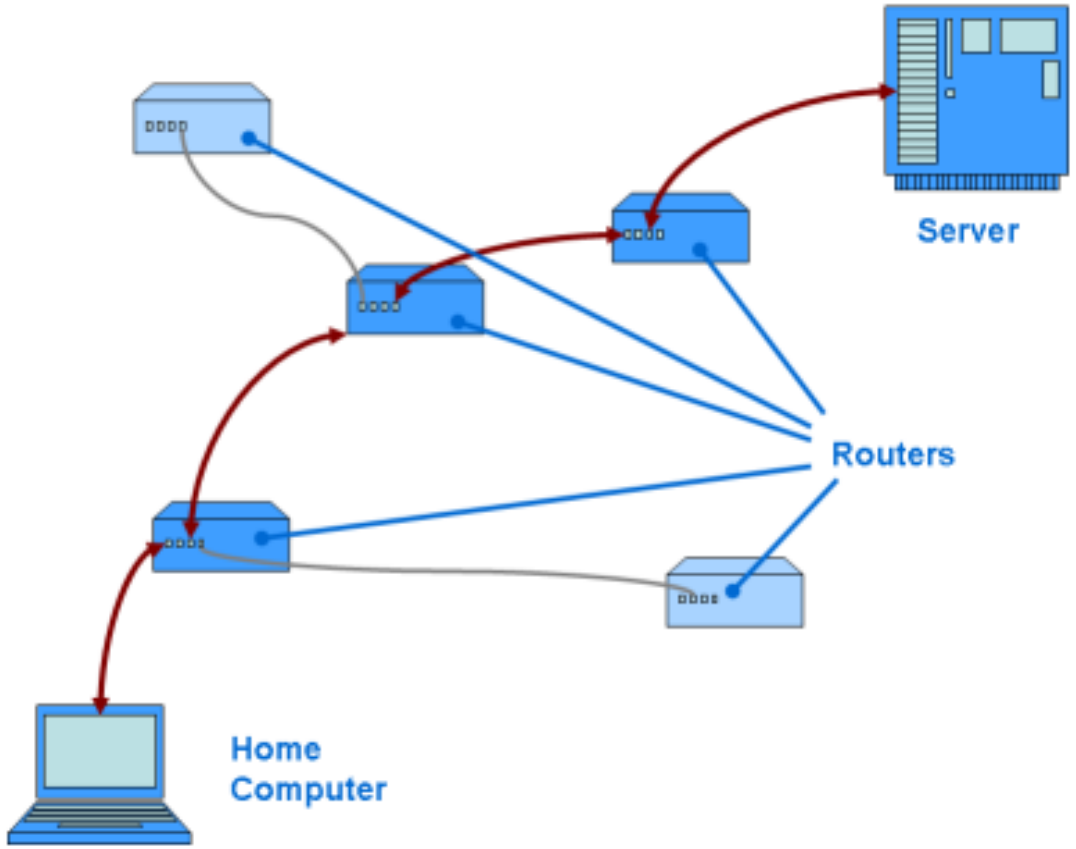
TCP/IP



- Entre quaisquer dois computadores conectados via Internet, geralmente há vários dispositivos intermediários que fazem parte da ligação entre os dois computadores.
- O mais comum destes dispositivos são **roteadores,** e eles agem um pouco como uma central telefônica.
- Os roteadores são dispositivos de computação especializados que direcionam o tráfego de rede ao longo da rota correta em direção ao seu destino.

Rota de rede

O diagrama mostra uma rota de rede que transporta informações entre um computador e um servidor Web através de roteadores



Rota de rede

- Por outro lado, neste ponto, você deve observar que ninguém realmente possui a Internet como um todo.
- Para um usuário doméstico, os primeiros roteadores ao longo de uma rota provavelmente pertencem a um provedor de serviços de Internet (*Internet Service Provider* - ISP) orientado ao consumidor.
- No outro extremo da rota, um provedor de hospedagem na Web ou uma empresa geralmente possui o próprio servidor da Web e os roteadores mais próximos do servidor..

Rota de rede

- Entre o ISP e o provedor de hospedagem na Web, os roteadores que os vinculam formam uma rede de *backbone*.
- A Internet consiste em muitas redes de *backbone* ligadas entre si, cada uma das quais geralmente é de propriedade comercial, educacional ou do governo.

Endereços de Rede

- Para que os roteadores possam determinar a rota correta entre dois computadores, cada computador precisa ter um **endereço** para que os roteadores saibam para onde enviar os dados.
- Em termos de rede, um endereço é um identificador exclusivo para um determinado dispositivo.
- Até os próprios roteadores têm endereços - na verdade, para qualquer endereço de destino (como um servidor da Web), os roteadores precisam apenas saber o endereço do próximo roteador mais perto do destino e simplesmente entregam a responsabilidade pela entrega dos dados a esse roteador.
- Eventualmente, o último roteador da cadeia conhece o próprio servidor da Web e pode fornecer os dados para ele.

Endereços de Rede

- Na Internet, um endereço **IP** (***Internet Protocol***) é um número de quatro bytes, geralmente escrito na notação "decimal pontilhada"; por exemplo **10.43.172.77**.
- Todo dispositivo no mundo que se conecta diretamente à Internet possui um endereço IP exclusivo de quatro bytes.
- Isso significa que existem mais de quatro bilhões de endereços IP possíveis (embora cerca de 20 milhões de endereços sejam reservados para usos especiais).
- Pode parecer um número grande, mas há um número crescente de computadores conectados à Internet.

Domain Name System (DNS)

- Mesmo usando notação decimal pontilhada, os endereços IP não são muito memoráveis.
- As pessoas tendem a achar mais fácil lembrar nomes do que números.
- Por esse motivo, a Internet possui um sistema que possibilita que um site tenha um nome 'amigável' e um endereço IP menos memorável.
- Este é o ***Domain Name System*** **(DNS -** Sistema de Nomes de Domínio**),** que funciona convertendo nomes de domínio em endereços IP.

Domain Name System (DNS),

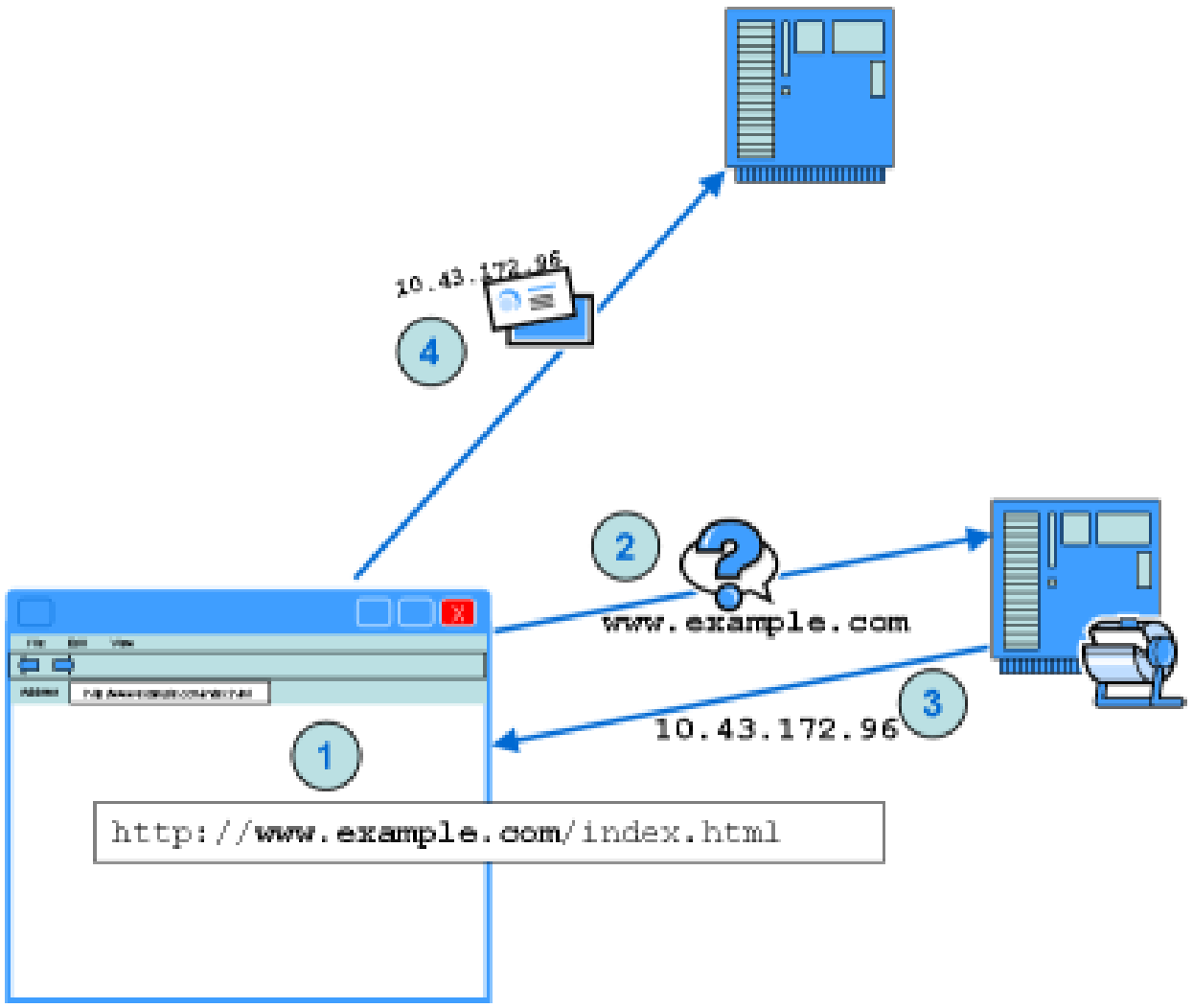
- Existem muitos servidores DNS conectados à Internet, e seu trabalho é agir como uma lista telefônica da Internet - eles procuram o endereço IP de um determinado nome de domínio em uma grande lista de todos os nomes de domínio.
- Por exemplo, quando você digita um endereço da Web na barra de endereços do navegador, ocorrem as seguintes etapas, conforme mostrado no diagrama:

❶ O navegador determina o nome do servidor de destino (***host name***) a partir do endereço Web.

❷ Seu computador usa DNS para obter o endereço IP do computador de destino. Ele entra em contato com o servidor DNS e passa o nome do host.

❸ O servidor DNS procura na lista de nomes de domínio por um registro correspondente e envia o endereço IP de volta ao seu computador.

❹ Seu computador usa o endereço IP para estabelecer comunicação com o servidor da Web.



- Depois que a conexão é estabelecida, o próximo estágio é o fluxo de dados entre os dois computadores.
- Na maioria das situações, o computador que inicia a troca é chamado de **cliente (*cliente*),** e o computador que recebe a conexão é chamado de **servidor (*server*).**
- Um programa de computador é executado no servidor o tempo todo, **escutando (*listening*)** as conexões dos clientes.
- No computador cliente, outro programa (como um navegador da web) se conecta ao servidor sempre que requer informações.
- Por exemplo, quando você solicita uma página da Web, o navegador faz uma conexão com o servidor da Web para essa página quando você clica no botão Ir.

- Em um cenário típico de **cliente / servidor**, o cliente envia alguns dados chamados de solicitação ao servidor, e o servidor determina a natureza da solicitação e formula uma resposta, que é enviada de volta ao cliente.
- Por exemplo, quando um programa de e-mail lê seu e-mail de um servidor de e-mail, ocorrem as seguintes etapas:

Navegadores e servidores Web - Browsers e Web Servers 20

❶ O cliente envia um nome de usuário e senha para o servidor.

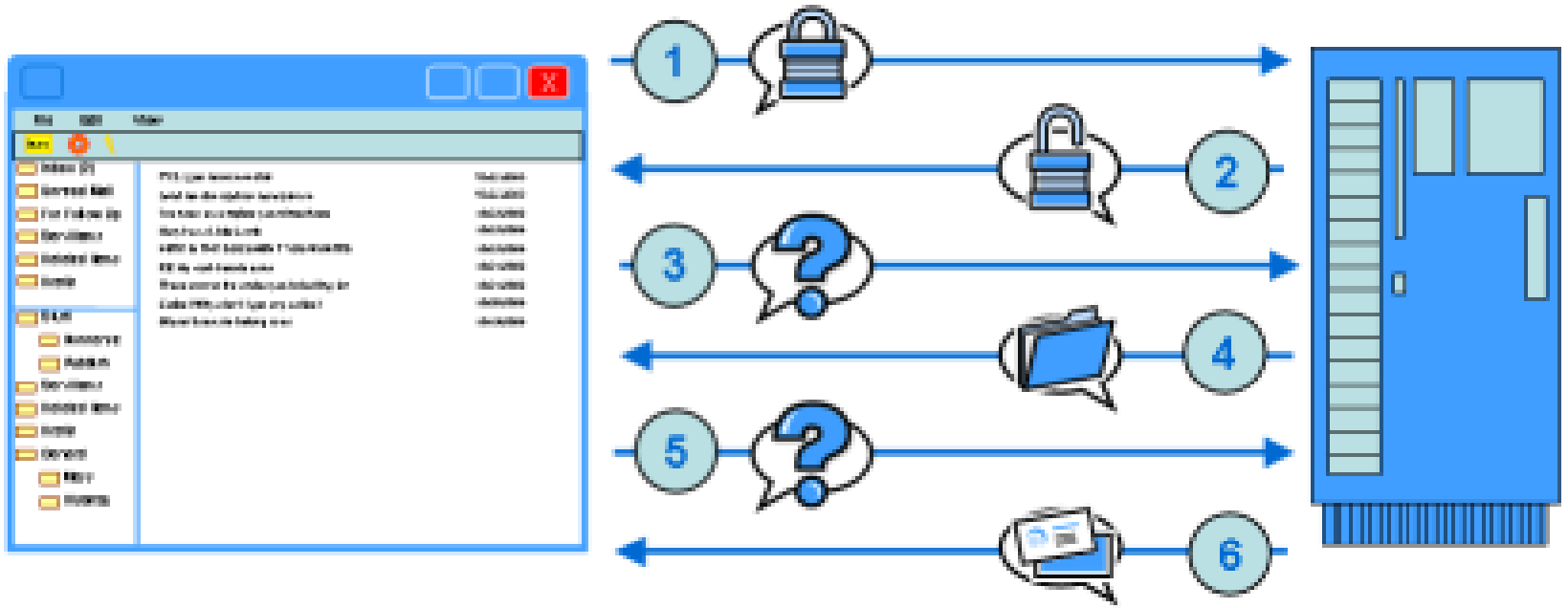
❷ O servidor responde dizendo que o nome de usuário e a senha são aceitos.

