

# Programação para a Internet

*Internet, Web, HTTP e requisições*

---

Ely – [elydasilvamiranda@gmail.com](mailto:elydasilvamiranda@gmail.com)

# WEB x Internet

São a mesma coisa? Qual sua opinião?

# A Internet

- É um conjunto de redes de computadores de alcance mundial;
- “Roda” sobre um conjunto de protocolos;
- Disponibiliza serviços aos usuários, por exemplo:
  - Correio eletrônico;
  - Acesso remoto via VPNs;
  - Colaboração;
  - Compartilhamento de arquivos;
  - Transmissão multimedia;
  - World Wide Web.

# A Internet

- Através do TCP/IP, tornou-se um grande sistema distribuído;
- Tornou-se ultra popular com a criação da World Wide Web(1992).



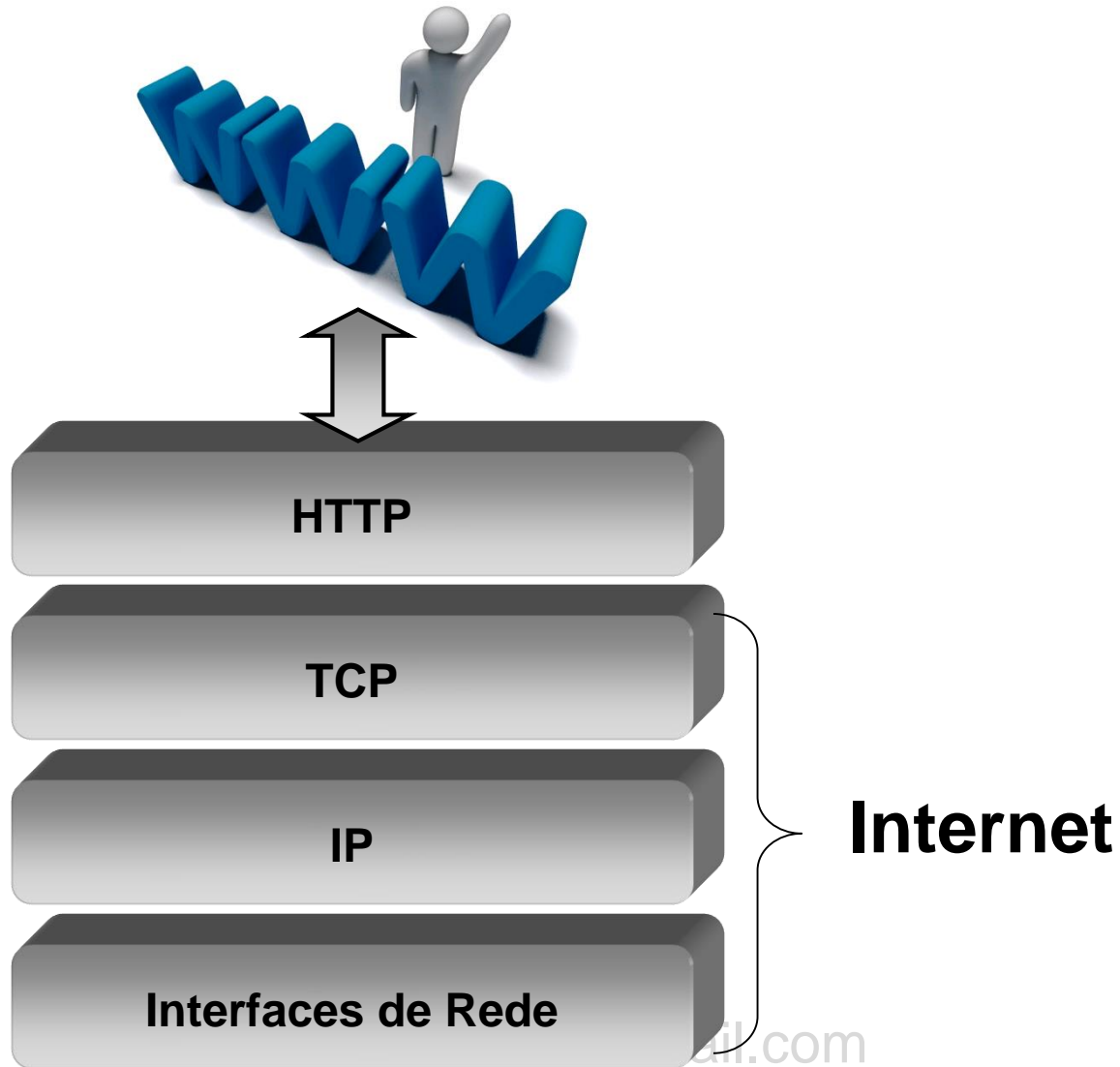
# World Wide Web

- Clientes e servidores se comunicam na Web através de redes *com ou sem fio* sobre o protocolo HTTP.



