

LÓGICA DE PROGRAMAÇÃO

Professor Jefferson Chaves
jefferson.chaves@ifc-araquari.edu.br



Técnico Integrado em
INFORMÁTICA

- Revisão de Conteúdo:
 - Como a web funciona:
 - Como os dados são trocados entre as páginas e arquivos: o protocolo http;
 - Guardando informações importantes com sessões e cookies;
 - Mantendo seus dados e o padrão JSON;
 - Tarefas comuns em banco de dados;
 - Manipulando formulários;
 - Segurança dos dados;



PROGRAMAÇÃO DE COMPUTADORES

Como a web funciona?

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 avô, 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;



PROGRAMAÇÃO DE COMPUTADORES

ARQUITETURA DA WEB: CLIENTES & SERVIDORES

1

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



2

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



2

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

- 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;

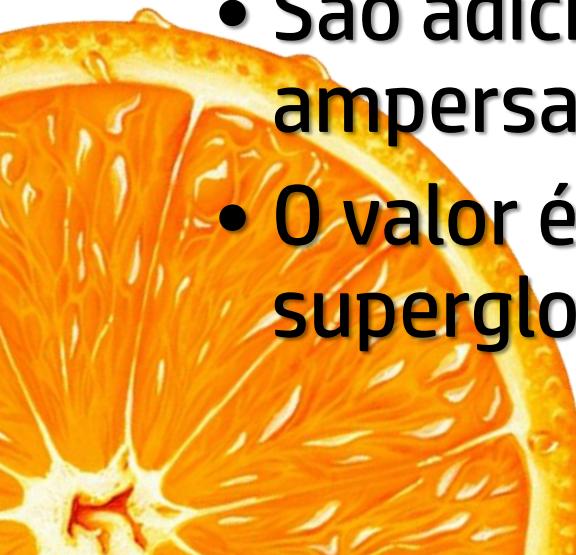
- Quando o navegador solicita uma página web é chamado de **requisição**;
- Quando o servidor web envia a página web solicitada de volta para o navegador é chamado de **resposta**.

- 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

- Esse processo de troca de informações entre páginas utiliza principalmente dois métodos do HTTP: **POST** e **GET**



- Ao usarmos o método **GET**, os dados são enviados por meio da URL;
 - Os dados na url funcionam como variáveis;
 - Começam a partir do sinal de interrogação (?);
 - São adicionados mais valores usando o ampersand ou “e comercial” (&);
 - O valor é recuperado por meio da variável superglobal `$_GET`



- Ao usarmos o método **POST**, os dados são enviados por meio do “envelope” http;
 - Os dados não são visíveis ao usuário;
 - O valor é recuperado por meio da variável superglobal **\$_POST**

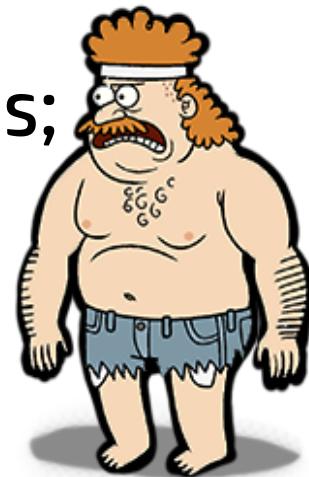


- Devo usar post ou get?
- **Não existe uma maneira definida** de se realizar a comunicação entre arquivos: **isso é uma decisão de arquitetura do sistema;**

- 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**;
- Desenvolva código que pode ser reutilizado;
- Verifique e valide a entrada de dados;
- Pense na organização dos arquivos e pastas;





**PROGRAMAÇÃO DE
COMPUTADORES**

**LEITURA DE
DADOS**

Jefferson de Oliveira Chaves

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

Um formulário web geralmente é parecido com:

<p>formulário: </p>

<form method="get">

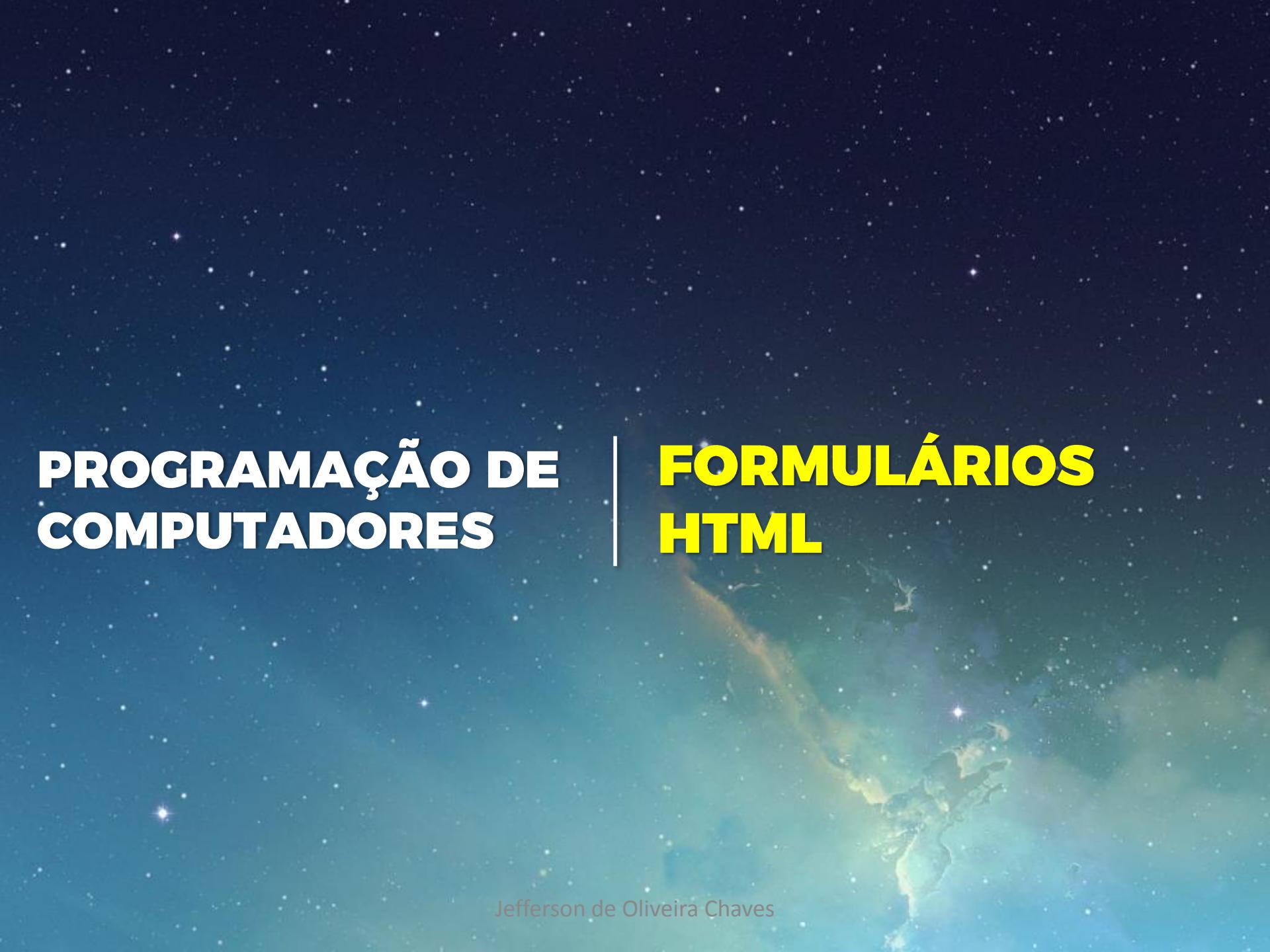
<input type="text" name="nome" required />

<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
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 arquivos 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



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['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 cadastros, 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:
 - $18000 + \text{time}();$
 - $(3600 * 5) + \text{time}();$

- 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*).**



**PROGRAMAÇÃO DE
COMPUTADORES**

**Manipulação de
dados**

Jefferson de Oliveira Chaves

- JSON: JavaScript Object Notation.
- Formato que permite o armazenamento e a **troca de dados** entre aplicações;
- Embora tenha sua origem no JavaScript é suportado por grande parte das linguagens modernas.