❸ O cliente solicita uma lista de e-mails que estão no servidor.

❹ O servidor responde com uma lista de e-mails, sem incluir o corpo do e-mail.

❺ O cliente solicita o corpo de um e-mail específico (por exemplo, quando você clica duas vezes no e-mail para visualizá-lo).

❻ O servidor envia o corpo do e-mail.



- O cliente e o servidor devem ter um entendimento comum do conteúdo das mensagens de solicitação e resposta.
- Vimos anteriormente como, quando os dois computadores fizeram uma conexão de rede, eles têm um entendimento comum, definido pelo protocolo **TCP/IP**, que lhes permite enviar dados e confirmar que eles foram entregues.
- No topo desse protocolo de rede, está outro protocolo de mensagens que especifica uma linguagem comum para aplicações de computador se comunicarem.
- O protocolo entre as duas aplicações de computador é chamado de **protocolo de aplicação (*application protocol*)**.

- Você pode imaginar um protocolo de rede como os regulamentos que regem como construir uma estrada e um protocolo de aplicação como os regulamentos que governam como dirigir na estrada.
- O protocolo de aplicação da Internet mais comum é o **HTTP** (***Hypertext Transfer Protocol***), que é o principal protocolo usado na ***World Wide Web***.
- Você também pode encontrar o **FTP** (***File Transfer Protocol***), que pode ser usado para transferir arquivos de um computador para outro.

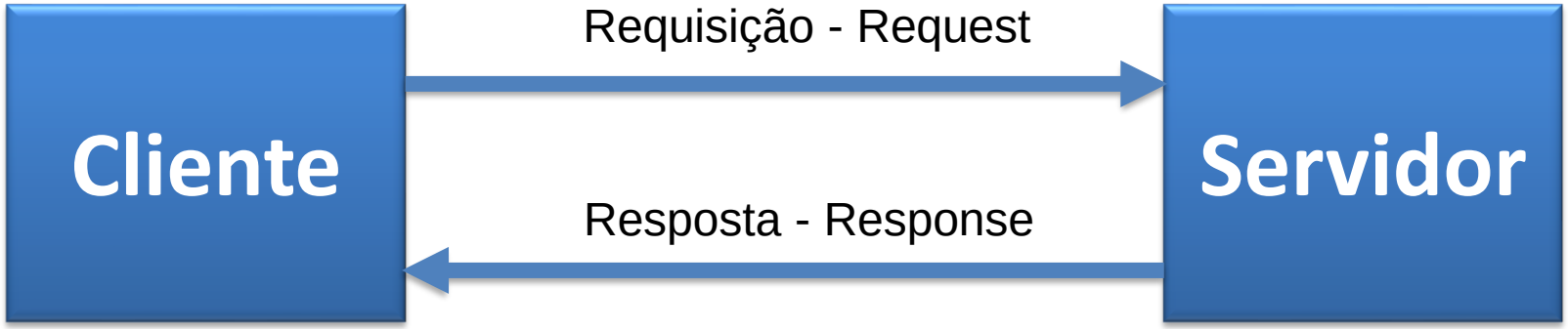
Hypertext Transfer Protocol (HTTP)

- Quando um navegador se comunica com um servidor Web, ele envia uma solicitação **HTTP (*request*)** ao servidor Web.
- O conteúdo da solicitação **HTTP** especifica qual ação o cliente exige que o servidor execute.
- O **HTTP** suporta vários métodos (às vezes chamados de '**verbos**') que o cliente pode especificar na solicitação; os métodos mais usados são **GET** e **POST**.
- Um navegador usa o método **GET** quando **solicita uma página da Web** e o método **POST** quando ele **envia os dados** a serem processados, por exemplo, quando você clica no botão Enviar ou Pesquisar em uma página da Web.

- A seguir, é apresentado um exemplo das informações enviadas entre o navegador e o servidor Web quando o navegador envia uma solicitação **GET** para **http://www.example.com/index.html**.

- Não se preocupe com os detalhes da solicitação e resposta; isso é simplesmente para dar uma amostra do que está acontecendo dentro do protocolo **HTTP**.

- Métodos HTTP = Verbos HTTP



Composição da requisição

[MÉTODO] [URI] HTTP/[Versão]
[Cabeçalhos]

[CORPO/PAYLOAD]

```
POST /clientes HTTP/1.1
Content-type: application/json
Accept: application/json

{
   “nome” : “João”
}
```

Métodos para fazer requisição HTTP

- Método indica a ação que desejamos que seja executada

  GET, POST, PUT, DELETE, etc.

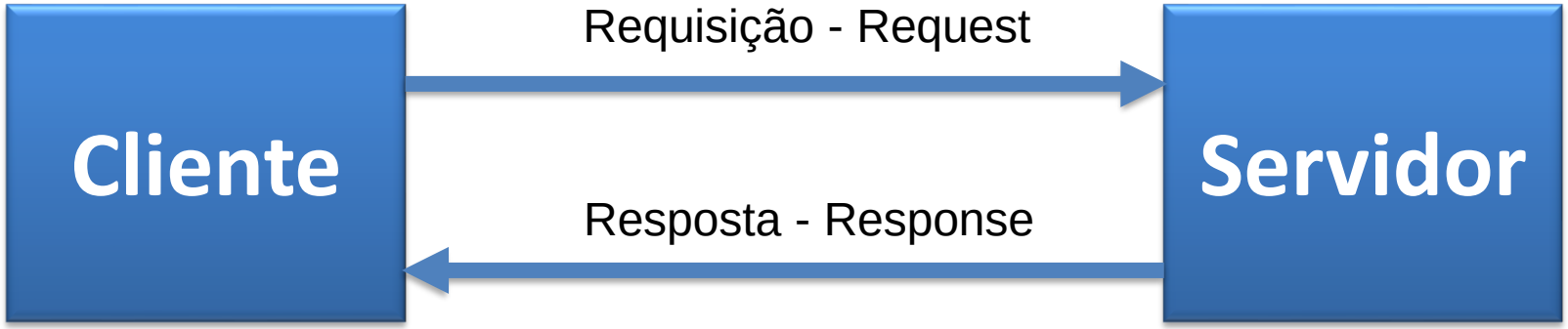
- **GET**: solicita que seja devolvido na resposta os dados
- **POST**: Envia dados para o servidor: adicionar dados no DB.

- **URI**: Caminho que identifica o que queremos dentro do servidor.

Métodos para fazer requisição HTTP

- **Cabeçalhos**: informações sobre a requisição. Definem nomes de chaves de valores que podem ser usados pelo servidor para interpretar a requisição e executar a operação.
- **Content-type**: servidor interpretar a mensagem no corpo corretamente.
- **Accept**: define quais os tipos de conteúdo aceitos como resposta – formato do retorno da resposta
- **Corpo/payload** : não é obrigatório. Depende do método HTTP. No corpo da requisição que enviamos os dados do cliente para o servidor.