# Internet x Web

- WWW é uma aplicação que roda sobre a Internet.

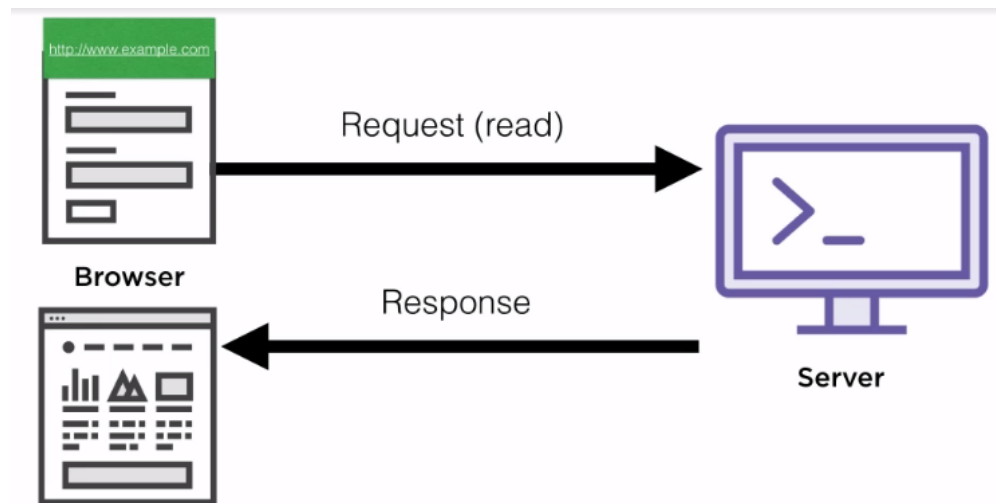


# World Wide Web

- Elementos correlatos:
  - HTTP: protocolo de aplicação usado para solicitar/receber os dados;
  - HTML: formato de Hipertexto predominante;
  - URL: caminho ou local de um recurso;
  - Recurso: página, arquivo binário, serviço...

# Interação Cliente-Servidor

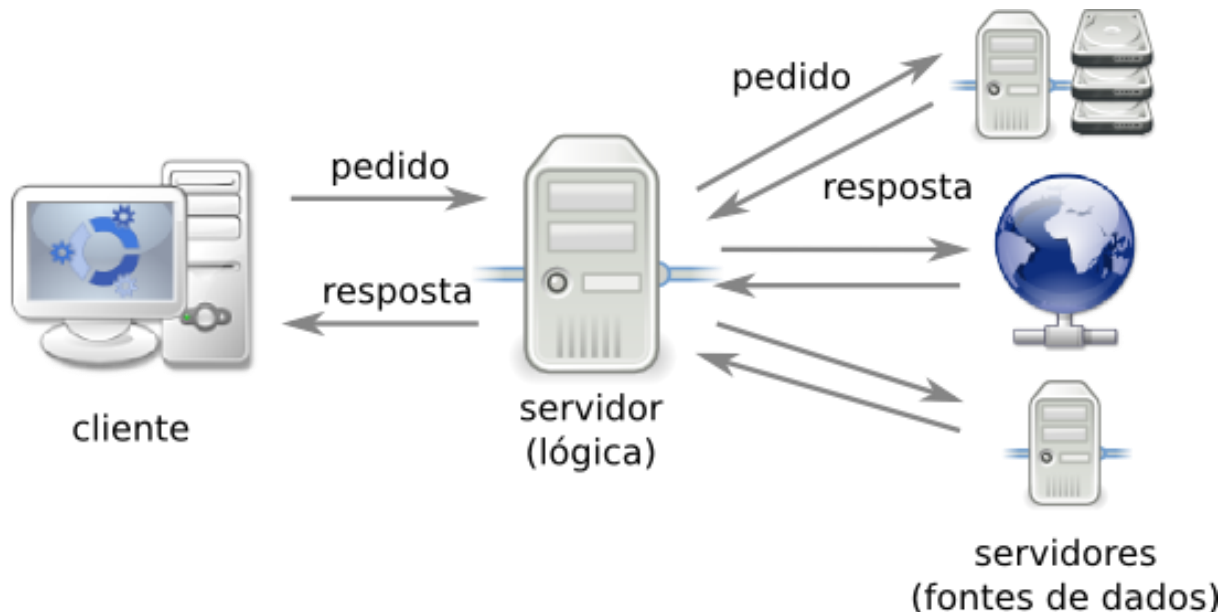
- Quando um cliente faz uma requisição, ele digita uma URL ou submete um formulário;
- O servidor então faz algum processamento e retorna uma resposta ao cliente;





