

# LÓGICA DE PROGRAMAÇÃO

Professor Jefferson Chaves

[jefferson.chaves@ifc-araquari.edu.br](mailto:jefferson.chaves@ifc-araquari.edu.br)



Técnico Integrado em  
**INFORMÁTICA**

- Como a web funciona:
  - Ambiente Cliente x Servidor;
  - Aprendendo a se comunicar na web: o protocolo http;
  - Manipulando formulários
  - Guardando seus dados com sessões e cookies;

# **PROGRAMAÇÃO DE COMPUTADORES**



## **Arquitetura da web: Clientes & Servidores**

Jefferson de Oliveira Chaves - IFC  
Araquari

- Certas tecnologias, tais como **HTML, CSS** e **JavaScript**, podem ser executadas apenas em navegadores (*browsers*);
- A essas tecnologias damos o nome de tecnologias **Client-Side** ou tecnologias do lado do cliente;

- Outras tecnologias, como o **PHP** e **sistemas gerenciadores de bancos de dados**, precisam de programas instalados para executar alguma tarefa.
- Para que seu vovô ao acessar um site de gravatas **não tenha** que instalar e configurar todos esses programas, é configurado um outro computador com todos esses programas. **O chamamos de servidor;**
- As tecnologias que rodam em um servidor damos o nome de tecnologias **Server-Side** ou tecnologias do lado do servidor;





# fantasticas Aplicacoes

E COMO SE COMUNICAM

# PROGRAMAÇÃO DE COMPUTADORES

# ARQUITETURA DA WEB: CLIENTES & SERVIDORES

1

Cliente (navegador) REQUISITA  
a página pela url, formulário  
ou link



2

Navegador interpreta e  
executa o conteúdo da  
RESPOSTA como uma página  
html.



2

Servidor (apache)  
recebe a  
**REQUISIÇÃO**,  
executa o código  
PHP e **RESPONDE**  
com o resultado

- Toda essa comunicação entre o navegador e o servidor segue uma série de etapas;
- Essas etapas são chamadas de protocolo;
- O **HTTP** é o **protocolo** usado na web;

# **PROGRAMAÇÃO DE COMPUTADORES**

**HTTP, o idioma dos  
navegadores e  
servidores web**

- Imagine o cenário fictício:
  - Você mora no prédio localizado no endereço IP 185.186.87.88;
  - seu apartamento é o de número 80;
  - O seu trabalho é enviar páginas com as informações variadas para quem as solicita através do correio (**SERVIDOR**).

- Uma pessoa qualquer (**CLIENTE**) te envia uma carta solicitando uma página com informações sobre futebol, por exemplo.
- Você recebe a carta, analisa a solicitação, monta a página com as informações solicitadas,
- Coloca a página num envelope e a envia de volta para o remetente.

- Esta comunicação se dá num idioma próprio, que somente vocês entendem.



- Esse idioma é o **HTTP** ou *Hypertext Transfer Protocol*;
- Este é o idioma que os **navegadores** e os **servidores web** conversam.
- É por meio do HTTP que o seu navegador informa ao servidor web qual a sua **versão**, qual o seu **idioma**, se aceita **conteúdo compactado** ou não e qual **página** foi solicitada.

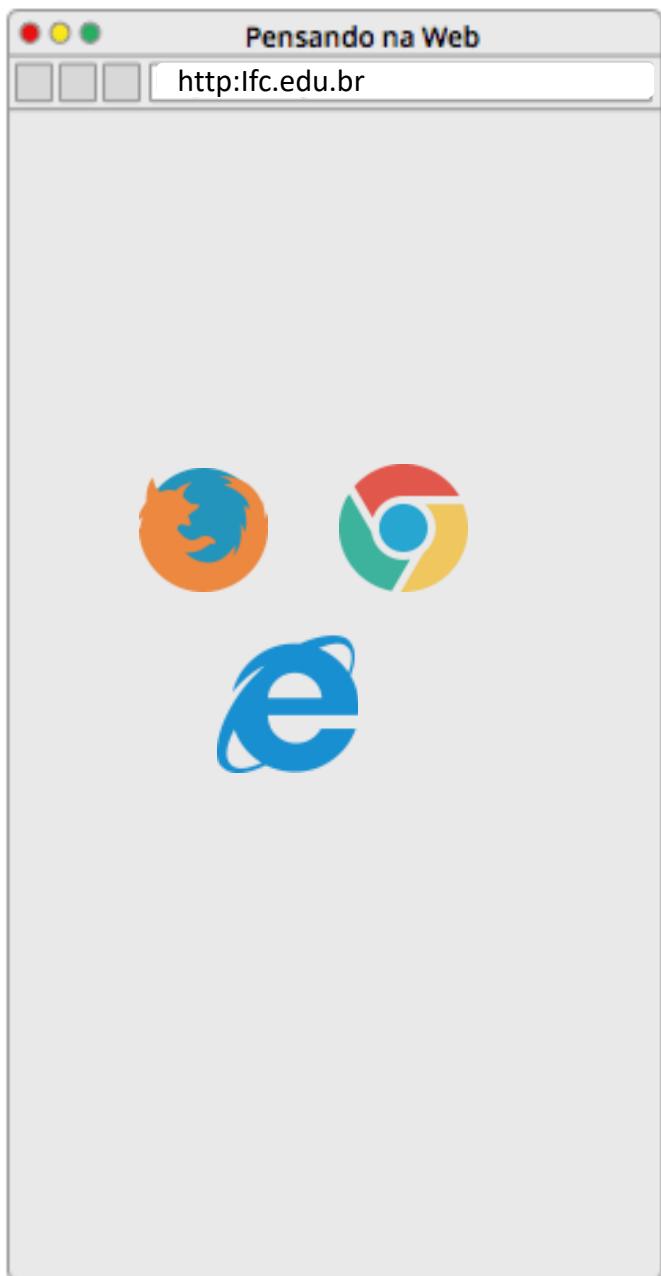
- É por meio também do **HTTP** que o servidor web informa ao seu navegador se a **página solicitada existe**, qual o seu **formato**, se a página enviada foi **compactada**, se existe algum **cookie** para ser gravado no seu computador e, principalmente, o **conteúdo da página** solicitada.

- Quando o navegador solicita uma página web é chamado de **requisição**;
- Quando o servidor web envia a página web solicitada de volta para o navegador é chamado de **resposta**.

- Ao requisitar uma página, o servidor pode responder com:
  - **200 (ok)**: solicitação e resposta foram enviadas;
  - **403 : (acesso proibido)**: O pedido é reconhecido pelo servidor mas este recusa-se a executá-lo;
  - **404 (Não encontrado)**: O recurso não existe ou não foi encontrado;
  - **500 (Erro do servidor)**: Indica um erro do servidor ao processar a solicitação.
  - **503 (Serviço indisponível )**: O servidor está em manutenção ou não consegue dar conta dos processamentos de recursos

o HTTP é considerado um protocolo **sem estado** ou *stateless*.

## Cliente



www.ifc.edu.br

185.186.87.88

Boa tarde 185.186.87.88, posso  
me conectar na porta 80?

Boa tarde 177.178.79.80, pode conectar.

Ok, iniciarei a conexão.

### Conexão TCP

#### Requisição HTTP

GET / HTTP/1.1  
Host: www.ifc.edu.br  
(...)

#### Resposta HTTP

HTTP/1.1 200 OK  
Content-Type: text/html  
  
<!doctype html>  
<html lang="en">  
...  
</html>

Parser do conteúdo  
HTML e exibição da  
página.

## Servidor

Servidor DNS



Servidor Web



Então  
precisarei  
de um  
servidor?

**Servidor** é um **programa de computador** responsável por aceitar pedidos **HTTP** de clientes, geralmente os navegadores, e servi-los com respostas **HTTP**, geralmente, páginas web, tais como documentos **HTML** com objetos incluídos (imagens, etc.)

- O **servidor Apache** (ou Servidor HTTP Apache, em inglês: *Apache HTTP Server*, ou simplesmente: *Apache*) é o mais bem sucedido servidor web livre.
- **66% dos sites do mundo** estão hospedados em servidores Apache;



# LÓGICA DE PROGRAMAÇÃO

# AMBIENTE PARA PROGRAMAR PHP

- A fim de facilitar a instalação do PHP, Apache além de outras configurações, um grupo de programadores criou o **XAMPP**, que **é um pacote** com tudo que precisamos.
- Sua instalação é bem simples. Pode ser baixado em:  
[https://www.apachefriends.org/pt  
br/index.html](https://www.apachefriends.org/pt_br/index.html)



- O **X** significa que roda em qualquer sistema operacional;
- O **A** é de Apache;
- O **M** é o banco de dados;
- O **P** é de PHP;
- O **P** é de Pearl;

- Sistemas para web, a medida que evoluem tornam-se complexos de se manter;
- Para evitar que isso aconteça, devemos seguir algumas práticas recomendadas pelo PHP-FIG.



- Entenda o HTTP;
- Mantenha a lógica separada da apresentação;
- Apresente seus dados usando as **short tags** na apresentação;
- Desenvolva código que pode ser reutilizado;
- Verifique e valide a entrada de dados;
- Pense na organização dos arquivos e pastas;



# **PROGRAMAÇÃO DE COMPUTADORES**

# **BOAS PRÁTICAS**

<b>Tag</b>	<b>Short tag</b>
<?php echo 'oie' ?>	<?= 'oie' ?>
if(condicao){ }	if(condicao): / endif;
foreach(condicao){ }	foreach (condicao): / endforeah;



# **PROGRAMAÇÃO DE COMPUTADORES**

## **LEITURA DE DADOS**

Jefferson de Oliveira Chaves

- Como visto o PHP pode ser usado para criar programas **desktop ou para web**.
- Para programa em linha de comando, a leitura de dados do teclado é feita pela função **fgets(STDIN);**

Quando se programa para web o cenário é diferente e a interface entre o usuário e o PHP é feita por meio dos **formulários HTML**.

Um formulário web geralmente é parecido com:

```
<p>formulário: </p>
```

```
<form method="get">
```

```
    <input type="text" name="nome" required />
```

```
    <input type="submit" value="cadastrar" />
```

```
</form>
```

formulario:

cadastrar

# **PROGRAMAÇÃO DE COMPUTADORES**

## **FORMULÁRIOS HTML**

Jefferson de Oliveira Chaves

- Os formulários HTML são a principal forma de entrada para qualquer aplicação para *web*.
- Cadastros, *logins* e buscas são exemplos de usos de formulários;
- O google.com é um site que possui um único campo de formulário e nada mais.

- O formulário deve conter os atributos:
  - **method**: define a forma com que os dados serão enviados. São aceitos os métodos **GET** e **POST**.
  - **action**: define qual arquivo processará o conteúdo do formulário;

Opcionalmente o atributo usa-se o atributo `<form enctype="multipart/form-data">` para informar que este formulário possui um campo para upload de arquivos.

- O principal elemento de um formulário é o **input** (entrada);
- Os formulários possuem diferentes tipos de input's como:
  - caixa de texto;
  - caixa de senha;
  - caixas de seleção;
  - botão de enviar;
  - Mas não somente estes;

// Para mudar o tipo de input basta dizer qual o tipo desejado no campo **type**:

```
<form>
  <input type="text"      name="nome">
  <input type="number"   name="idade">
  <input type="submit">
</form>
```

```
print $_POST['name'];
```

- O HTML5 trouxe um número expressivo de novos campos: color, date, datetime, datetime-local, email, month, number, range, search, tel, time, url, week, entre outros.



Saiba mais sobre os formulários em [w3schools.com/html/html\\_form\\_input\\_types.asp](http://w3schools.com/html/html_form_input_types.asp)  
Pesquise sobre como usar expressões regulares em campos de formulários

- **O parâmetro action em formulários:**
- Define para onde os dados do formulários serão enviados quando os dados forem submetidos;
- Se o parâmetro **action** estiver vazio, o formulário enviará os dados para a própria página;

```
<form method="post" action="valida.php">  
  <input type="text" name="busca" >  
  <input type="submit">  
</form>
```

// Os dados desse formulário serão enviados para  
o arquivo valida.php

# PROGRAMAÇÃO DE COMPUTADORES

## ATRIBUTO NAME EM CAMPOS DO FORMULÁRIO



- Todos os campos dos formulário (**input**, **select**, **radio**) devem **OBRIGATORIAMENTE** possuir um nome;
- É por meio dos nomes de cada campo que conseguiremos recuperar os dados informados!

```
<form method="post" action="usuario.php">  
  <input type="text" name="cor_favorita" >  
  <input type="submit">  
</form>
```

// Podemos ler este formulário da seguinte forma:  
Formulário enviará para o arquivo **usuario.php** por  
**método post** os dados **cor\_favorita** preenchidos pelo  
usuário.

# **PROGRAMAÇÃO DE COMPUTADORES**

## **VARIÁVEIS SUPERGLOBAIS**

Jefferson de Oliveira Chaves

- O PHP possui variáveis que “sempre existem” em sua aplicação;
- Essas variáveis são chamada de **superglobais** ou **supervariáveis**;
- Estas variáveis estão disponíveis em todos escopos da aplicação;

- **Variável superglobal `$_GET`**
- Um array **associativo** de variáveis passadas para o script atual via o método **HTTP GET**;
- Os dados ficam expostos na *url*;
- O envio repetidas vezes não tem efeitos colaterais;
- Os dados podem ser usados para *bookmark (favoritos)*;

- **Variável superglobal `$_GET`**
- Em um URL sinal de interrogação (?) todo o conjunto `?chave=valor` pode ser acesso por meio da variável superglobal `$_GET`
- Se existirem mais valores na url eles serão separados pelo ampersand (&);

// O formulário a baixo, ao ser submetido:

```
<form method="GET" action="">  
    <input type="text" name="busca">  
    <input type="submit">  
</form>
```

// Formará a URL abaixo, onde instituto é a palavra digitada pelo usuário:



localhost/ifc/?busca=instituto



// Para acessar as variáveis que estão na URL, devemos utilizar a supervariável `$_GET`;

// Dessa forma, ao se executar o comando abaixo, o que será impresso na tela?

```
echo $_GET['busca']; //vai imprimir instituto
```

- **Superglobais `$_POST`**
- Um array associativo de variáveis passadas para o script atual via o método HTTP POST;
- Os dados ficam encapsulados dentro do corpo da mensagem;
- O usuário não “enxerga” os parâmetros enviados;
- Normalmente usado em formulários de *cadastrros, logins, etc.*;

O formulário a baixo, ao ser submetido:

```
<form method="POST" action="">  
    <input type="text" name="login">  
    <input type="password" name="senha">  
    <input type="submit">  
</form>
```

// Enviará os dados de forma **encapsulada** dentro do corpo da mensagem http.

```
<form method="POST" action="">  
    <input type="text" name="login">  
    <input type="password" name="senha">  
    <input type="submit">  
</form>
```

// Os dados vindos desse formulário, podem ser acessados por meio de:

```
$login = $_POST['login'];  
$senha = $_POST['senha'];
```

- Superglobais `$_SESSION`
- Um array associativo contendo variáveis de sessão disponíveis para a atual aplicação.
- `session_start()` - Inicia dados de sessão;
- `session_destroy()` – Limpa os dados da sessão;
- Armazena as informações no cookie chamado `PHPSESSID`



- Superglobais `$_SESSION`
- Um visitante acessando o seu web site ganha um identificador único, o assim chamado **id de sessão**.
- Este é salvo em um **cookie** do lado do usuário ou propagado via URL.
- Uma sessão dura enquanto o navegador estiver aberto;
- Uma sessão acaba ao fechar o navegador;



- **Superglobais `$_COOKIE`**
- Sua função principal é manter a persistência de sessões HTTP;
- Um exemplo prático, são os sites que criam um cookie para que você não precise digitar sua senha novamente ao retornar ao site.

- Superglobais `$_COOKIE`
- Um cookie ficará armazenado no navegador do usuário por um tempo determinado;

// Ex.: 1 - Cria um cookie

```
setcookie('usuario', 'Fulano');
```

// Ex.: 2 - Cria o novo cookie p/ durar duas horas

```
setcookie('nome', 'Ciclano', (time() + (2 * 3600)));
```

// Ex.: 2 - Cria um cookie que durará três dias

```
setcookie('usuario', 'Fulano', (time() + (3 * 24 * 3600)));
```

- O tempo de vida do cookie deve ser feito por meio da soma do número de segundos desejados com a função **time()**:
  - cada hora tem 3600 segundos
  - cada dia tem 86400 segundos.
- Ex.: : cinco horas são 18000 segundos, então:
  - **18000 + time( );**
  - **(3600 \* 5) + time( );**

- O cookie precisa existir para obtermos seu valor;
- Se passou da validade (expirou) ele não existe mais, então é boa prática sua verificação com a função **isset( )**;

// Pega o valor do Cookie 'usuario' definido anteriormente:

**\$valor = \$\_COOKIE['usuario']; // Fulano**

// Pega o valor do Cookie 'nome' definido anteriormente:

**\$valor = \$\_COOKIE['nome']; // Ciclano**

- **Atenção:** Cookie e Sessões manipulam headers HTTP;
- Por isso eles precisam ser definidos e manipulados antes de você enviar qualquer HTML para o navegador do visitante ou você receberá um erro.
- Outro ponto importante é que, por esse fator, os cookies só estarão disponíveis para leitura (através da variável `$_COOKIE`) após o próximo carregamento de página.

- Cenário comuns para aplicações web:
  - Receber os dados de um formulário e processá-los (busca);
  - Receber dados de um formulário, processá-los e guarda-los em um função (*login*);
  - Remover os dados de uma sessão (*logout*).

- Faça uma programa que calcule o grau de anjo e de safado de uma pessoa pela data de nascimento

Implemente a função `wesley_safadao` que recebe uma data de nascimento (dia, mês e ano) e calcula a porcentagem de anjo e a porcentagem de safado de uma pessoa que nasceu na data de entrada. Os cálculos obedecem a seguinte fórmula:



$$\begin{aligned}\text{safadeza} &= \text{somatorio(mes)} + (\text{ano}/100) * (50 - \text{dia}) \\ \text{anjo} &= 100 - \text{safadeza}\end{aligned}$$

A sua função não retorna nada, apenas escreve na tela a % de safadeza e a % de anjo da pessoa nascida naquela data.

\* A função `somatorio` deve receber um número inteiro `n` e retorna a soma de todos os seus números anteriores. Exemplo: se a entrada for 5, então a função `somatorio` deve retornar  $5+4+3+2+1 = 15$ .