- Métodos HTTP = Verbos HTTP



Composição da resposta

HTTP/[Versão] [STATUS]
[Cabeçalhos]

[CORPO]

HTTP/1.1 201 Created
Content-type: application/json

{
"id" : 571,
"nome" : "João"
}

Request - Response

Request from Client Browser

```
GET /index.html HTTP/1.1
Host: www.example.com
Accept-Language: en
```

Response from Web Server

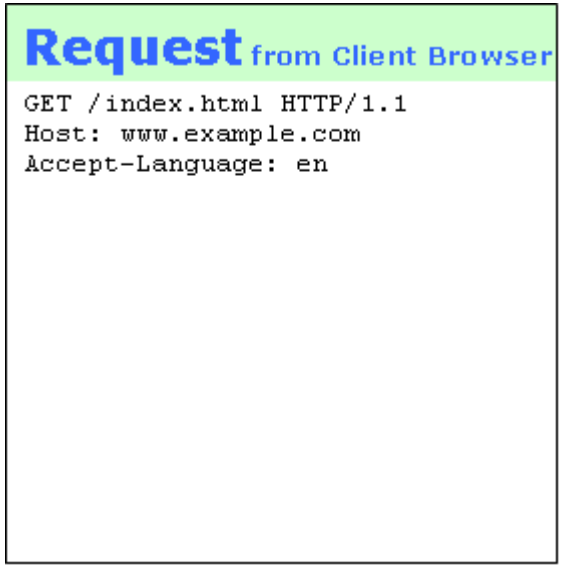
```
HTTP/1.1 200 OK
Date: Mon, 23 Oct 2006 12:05:20
GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Content-Type: text/html;
charset=utf-8
Content-Length: 194

<html>
  <head>
    <title>Welcome</title>
  </head>
  <body>
    <h1>Welcome to
Example.com</h1>
```

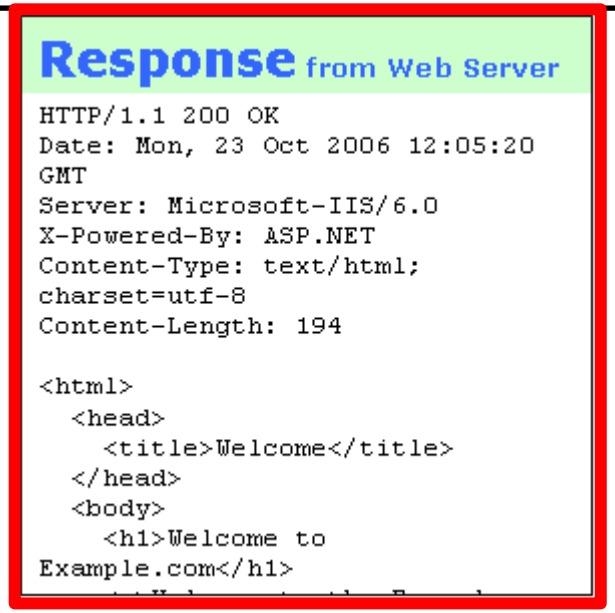
Request:

A primeira linha da solicitação especifica o método **(GET)**,
a página ou recurso **(/index.html)**
e o protocolo **(HTTP 1.1).**

As linhas seguintes são cabeçalhos de solicitação que indicam informações extras para o servidor Web

Request - Response





Response:

A resposta contém uma linha de status que inclui o protocolo **(HTTP 1.1),**
um código de **status (200)** e,
uma descrição de texto do que significa o código de status **(OK).**

As próximas linhas são cabeçalhos de resposta que contêm informações adicionais sobre o servidor e a página Web.

Após os cabeçalhos, há uma linha em branco, seguida pelo conteúdo da página Web. Essa é a carga real **(*payload*)** da resposta e contém as informações que o navegador realmente exibirá.

- O objetivo do HTTP é simplesmente fornecer uma 'linguagem' comum na qual o navegador e o servidor Web possam trocar informações sobre páginas da Web e outros recursos.
- A substância real da troca, quando o navegador solicita uma página da Web, é a seção da resposta que descreve como a página aparecerá dentro da janela do navegador.
- A descrição da página da Web que o servidor envia está em um formato padrão, para que o navegador possa entender como o servidor Web deseja que ele exiba a página.

- Desta vez, o formato padrão não é de fato um protocolo, porque não define como dois computadores trocam informações.
- Na verdade, é uma linguagem, embora a distinção seja boa:

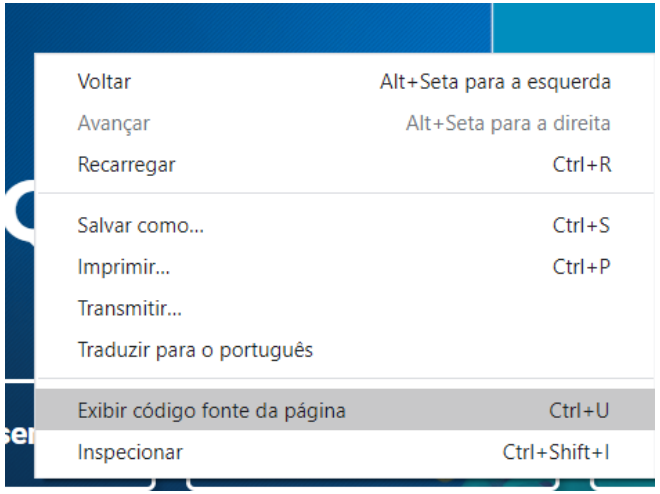
  -existem regras sobre linguagens de computador, assim como existem protocolos de rede.
- A linguagem que os servidores Web e navegadores usam para descrever uma página da Web é **HTML (*Hypertext Markup Language*).**
- **HTML** é uma linguagem usada principalmente para marcação de textos em uma página.
- Ele não contém nenhum suporte avançado para operações complexas; serve apenas para organizar o conteúdo de maneira legível na página da web.

- Quando o navegador recebe uma página HTML, ele converte a descrição HTML em uma tela por um processo chamado **renderização (*rendering*)**.
- O navegador lê as instruções em HTML e **“renderiza“ (“*renders*”)** o resultado na tela.
- HTML é uma linguagem baseada em texto, o que significa que você pode visualizar e editar HTML em um editor de texto padrão como o Bloco de Notas.
- Ele consiste do texto na página da Web, juntamente com as **"tags de marcação"** que indicam ao navegador como esse texto deve ser exibido.

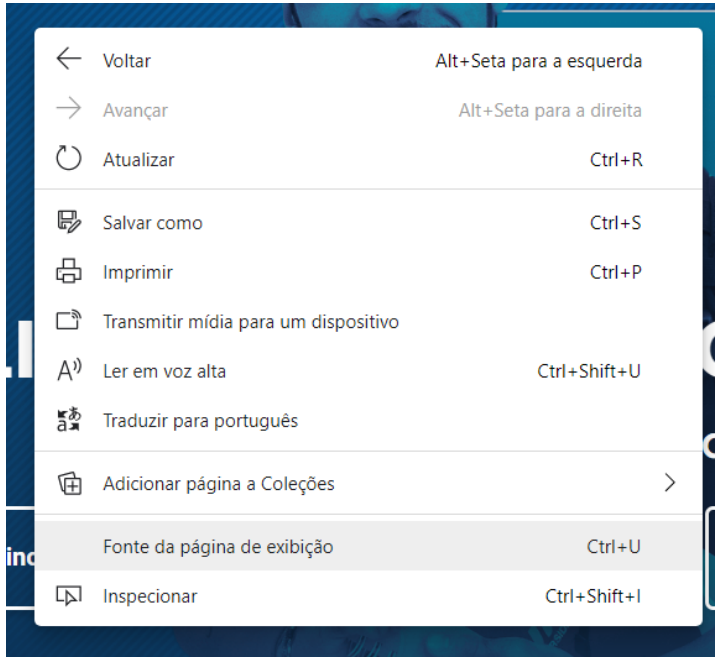
- A marcação especifica itens como o tipo de fonte (letra) a ser usada para seções de texto, onde pode exibir imagens incorporadas e, é claro, hiperlinks que permitem vincular a diferentes páginas da Web.

- Você pode olhar para o HTML de qualquer página da Web no seu navegador, visualizando a fonte da página.

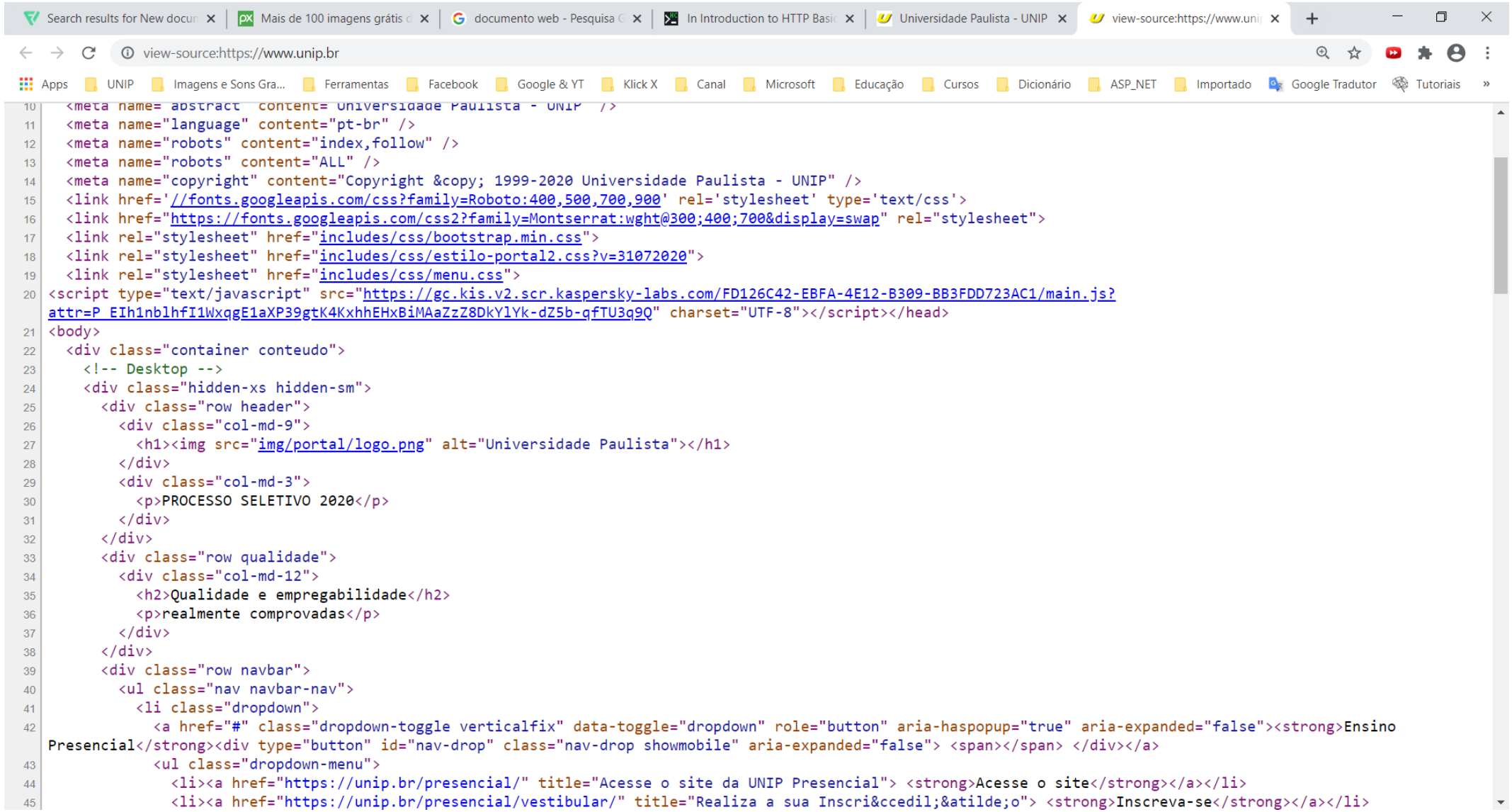
- (Clicar com o botão direito na página)



Google Chrome



Microsoft Edge




```
  <meta name="abstract" content="Universidade Paulista - UNIP" />
  <meta name="language" content="pt-br" />
  <meta name="robots" content="index,follow" />
  <meta name="robots" content="ALL" />
  <meta name="copyright" content="Copyright &copy; 1999-2020 Universidade Paulista - UNIP" />
  <link href='//fonts.googleapis.com/css?family=Roboto:400,500,700,900' rel='stylesheet' type='text/css'>
  <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@300;400;700&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="includes/css/bootstrap.min.css">
  <link rel="stylesheet" href="includes/css/estilo-portal2.css?v=31072020">
  <link rel="stylesheet" href="includes/css/menu.css">
<script type="text/javascript" src="https://gc.kis.v2.scr.kaspersky-labs.com/FD126C42-EBFA-4E12-B309-BB3FDD723AC1/main.js?
attr=P_EIh1nblhfI1WxqgE1aXP39gtK4KxhhEHxBiMAaZzZ8DkYlYk-dZ5b-qfTU3q9Q" charset="UTF-8"></script></head>
<body>
  <div class="container conteudo">
    <!-- Desktop -->
    <div class="hidden-xs hidden-sm">
      <div class="row header">
        <div class="col-md-9">
          <h1><img src="img/portal/logo.png" alt="Universidade Paulista"></h1>
        </div>
        <div class="col-md-3">
          <p>PROCESSO SELETIVO 2020</p>
        </div>
      </div>
      <div class="row qualidade">
        <div class="col-md-12">
          <h2>Qualidade e empregabilidade</h2>
          <p>realmente comprovadas</p>
        </div>
      </div>
      <div class="row navbar">
        <ul class="nav navbar-nav">
          <li class="dropdown">
            <a href="#" class="dropdown-toggle verticalfix" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false"><strong>Ensino
Presencial</strong><div type="button" id="nav-drop" class="nav-drop showmobile" aria-expanded="false"> <span></span> </div></a>
            <ul class="dropdown-menu">
              <li><a href="https://unip.br/presencial/" title="Acesse o site da UNIP Presencial"> <strong>Acesse o site</strong></a></li>
              <li><a href="https://unip.br/presencial/vestibular/" title="Realiza a sua Inscri&ccedil;&atilde;o"> <strong>Inscreva-se</strong></a></li>
```