- Todas as informações armazenadas em um JSON são chamadas de objetos;
- Objetos devem ser representados por chaves { };
- Todo atributo de um objeto deve seguir o padrão chave:valor;

PROGRAMAÇÃO DE COMPUTADORES | O FORMATO JSON

//Exemplo de arquivo .json

```
{ chave
```

```
“usuario” : “gladistone”
```

```
}
```

```
valor
```

Chaves:
indicam um
objeto

- Palavras e textos devem estar entre aspas duplas;
- Atributos devem ser separados por vírgulas;
- Listas são representadas com o uso dos colchetes [];

//Exemplo de arquivo .json

```
{  
    "usuario" : "aecio", // virgs  
    "senha" : "123456",  
    "amigos" : ["Joesley", "Wesley"]  
}
```

Listas



- O PHP possui dois funções principais para se manipular .json:
 - **json_encode** - transforma um array em formato json;
 - **json_decode** - transforma um json em array;

//converter um array para o formato json

```
$amigos = array("Mary", "Peter", "Sally");
```

```
$meuJSON = json_encode($amigos);
```

```
echo $meuJSON;
```

//leitura de um arquivo .json

```
$dados = file_get_contents("arq.json");
```

//converte os dados json em um array

```
$convertido = json_decode($dados, true);  
print_r($convertido);
```

- o argumento `true` para a função `json_decode` faz com que os dados do arquivo sejam lidos como um array associativo;

//salvando dados em um arquivo .json

```
$amigos = array("Mary", "Peter", "Sally");
```

```
$meuJSON = json_encode($amigos);
```

```
file_put_contents($meuJSON);
```



**PROGRAMAÇÃO DE
COMPUTADORES**

**ESTRUTURA DOS
DADOS**

- Hipoteticamente, vamos ter uma entidade **Funcionários**. Esse entidade teria os seguintes atributos:
 - **Nome;**
 - **Sobrenome;**
 - **Email;**
 - **Data de cadastro;**

- É importante ao se criar uma entidade, criar um campo que sirva como identificador único para esse campo;
- Essa identificação única pode ser criada usando a função uniqid () do próprio PHP;



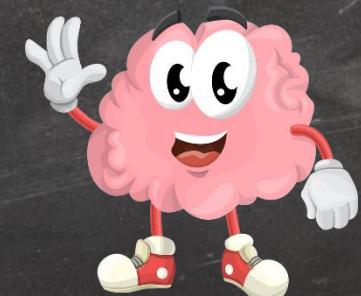
PROGRAMAÇÃO DE COMPUTADORES

AS OPERAÇÕES CRUD

Jefferson de Oliveira Chaves

- **CRUD** é um termo convencionado para designar as quatro operações básicas de manipulação de dados;
- Essa é a conceitualização mais comum:
 - Create - cadastrar
 - Read - leitura/exibição/consulta
 - Update - atualizar
 - Delete - remover

Essas quatro operações são comuns em grande parte das aplicações!





PROGRAMAÇÃO DE COMPUTADORES

CADASTRANDO REGISTROS

Jefferson de Oliveira Chaves

- A maioria das aplicações precisa armazenar dados: usuários, postagens, produtos;
- Guardar os dados em uma variável pode não ser a melhor solução para esses casos, já que são perdidos ao sair da aplicação;
- Nesses casos devemos armazenar tais dados em um arquivo para persisti-lo.
- Persistência em programação está relacionado ao fato de armazenar!



1. Primeiro passo: crie um arquivo `.json` e verifique suas permissões de leitura e escrita;
2. Segundo passo: organizar os dados em um array para serem persistidos;
3. Terceiro passo: converta o array criado para o formato JSON;
4. Quarto passo: grave os dados do arquivo;



//converter um array para o formato json

```
$amigos = array("Mary", "Peter", "Sally");
```

```
$meuJSON = json_encode($amigos);
```

```
//echo $meuJSON;
```

```
//grava os dados no arquivo
```

```
file_put_contents("arquivo.json", $amigos);
```



PROGRAMAÇÃO DE COMPUTADORES | **LEITURA DE REGISTROS**

Jefferson de Oliveira Chaves

- Consulta é uma tarefa frequente na manipulação de dados;
- Existem formas de consultar um ou mais registros;
- Cada consulta variará a depender de sua complexidade. Exemplo:
 - Consultar o registro de um produto;
 - Consultar o registro de um amigo de um amigo meu.



PROGRAMAÇÃO DE COMPUTADORES | **ATUALIZANDO REGISTROS**

Jefferson de Oliveira Chaves

- Para atualizar (e remover) um registro, é importante que cada registro tenha um campo de identificação única;
- Isso para que ao atualizar um registro apenas ele e não todos seja atualizado;