# Servidor Web

- Recebe uma solicitações e devolve algo para o cliente Web:
  - Página, arquivo de áudio, imagem, som, pdf, mensagem de erro e etc;
- Web server = HTTP server.



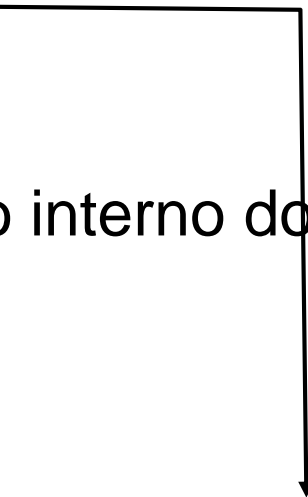
# Servidor Web

- Tarefas básicas:
  - Aceitar conexões de rede;
  - Responder solicitações;
    - Responde apenas **páginas estáticas**;
    - Uma página estática apenas repousa dentro de sua estrutura interna de diretórios;
    - Dada uma solicitação, o servidor busca a página na sua estrutura de diretórios e a devolve para o cliente como está;
    - Todo cliente vê a mesma coisa;
  - Encerrar conexões;

# Cliente Web

- Permite ao usuário **final** fazer solicitações ao servidor, exibindo para ele o resultado pedido;
- Não necessariamente um computador tradicional:
  - Celulares;
  - Tablets;
  - Eletrodomésticos;
  - Dispositivos de automóveis;
  - Scripts;
  - Etc.

# HTTP - Características

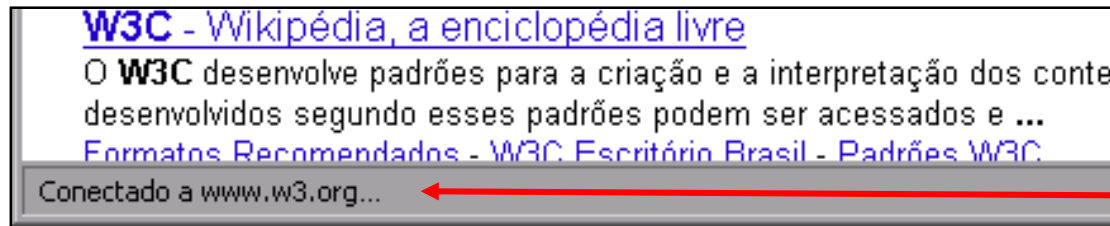
- HyperText Transport Protocol;
  - Mecanismo request-response:
    - A conversação entre cliente e servidor começa pelo cliente através de uma requisição/solicitação;
    - O servidor gera *sempre* uma resposta:
      - retorna o que foi solicitado ou
      - uma erro como acesso proibido ou erro interno do servidor;
- 
- A menos que o servidor esteja indisponível

# HTTP - Características

- Sem estado(*stateless*):
  - O servidor não mantém nenhuma informação de requisições passadas;
- Suporte a metadados:
  - Metainformações podem ser incluídas nas mensagens, incluídas nos pacotes HTTP. Ex: endereço do cliente, tipo de navegador, parâmetros de formulário...

# HTTP - Características

- A cada requisição é feita uma nova conexão

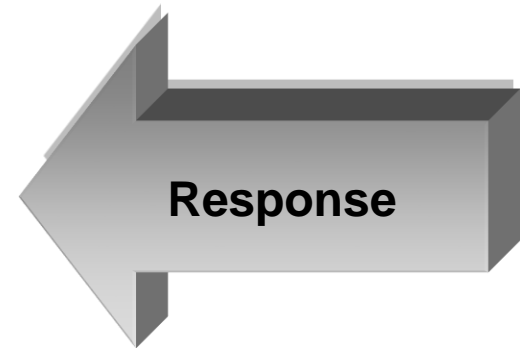
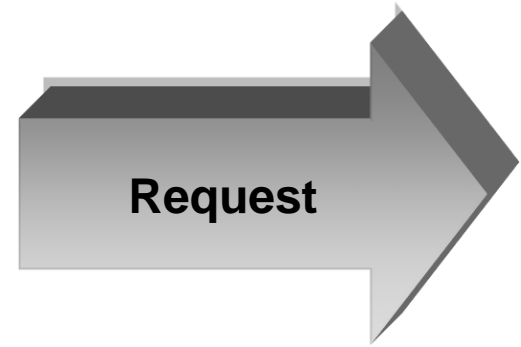


- Um pedido/requisição pode gerar o download de outros arquivos:



# HTTP

- A Solicitação/requisição é composta por:
  - Página ou recurso a ser acessado;
  - Método HTTP;
  - Cabeçalho com parâmetros.
- A Resposta é composta por:
  - Código de Status;
  - Cabeçalho com o tipo de conteúdo;
  - Conteúdo propriamente dito.



# HTTP

- URL:

***protocolo://servidor.domínio:porta/caminho/recurso***

- A porta é opcional em alguns casos, como a porta 80
- O caminho e o recurso
- As URLs podem conter parâmetros depois do nome do recurso começando com “?” e separados por “&”



# HTTP

- Exemplos de URL:

***http://localhost:8080/minha\_app/index.html***

servidor

caminho

recurso

***http://www.google.com/search?q=elvis***

domínio

parâmetros

***http://www.google.com/search?q=elvis&lr=lang\_pt***

# HTTP

- A primeira parte da solicitação é método HTTP;
- O método informa ao servidor o tipo de solicitação que está sendo feita e como a mensagem será formatada;
- Alguns métodos: GET, POST, PUT e DELETE.

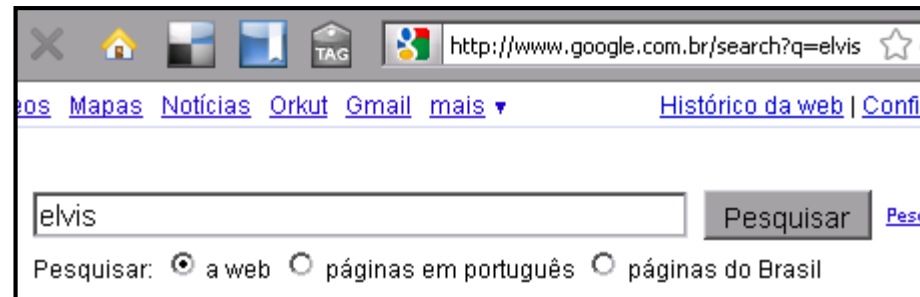
```
GET /index.html HTTP/1.1  
Host: www.exemplo.com  
User-agent: Mozilla/4.0
```

```
POST /index.html HTTP/1.1  
Accept: text/html  
If-modified-since: Sat, 29 Oct1999 19:43:31 GMT  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 37
```

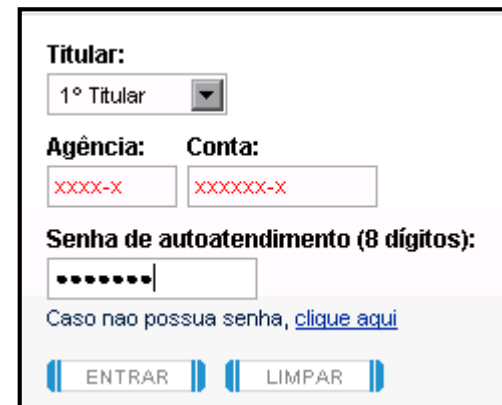
```
Nome=Pedro&Idade=25&Curso=Computacao
```

# HTTP

- Métodos:
  - Get: as informações/parâmetros enviados ao servidor estão contidas na URL



- POST: os parâmetros são incluídos no corpo da mensagem HTTP

A screenshot of a login form. It contains the following fields and labels: 'Titular:' with a dropdown menu showing '1º Titular'; 'Agência:' with a text input field containing 'xxxx-x'; 'Conta:' with a text input field containing 'xxxxxx-x'; and 'Senha de autoatendimento (8 dígitos):' with a text input field containing '.....'. Below the password field, there is a link that says 'Caso não possua senha, clique aqui'. At the bottom, there are two buttons: 'ENTRAR' and 'LIMPAR'.

# Métodos HTTP

GET /index.html HTTP/1.1

Host: exemplo.com

Referer: search.yahoo.com

Método + Recurso

Cabeçalho

Request

HTTP/1.1 200 OK

Date: Wed, 28 Jan 2009 19:32:18 GMT

Server: Apache/2.2.3 (CentOS)

Content-Length: 438

Connection: close

Content-Type: text/html; charset=UTF-8

Código de status

Cabeçalho

Response

<html>

<head>

<title>página web de exemplo</title>

</head>

<body>

.....

...

..

Corpo

# Métodos HTTP

- Os métodos HTTP são conhecidos como "verbos";
- Os mais comuns são análogos às necessidades CRUD;



# Métodos HTTP

- GET:
  - uma operação simples de leitura;
  - Usado para obter um ou mais recursos do sistema;
  - O verbo mais comum;
- PUT:
  - Utilizado para se atualizar um recurso específico e já existente;
  - Em alguns casos, é utilizado para atualizar uma lista de recursos
  - Ao utilizá-lo, espera-se que sejam enviadas todas a informações do objeto, não somente as que serão atualizadas.
- POST:
  - Usado para adicionar um novo recurso;
  - A API deve responder à criação de um novo recurso com um ID ou a URL do item recém criado;
- DELETE:
  - Apaga um recurso no servidor;

# Códigos de Status

- Presentes na resposta;
- Indicam sucesso ou falha da requisição.



# Tipos de Status

- 1XX: Informações
  - 100: Continuar
- 2XX: Sucesso
  - 200 – Recurso obtido com sucesso
- 3XX: Redirecionamento
  - 301 – Movido Permanentemente
  - 302 – Encontrado
  - 304 – Não modificado (usado com cache)
- 4XX: Erros do lado do cliente
  - 401: Não autorizado
  - 404: Recurso não encontrado
- 5XX: Erros do servidor
  - 500 – Erro interno do servidor



# Páginas estáticas x dinâmicas

- E se desejarmos que a hora atual do servidor apareça na página?
- E se desejarmos montar a página a partir de um banco de dados?
- Nesses casos precisamos de mais do que só o servidor Web:
  - Páginas deverão ser geradas instantaneamente, de forma dinâmica;
  - Precisamos de outra aplicação capaz de ser executada pelo servidor;
  - Geralmente, se implementa um módulo que é acoplado ao servidor Web para gerar as páginas dinamicamente.

# Alternativas para páginas dinâmicas

- Common Gateway Interface (CGI);
- Ruby On Rails;
- Django;
- PHP;
- Java: servlets, JSF.

*Não existe nenhuma bala de prata, o ideal é ter uma boa noção de alguns ambientes*

# Requisições HTTP via curl

- Curl é um software de linha de comando que permite fazer requisições HTTP;
- Para Windows, instalar o pacote Cygwin:
  - <https://cygwin.com/install.html>
- Ex:
  - > curl -v http://www.google.com
    - Retorna um redirecionamento (302 – Moved);
  - > curl -v http://www.google.com.br
    - Retorna a página do google (200 – ok)
  - > curl -v http://httpstat.us/500
    - Retorna um erro 500

# Programando com HTTP com Python

- Biblioteca requests;
- "HTTP para Humanos";
- Documentação:
  - [http://docs.python-requests.org/pt\\_BR/latest/](http://docs.python-requests.org/pt_BR/latest/)
- Instalação:
  - `pip install requests`

# Programando com HTTP com Python

```
# -*- coding: utf-8 -*-  
import requests  
response = requests.get('http://www.google.com')  
print(response.status_code)  
print(response.headers['content-type'])  
print(response.text)
```

# Programação para a Internet

*Internet, Web, HTTP e requisições*

---

Ely – [elydasilvamiranda@gmail.com](mailto:elydasilvamiranda@gmail.com)