Exemplo HTML 4.01

- Exemplo de uma página HTML simples, que demonstra alguns conceitos importantes.

```
  DOCTYPE HTML PUBLIC "- / / W3C / / DTD HTML 4.01 Transitional / / EN">
<html>
  <head>
    <title> Bem-vindo </ title>
  </ head>
  <body>
    <img src="/images/logo.jpg" height="50px" width="200px" border="0" />
    <h1> Welcome to Example.com </ h1>
    <p> Bem-vindo ao site da Example.com. Nós esperamos que você vai encontrar muitos dos exemplos aqui. </ P>
    <p> A lista completa dos exemplos está no <a href="/example-list.html"> nesta página web. </ a> </ p>
  </ body>
</ html>
```

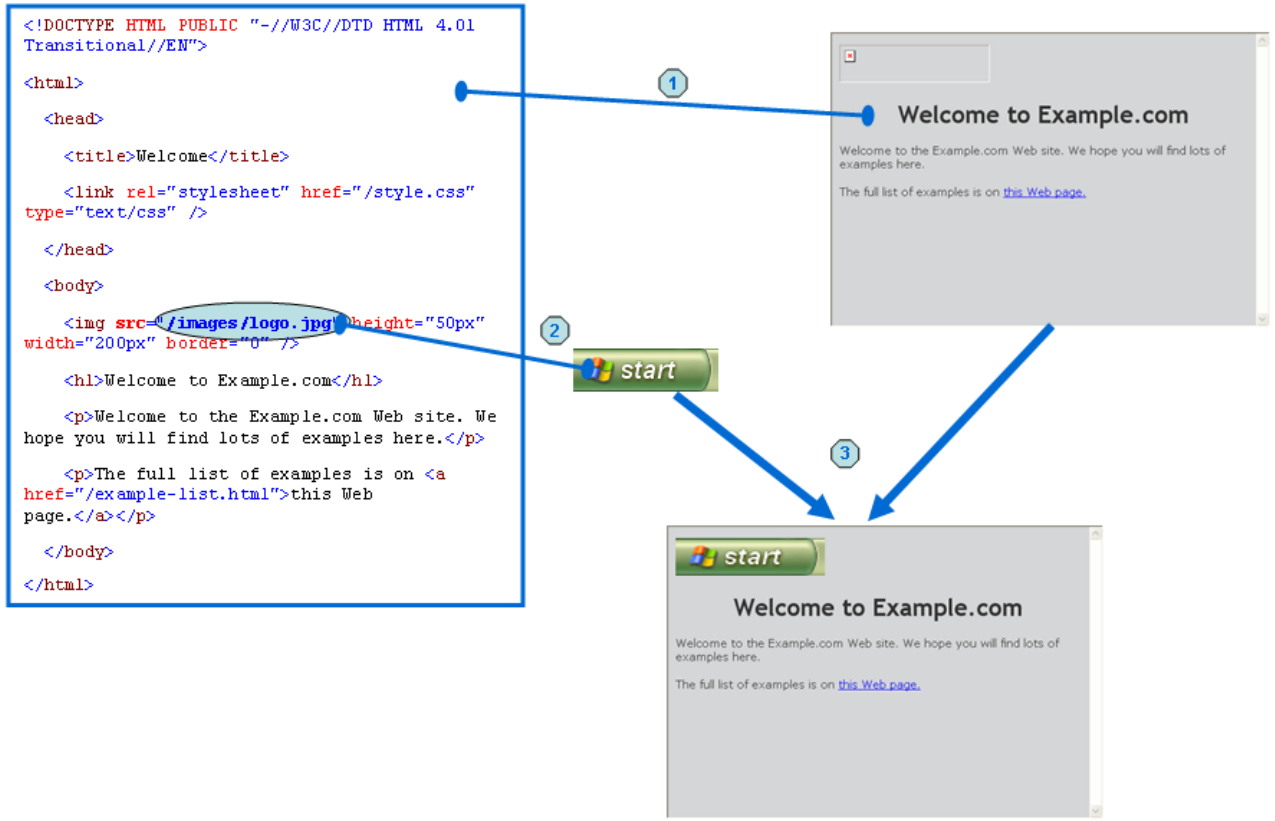
Exemplo HTML5

```
<!DOCTYPE html>
<html>

 <head>
    <title>Welcome</title>
  </head>

<body>
    <img src="/images/logo.jpg" height="50px" width="200px" border="0" />
    <h1>Welcome to Example.com</h1>
    <p>Bem vindo ao site Example.com. Nós esperamos que você encontre muitos exemplos aqui.</p>
    <p>A lista completa dos exemplos está <a href="/example-list.html">nesta página.</a></p>

</body>

</html>
```

- HTML consiste em uma série de **tag** que estão entre o sinal de ”menor que” (**<**) e o sinal de “maior que” (**>**), em torno de uma marcação, como o **tag <html>,** no exemplo.
- No final do documento existe de outra marcação, **</html>.**
- A barra indica que esta é a marcação de fim que corresponde ao tag <html> no início do documento.
- Tudo entre as duas marcas é chamado de elemento <html> .
- Dentro do elemento <html> existe um elemento**<head>,** que contém **<title>**.

- Um documento HTML é composto de elementos que estão aninhados um dentro do outro.
- Alguns dos elementos que definem **atributos** dentro dos < >, bem como um nome de tag: tag **<a>** e a tag **<img>**, por exemplo.
- Atributos contêm informações adicionais sobre essa tag que o navegador utiliza para renderizar a página web.

- A tag **<img>** no exemplo indica que o navegador deve inserir uma imagem na página Web renderizada, e deve conter os seguintes atributos:
- **src**. Refere-se à **Uniform Resource Locator (URL)** da imagem a inserir na página.
- **height**. Isto indica a altura da imagem na página da Web renderizada - neste caso, 50 pixels.
- **width**. Isto indica a largura da imagem na página da Web renderizada.
- **border**. Isto define a largura da borda da imagem - neste caso, o limite é definido como 0 para que o navegador não exiba uma borda.
- Observe que o documento HTML não contém a própria imagem. Em vez disso, o documento contém apenas a URL da imagem. Quando o navegador renderiza este documento, ele executa as seguintes etapas.



❶ O navegador vê a tag **<img>** no documento e reconhece que deve exibir uma imagem incorporada.

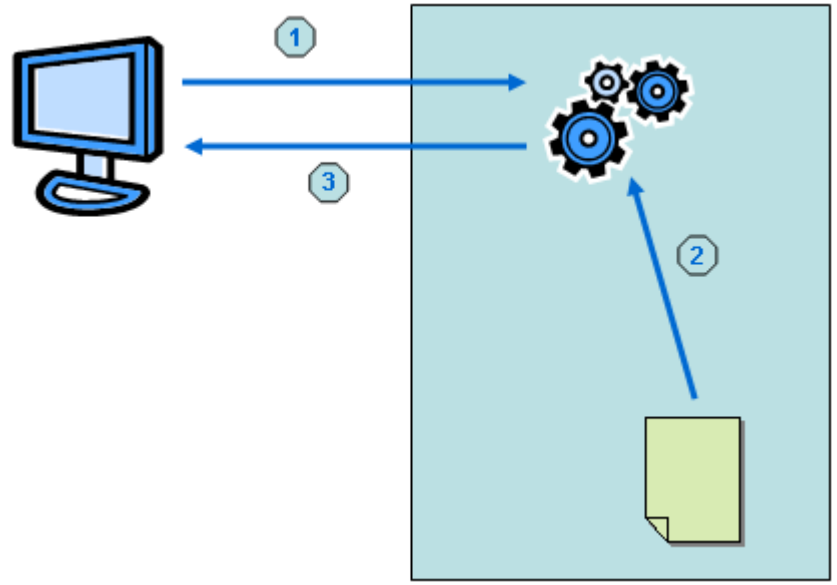
❷ O navegador cria uma solicitação **HTTP GET** para o **URL** especificado para o arquivo de imagem e envia essa solicitação ao servidor Web.

❸ Quando o servidor responde enviando a imagem, o navegador incorpora a imagem na página Web renderizada.

- O cenário apresentado mostra como um navegador baixa e renderiza uma página da Web simples.

- Vamos dar uma olhada na solicitação **HTTP**, para que possamos ver o que o servidor faz quando responde à solicitação:

```
GET /index.html HTTP/1.1
Host: www.example.com
Accept-Language: en
```



❶ O navegador envia a solicitação **GET** para o servidor. Como você viu acima, a primeira linha da solicitação especifica o recurso **(/index.html)** que o navegador está solicitando.

❷ Quando o servidor Web recebe essa solicitação, ele localiza a pasta raiz do site no disco rígido, procura um arquivo chamado **index.html** e lê esse arquivo no disco.

❸ O servidor da Web envia o conteúdo do arquivo de volta ao navegador dentro da **resposta HTTP**.

Portanto, se a pasta raiz do site for **C:\Website**, o servidor retornará o conteúdo de **C:\Website\index.html**.

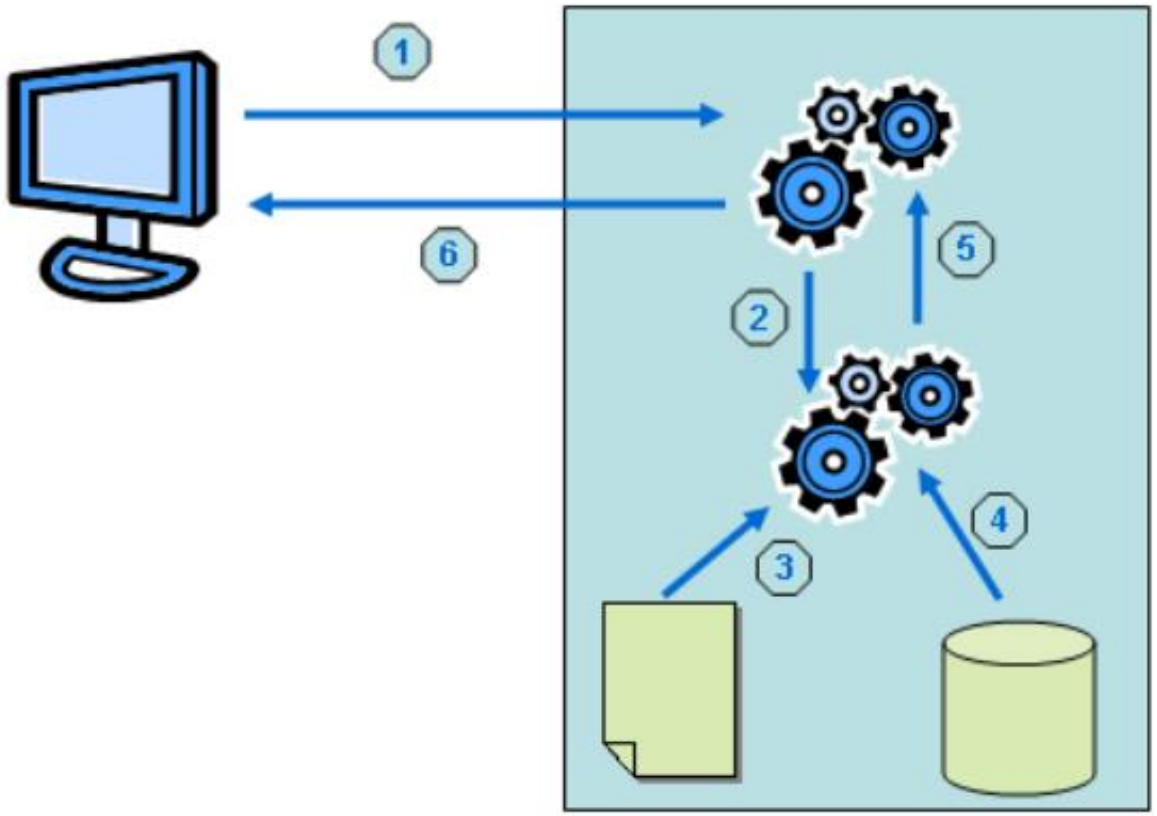
Da mesma forma, quando o navegador solicita **/images/logo.jpg**, o servidor da Web retorna **C:\Website\images\logo.jpg**.

- Esse sistema funciona muito bem se um site da Web não altera seu conteúdo com muita frequência.
- Sempre que os proprietários do site desejam alterar o conteúdo, podem fazer o upload de um substituto para index.html (ou qualquer arquivo que precise ser alterado).
- No entanto, o sistema é menos eficiente se o conteúdo do site for alterado regularmente, por exemplo, se o site exibir dados que sejam alterados regularmente.
- Nesse cenário, não é prático criar constantemente arquivos HTML que contêm os dados.

- A solução para cenários como esse é criar uma aplicação Web que funcione com o servidor Web para criar a página dinamicamente sempre que um navegador solicitar.
- Geralmente, as informações na página são armazenadas em um banco de dados.
- Esse tipo de site é chamado de **site dinâmico**, em contraste com um **site estático**, em que o conteúdo não é gerado toda vez que uma página é solicitada.

- Imagine se seu aplicativo de e-mail da Web funcionasse lendo arquivos do disco rígido.
- Sempre que você excluir um e-mail, o aplicativo de e-mail na Web precisaria de uma nova página HTML para representar sua Caixa de Entrada, para que a Caixa de Entrada não contivesse mais o e-mail excluído.
- O aplicativo de e-mail da Web precisaria, portanto, criar um novo arquivo HTML no disco rígido para representar a nova caixa de entrada.
- A solução para cenários como esse é criar uma aplicação Web que funcione com o servidor Web para criar a página dinamicamente sempre que um navegador solicitar. Geralmente, as informações na página são armazenadas em um banco de dados. Esse tipo de site é chamado de **site dinâmico**, em contraste com um **site estático**, em que o conteúdo não é gerado toda vez que uma página é solicitada.

Quando um navegador solicita que uma página seja executada em um site dinâmico, o ciclo de solicitações funciona da seguinte maneira:



❶ A primeira parte do ciclo funciona da mesma forma que em um site estático - o navegador especifica o nome do recurso dentro de uma solicitação **HTTP GET**.

❷ Depois disso, porém, as coisas funcionam de maneira um pouco diferente. O servidor da Web reconhece que a extensão do arquivo é algo diferente de .html - por exemplo, a extensão do arquivo pode ser .aspx, que é um arquivo ASP.NET - e chama o aplicativo Web apropriado para processar a requisição.

❸ O aplicativo da Web lê o arquivo solicitado no sistema de arquivos.

❹ Em vez de retornar o arquivo diretamente, o aplicativo Web processa o arquivo e executa todas as instruções que encontrar (veja abaixo).

❺ Essas instruções podem carregar dados de um banco de dados ou executar algum outro processamento. O aplicativo da Web gera dinamicamente algum HTML de acordo com as instruções no arquivo e retorna o código HTML gerado para o servidor da Web.

❻ O servidor Web encaminha esse conteúdo HTML para o navegador. O navegador não sabe como a geração do HTML ocorreu - apenas trata o HTML da mesma maneira como se fosse um arquivo estático.

- A maneira exata como o aplicativo Web gera o HTML varia de aplicação para aplicação, assim como a linguagem de computador usada para especificar as instruções para a aplicação Web.
- Normalmente, no entanto, o arquivo que contém as instruções é uma mistura de HTML e instruções especiais de programação.
- O HTML descreve a aparência e a estrutura da página da Web e as instruções de programação descrevem como acessar as informações no banco de dados.
- Quando o servidor Web processa o arquivo, ele substitui as instruções especiais de programação por HTML comum.
- As instruções informam a aplicação Web como gerar o HTML que será usado para substituir as instruções.

- Um exemplo ajuda a ilustrar como isso funciona. Veja como a página Web estática pode parecer se for gerada dinamicamente usando o ASP.NET:

```
  DOCTYPE HTML PUBLIC "- / / W3C / / DTD HTML 4.01 Transitional / / EN">
<html>
  <head>
    Example <% =% Page.Title> </ title>
    <link rel="stylesheet" href="/style.css" type="text/css" />
  </ head>
  <body>
    <asp:image id="imgLogo" ImageUrl="~/images/logo.jpg" height="50px" width="200px" BorderWidth="0px" runat="server">
</ asp: image>
    <h1> <asp:label id="lblPageTitle" Text='Welcome para Example.com' Runat=> </ asp: label> </ h1>
    <p> Bem-vindo ao site da Example.com. Nós esperamos que você vai encontrar muitos dos exemplos aqui. </ P>
      <p> A lista completa dos exemplos está no <asp:HyperLink id="lnkExamples" NavigateUrl='~/example-list.html'
runat=> nesta página web. </ asp: HyperLink> </ p>
  </ body>
</ html>
```

```
<!DOCTYPE html>
<html>

 <head>

    <title><%=Page.Title%></title>
    <link rel="stylesheet" href="/style.css" type="text/css" />

  </head>

  <body>

    <asp:image id="imgLogo" ImageUrl="~/images/logo.jpg" height="50px" width="200px" BorderWidth="0px"
    runat="server"></asp:image>

    <h1><asp:label id="lblPageTitle" Text='Welcome to Example.com' Runat="server"></asp:label> </h1>

    <p>Welcome to the Example.com Web site. We hope you will find lots of examples here.</p>

    <p>The full list of examples is on <asp:hyperlink id="lnkExamples" NavigateUrl='~/example-list.html'
    runat="server"> this Web page.</asp:hyperlink></p>

  </body>
</html>
```

- Os detalhes de como isso funciona não são importantes no momento. No entanto, você pode ver que certas partes da página original foram substituídas por códigos diferentes que não são HTML (as tags que começam <asp: não são HTML).
- O aplicativo ASP.NET da Web substitui essas tags por HTML válido quando processa a página, para que o resultado seja semelhante à página estática que você viu anteriormente. No contexto do ASP.NET, normalmente haveria algum código do Visual Basic ou C # que funcionaria com esta página para criar os resultados dinâmicos.
- **Fonte**
- https://msdn.microsoft.com/pt-br/library/bb330932(en-us,VS.80,lightweight).aspx

Códigos do status HTTP

- https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status

MDN web docs
moz://a

Tecnologias ▾ Referências e guias ▾ Comentários ▾

Códigos de status de respostas HTTP

Tecnologia Web para desenvolvedores > HTTP > Códigos de status de respostas HTTP

Nesta página

- Respostas informativas
- Respostas de sucesso
- Mensagens de redirecionamento
- Respostas de erro do Cliente
- Respostas de erro do Servidor
- Veja também

Tópicos relacionados

HTTP

Guides:

- Resources and URIs
- HTTP guide
- HTTP security

HTTP access control (CORS)

Os códigos de *status* das respostas HTTP indicam se uma requisição HTTP foi corretamente concluída. As respostas são agrupadas em cinco classes:

1. Respostas de informação (`100-199`),
2. Respostas de sucesso (`200-299`),
3. Redirecionamentos (`300-399`)
4. Erros do cliente (`400-499`)
5. Erros do servidor (`500-599`).

Os status abaixo são definidos pela seção 10 da RFC 2616. Você pode encontrar uma versão atualizada da especificação na RFC 7231.

Se você receber uma resposta que não está nesta lista, é uma resposta não padrão, provavelmente personalizada pelo software do servidor.

Respostas informativas

Códigos do status HTTP

- https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml

Hypertext Transfer Protocol (HTTP) Status Code Registry

Last Updated
2018-09-21
Available Formats
XML HTML Plain text

Registry included below

- HTTP Status Codes

HTTP Status Codes

Registration Procedure(s)
IETF Review
Reference
[RFC7231]
Note

```
1xx: Informational - Request received, continuing process
2xx: Success - The action was successfully received, understood, and accepted
3xx: Redirection - Further action must be taken in order to complete the request
4xx: Client Error - The request contains bad syntax or cannot be fulfilled
5xx: Server Error - The server failed to fulfill an apparently valid request
```

Available Formats
CSV

Value	Description	Reference
100	Continue	[RFC7231, Section 6.2.1]
101	Switching Protocols	[RFC7231, Section 6.2.2]
102	Processing	[RFC2518]
103	Early Hints	[RFC8297]
104-199	Unassigned	
200	OK	[RFC7231, Section 6.3.1]
201	Created	[RFC7231, Section 6.3.2]
202	Accepted	[RFC7231, Section 6.3.3]
203	Non-Authoritative Information	[RFC7231, Section 6.3.4]
204	No Content	[RFC7231, Section 6.3.5]
205	Reset Content	[RFC7231, Section 6.3.6]
206	Partial Content	[RFC7233, Section 4.1]
207	Multi-Status	[RFC4918]
208	Already Reported	[RFC5842]
209-225	Unassigned	
226	IM Used	[RFC3229]
227-299	Unassigned	
300	Multiple Choices	[RFC7231, Section 6.4.1]
301	Moved Permanently	[RFC7231, Section 6.4.2]
302	Found	[RFC7231, Section 6.4.3]