

**CURSOS
TