

# Arquitetura Web

# A infra-estrutura da Internet é baseada no modelo cliente x servidor,

onde os clientes efetuam requisições junto aos servidores no intuito obter determinados

tipos de dados.

# Arquitetura Web

# Tecnologias como **HTML e CSS encontram-se no lado do cliente** (*client side*),  
pois podem ser interpretadas e renderizadas pelo navegador executado no  
computador do usuário  
não há necessidade de conexão com um servidor remoto.

# Arquitetura Web

Já tecnologias como PHP/ASP/JSP e outros encontram-se no lado do servidor (*server side*)

pois sua interpretação é feita remotamente, visto que o navegador não é capaz disso.

Para tal **é necessário que o cliente efetue requisições** a um **servidor** (*Apache*, *IIS*, *Tom-CAT*),

que após interpretar o script retorna como resultado dados que podem ser manipulados e exibidos pelo navegador.

# As requisições e respostas geradas precisam obedecer a um padrão,

para que ambos os lados possam trocar informações e compreender o que está sendo requisitado e respondido, Por esse motivo o **protocolo HTTP** é utilizado.

## Protocolo e Requisição HTTP

A sigla **HTTP** refere-se a **Protocolo de Transferência de Hipertexto** (*Hyper Text Transfer Protocol*). Tal protocolo **define regras e padrões que permitem que clientes e servidores Web possam se comunicar adequadamente, efetuando requisições e obtendo suas respectivas respostas.**

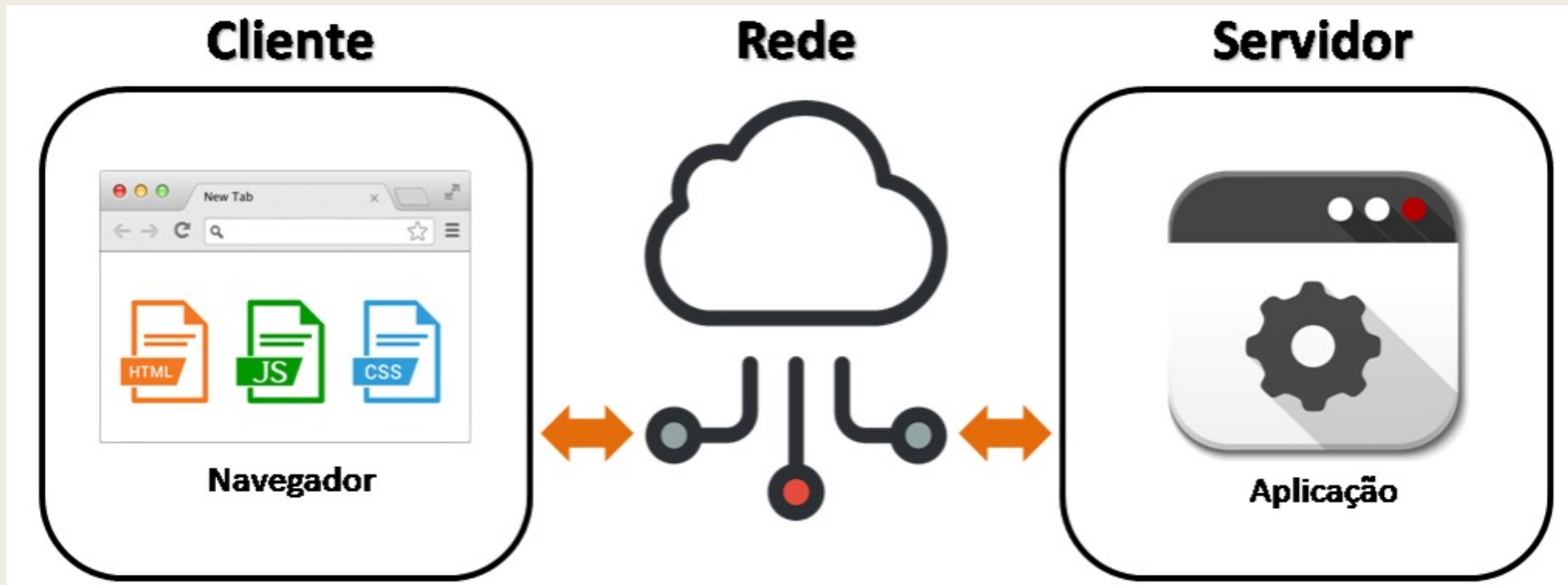
Por exemplo, quando um cliente acessa uma determinada URL (endereço web) através do navegador, uma requisição será efetuada a um determinado servidor, tendo como procedimentos os seguintes passos:

## Protocolo e Requisição HTTP

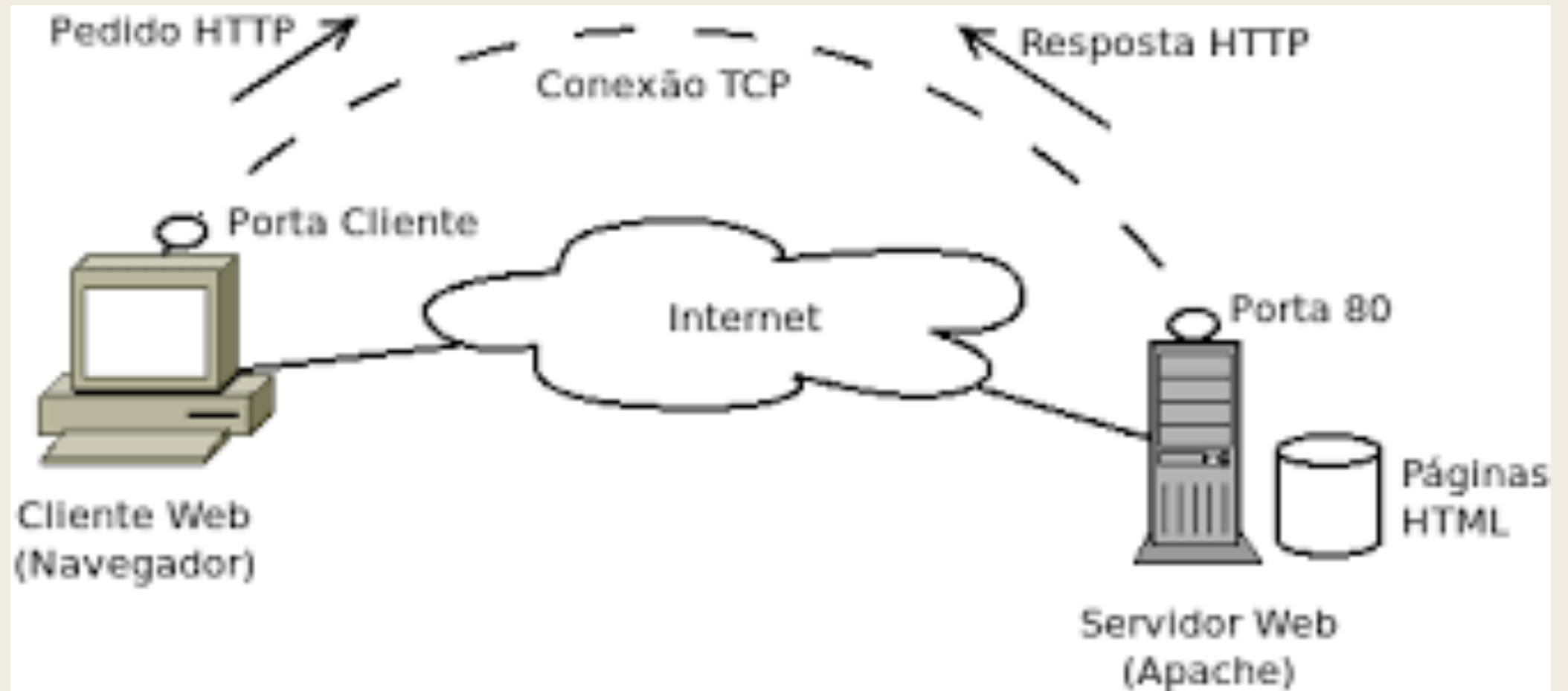
1. **Navegador** (cliente) efetua uma conexão com o servidor e envia uma solicitação HTTP para a página da web especificada;
2. **Servidor** recebe e verifica a solicitação, sendo adequada, o servidor devolve como resposta os dados para página especificada e um código indicando que a solicitação foi atendida corretamente. Caso o servidor, por algum motivo, não consiga atender a solicitação, enviará uma mensagem de erro juntamente com um código que permite sua identificação;

## Protocolo e Requisição HTTP

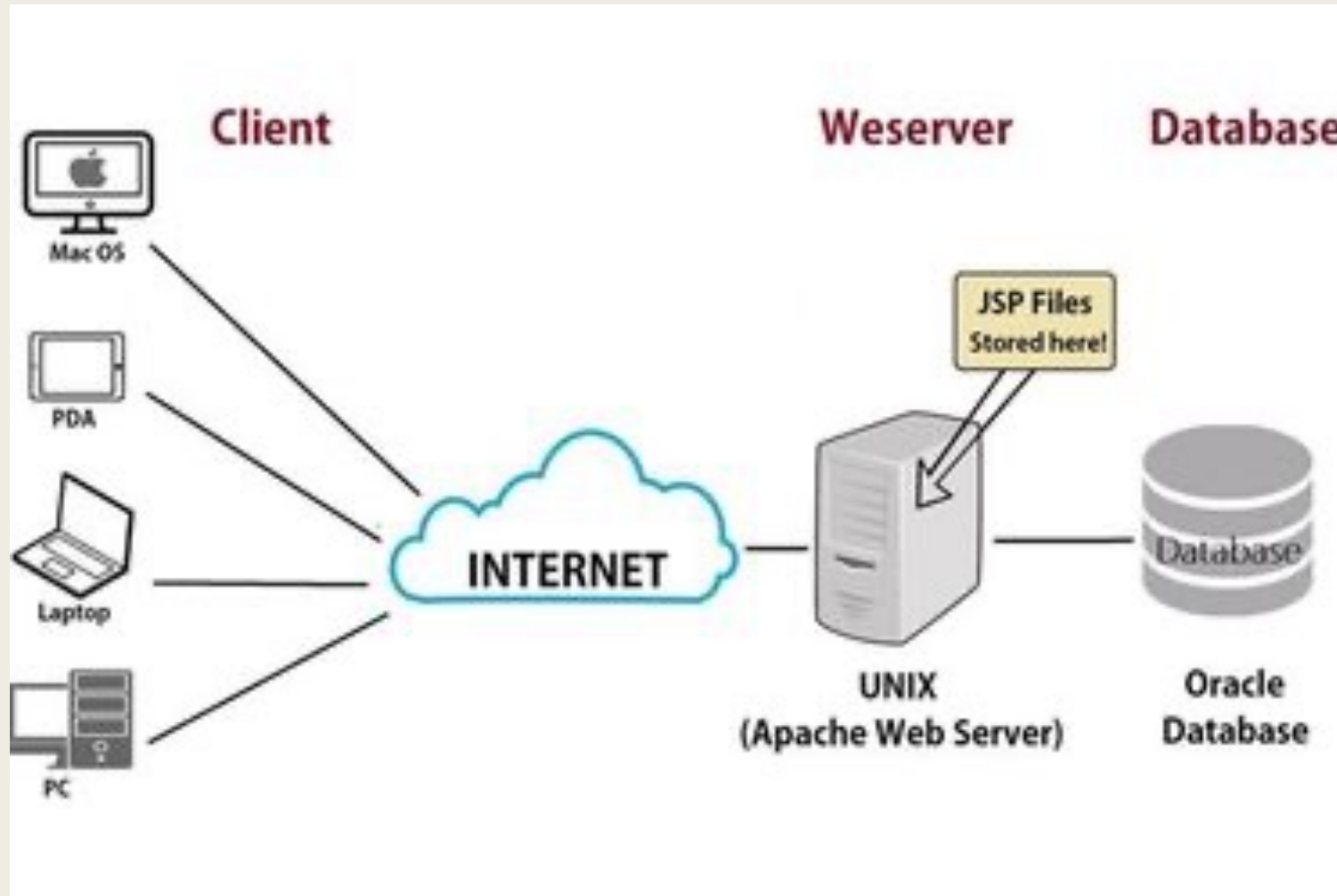
3. **Navegador** recebe a resposta do servidor(“página/código”ou“mensagem de erro / código”), e a conexão é finalizada;
4. **Navegador** analisa a resposta, caso indique que a solicitação foi atendida adequadamente o navegador adota os procedimentos necessários para exibir a página especificada na solicitação;



# Exemplo: Cliente servidor - HTTP

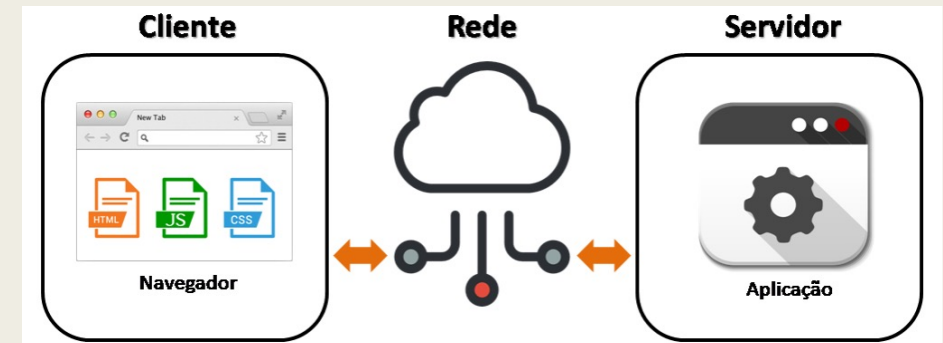
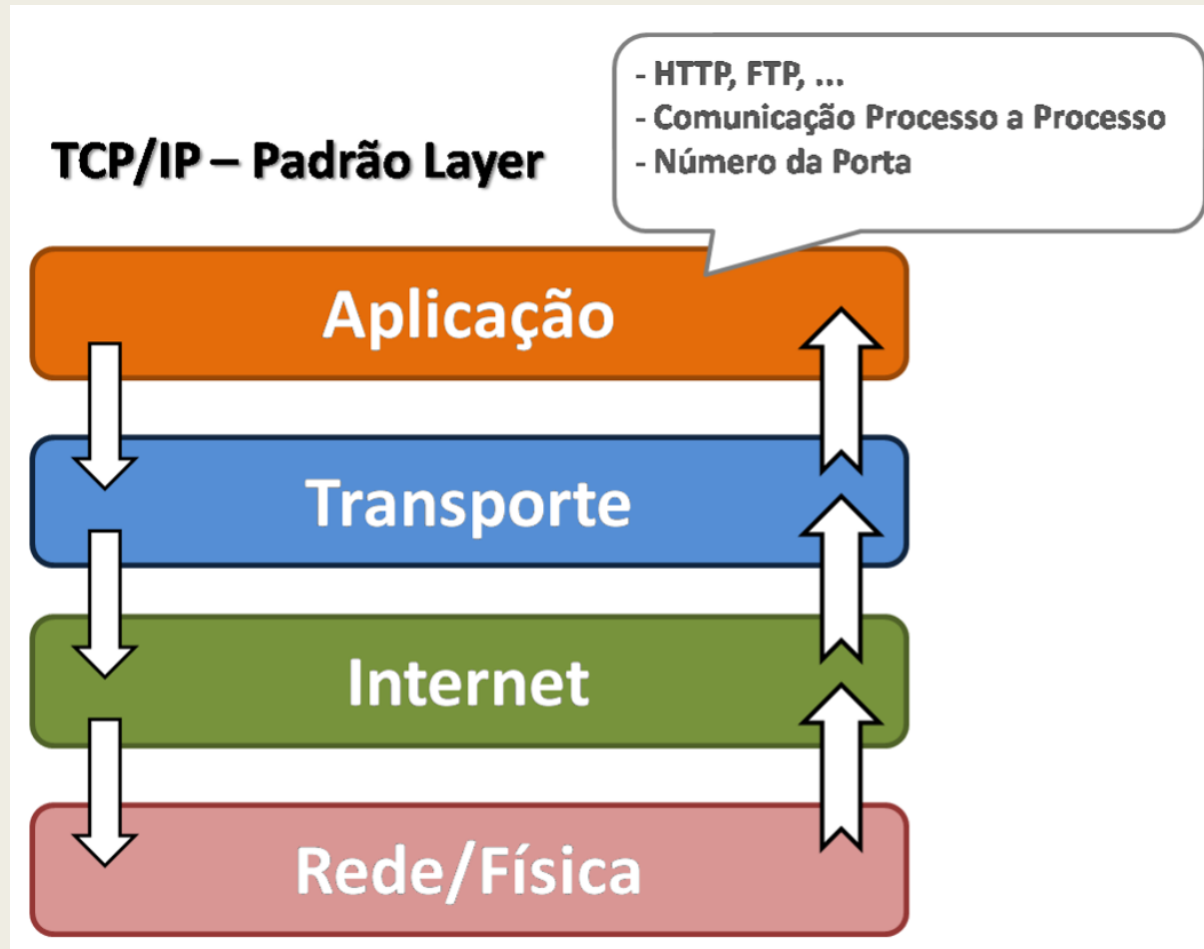


# Arquitetura WEB SERVER





# Comunicação processo a processo



# Requisições HTTP

- **requisições efetuadas por um cliente são denominadas *HTTP Request*.**
- As requisições são recebidas e processadas pelos servidores (como o Apache), que devolvem como resposta,
  - \* tanto os conteúdos das páginas web solicitadas pelo usuário,
  - \* quanto códigos que permitem ao cliente identificar se a requisição ocorreu como esperado.

## Códigos HTTP

- 1XX: Informativo – solicitação foi aceita ou continua em andamento;
- ● 2XX: Confirmação – solicitação foi concluída ou entendida;
- ● 3XX: Redirecionamento – um ou mais procedimentos são necessários para atender a solicitação;
- 4XX: Erro/Cliente – solicitação não pode ser atendida ou contém erro de sintaxe;
- 5XX: Erro/Servidor – servidor falhou durante o atendimento da solicitação;
- A lista completa pode ser encontrada em: <https://httpstatuses.com/>

# Métodos HTTP

- O **HTTP** possui um conjunto de **métodos** que podem ser **utilizados** quando uma **solicitação** é efetuada por um determinado cliente.
- Esses métodos definem o modo como os **parâmetros** são enviados quando uma **requisição** é efetuada ao servidor.

# principais - Métodos HTTP

- **GET:** é o método padrão utilizado ao efetuar uma solicitação, nele os parâmetros são passados juntamente com cabeçalho da requisição HTTP, sendo possível vê-los na URI. Exemplo:
  - - [www.pagina.com.br/cadastrar/cpf=00000000001](http://www.pagina.com.br/cadastrar/cpf=00000000001)
- **POST:** é um dos métodos que podem ser especificados no momento em que uma solicitação é efetuada. Diferentemente do método *GET*, o *POST* permite que os parâmetros sejam passados junto ao corpo da requisição, não sendo mais visíveis na URI. Exemplo: - [www.pagina.com.br/cadastrar](http://www.pagina.com.br/cadastrar)

Métodos	Responsabilidade
GET	Buscar dados
POST	Salvar os dados
PUT	Substitui determinado dado
DELETE	Apaga determinado dado

# Linguagem PHP

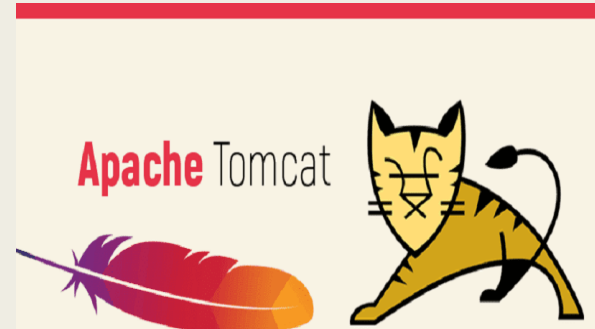
- Linguagem de script (interpretadas em tempo de execução – não são compiladas) que possui sua sintaxe baseada, em grande parte, nas linguagens C e Java, incluindo algumas características específicas.

O PHP possibilita o desenvolvimento de páginas dinâmicas, sua sigla significa Pré-processador de hipertexto (*Hypertext Preprocessor*).

É uma linguagem scripts do tipo *backend*, ou seja, é executada no servidor antes de enviada ao cliente (navegador).

Sendo assim, o servidor não envia código-fonte ao cliente, mas sim processa e transforma esse código em formato HTML, para após isso enviar ao cliente.

# Servidor HTTP - HTTPS



# aplicações

O servidor Apache ou também Apache HTTP Server dá suporte a aproximadamente **metade das aplicações web disponíveis atualmente.**

O Apache, assim como outros servidores web (IIS – *Internet information Server*), tem como objetivo principal dar suporte adequado ao maior número de clientes simultaneamente.

Ele é capaz de processar não apenas arquivos escritos em PHP, mas também outras linguagens, tais como Python e Perl, JSP,

O Apache pode ser visto, de modo simplificado, como um interpretador de código-fonte, **capaz de transformar a codificação interpretada em conteúdo HTML.**



# Pacotes de serviços integrados

**XAMPP, LAMPP, WAMPP**

**Ambiente de desenvolvimento integrado que permite criar aplicações WEB com Apache2, PHP e MySQL**

**<https://sourceforge.net/projects/wampserver/>**

**<https://www.apachefriends.org/download.html> (XAMMP)**

**<https://www.edivaldobrito.com.br/instale-lamp-no-linux-e-tenha-um-servidor-web-em-seu-pc/>**

# Instalar XAMMP e testar



# Instalar XAMMP e testar apache

- Teste no terminal: `apache2 -v`
- Crie arquivos dentro do diretorio `HTTP: /var/www/html`
- Páginas com conteudo html e abra o browser:
- `localhost://nome-arquivo.html`

# Verificar PHP

[illegible]

# Criar Arquivo PHP para testes

- Teste no Terminal:

`php -v`

- Crie um arquivo com o nome: `informacoes.php`

- No seu conteúdo adicione:

```
<? php
```

```
phpinfo();
```

```
?>
```

Salve o arquivo e abra-o no navegador

# Testar servidor funcionando em outras maquinas

- Identificar o IP da maquina

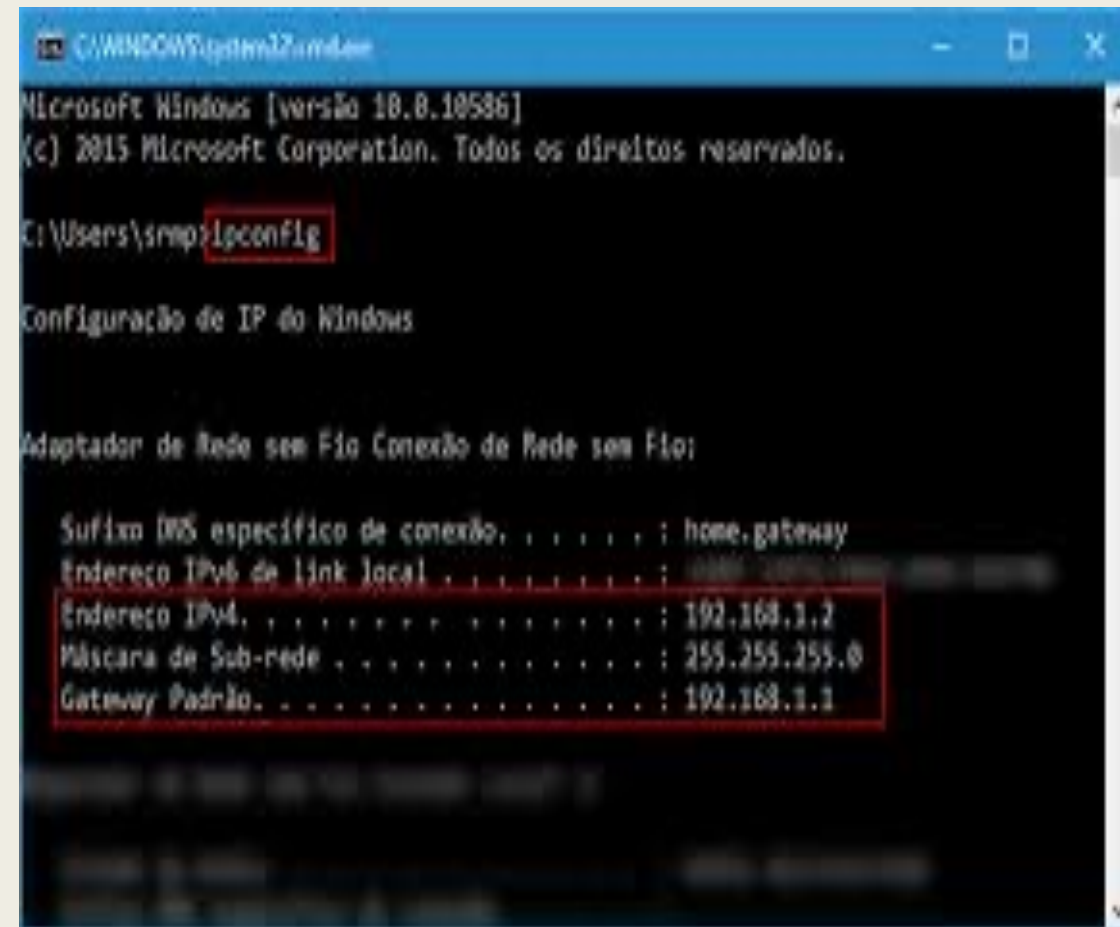
- no terminal digite:

no linux: ipconfig

no windows: ifconfig.

Ou acesse as configurações de rede e analise

o IPV4 da maquina



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [versão 10.0.10586]
(c) 2015 Microsoft Corporation. Todos os direitos reservados.

C:\Users\smp>ipconfig

Configuração de IP do Windows

Adaptador de Rede sem Fio Conexão de Rede sem Fio:

Sufixo DNS específico de conexão. . . . . : home.gateway
Endereço IPv6 de link local . . . . . : fe80::...
Endereço IPv4. . . . . : 192.168.1.2
Máscara de Sub-rede . . . . . : 255.255.255.0
Gateway Padrão. . . . . : 192.168.1.1
```

# PHP - início - TAGS php

## Estrutura básica de uma codificação PHP

```
<?php  
  
    echo "Hello";  
    print "World!";  
?>
```

### Estrutura Básica - PHP

[<?php] - define o início de um novo trecho de código em linguagem PHP  
[echo][print] - comandos de saída, permitem apresentar dados no navegador  
[?>] - define o término do trecho de código em linguagem PHP

(Arquivo-fonte: aula01/estrutura basica.php)



localhost/php/aula01/estrutura\_basica.php

HelloWorld!

# PHP - início - TAGS php -escape

- `<p>`Isto vai ser ignorado pelo PHP e exibido pelo navegador.`</p>`  
`<?php echo 'Enquanto isto vai ser interpretado.'; ?>`  
`<p>`Isto também vai ser ignorado pelo PHP e exibido no navegador.`</p>`
- Exemplo #1 Exemplo mostrando a tag de fechamento incluindo uma nova linha final
- `<?php echo "Algum texto"; ?>`  
Sem nova linha  
`<?= "Mas uma nova linha agora" ?>`



# PHP - TAGS php

- Exemplos de entrar e sair do modo PHP:

```
<?php  
    echo 'Isto é um teste';  
?>
```

```
<?php echo 'Isto é um teste' ?>
```

```
<?php echo 'Nós omitimos a última tag de  
fechamento';
```

# PHP - Comentários

```
<?php
    echo 'Isto é um teste'; // Estilo de comentário de
uma linha em C++
    /* Este é um comentário de múltiplas linhas
    ainda outra linha de comentário */
    echo 'Isto é ainda outro teste';
    echo 'Um teste final'; # Este é um comentário de uma
linha no estilo shell
?>
<h1>Isto é um <?php # echo 'simples';?> exemplo.</h1>
<p>O cabeçalho acima irá dizer 'Isto é um exemplo'.</p>
```

# PHP - Comentários

```
<?php
```

```
/*
```

```
    echo 'Isto é um teste'; /* Este comentário irá causar um problema */
```

```
*/
```

```
?>
```

# PHP - Syntax

```
■ <!DOCTYPE html>
  <html>
  <body>

  <h1>My first PHP page</h1>

  <?php
  echo "Hello World!";
  ?>

  </body>
  </html>
```

# PHP - no case sensitive

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<?php
```

```
$color = "red";
```

```
echo "My car is " . $color . "<br>";
```

```
echo "My house is " . $COLOR . "<br>";
```

```
echo "My boat is " . $coLOR . "<br>";
```

```
?>
```

```
</body>
```

```
</html>
```

# PHP - no case sensitive

- ```
<!DOCTYPE html>
<html>
<body>
```

```
<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
```

```
</body>
</html>
```

- In PHP, keywords (e.g. `if`, `else`, `while`, `echo`, etc.), classes, functions, and user-defined functions are not case-sensitive.

# Variaveis

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). Rules for PHP variables:

- A variable starts with the \$ sign, followed by the name of the variable

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
- Variable names are case-sensitive (\$age and \$AGE are two different variables)

```
■ <?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

## Variáveis PHP / Sintaxe

```
<?php
$curso = "WEB Médio - Aluno 10!";
$disciplina = "PSW1";
$turma = 2022;
echo "[CURSO]: $curso<br>";
echo "[DISCIPLINA]: $disciplina<br>";
echo "[TURMA]: $turma";

?> |
```

### Variáveis - PHP

---

[\$curso] - variáveis php sempre começam pelo caractere especial "\$"

[\$disciplina] - variáveis php não precisam ser declaradas, basta usá-las

[\$turma] - PHP é dito "fracamente tipado", variáveis podem receber qualquer tipo de valor  
(Arquivo-fonte:

aula01/variaveis.php)



# ECHO - saída de dados para tela

- ```
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
```

- ## Concatenação
- ```
<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!";
?>
```

```
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ",
"with multiple parameters.";
?>
```

# Print - Saída de dados para tela

## ■ <?php

```
$txt1 = "Learn PHP";  
$txt2 = "W3Schools.com";  
$x = 5;  
$y = 4;
```

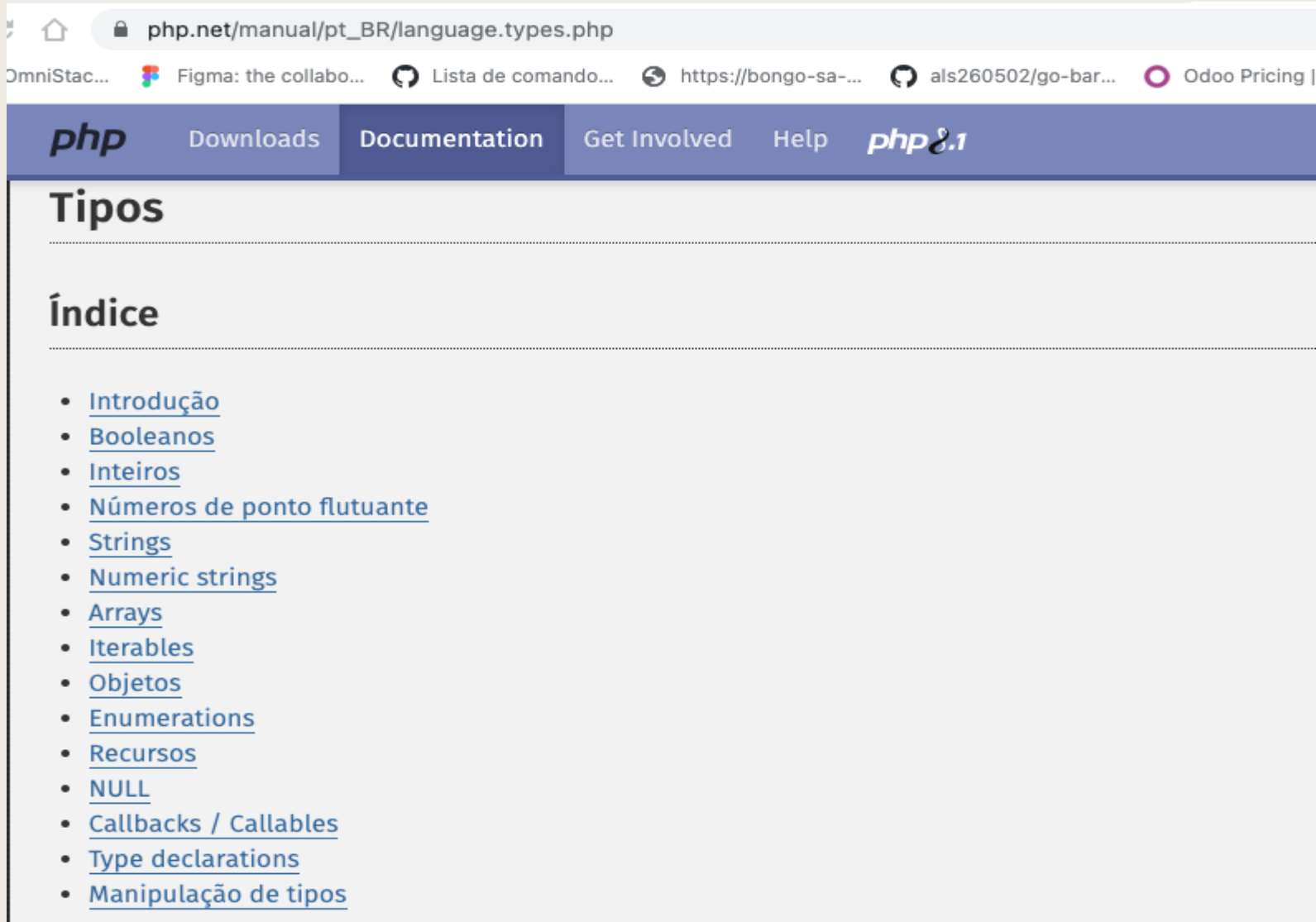
```
print "<h2>" . $txt1 . "</h2>";  
print "Study PHP at " . $txt2 . "<br>";  
print $x + $y;  
?>
```

```
<?php  
print "<h2>PHP is Fun!</h2>";  
print "Hello world!<br>";  
print "I'm about to learn PHP!";  
?>
```

# REFERENCIAS

- [https://www.w3schools.com/php/php\\_echo\\_print.asp](https://www.w3schools.com/php/php_echo_print.asp)
- [https://www.php.net/manual/pt\\_BR/](https://www.php.net/manual/pt_BR/)

# PHP - TIPOS



The screenshot shows the PHP 8.1 documentation page for language types in Portuguese. The browser address bar displays 'php.net/manual/pt\_BR/language.types.php'. The navigation bar includes links for 'php', 'Downloads', 'Documentation' (which is highlighted), 'Get Involved', 'Help', and 'php 8.1'. The main heading is 'Tipos'. Below it is the 'Índice' (Index) section, which contains a list of links to various data types and concepts in PHP.

php Downloads Documentation Get Involved Help php 8.1

## Tipos

---

### Índice

---

- [Introdução](#)
- [Booleanos](#)
- [Inteiros](#)
- [Números de ponto flutuante](#)
- [Strings](#)
- [Numeric strings](#)
- [Arrays](#)
- [Iterables](#)
- [Objetos](#)
- [Enumerations](#)
- [Recursos](#)
- [NULL](#)
- [Callbacks / Callables](#)
- [Type declarations](#)
- [Manipulação de tipos](#)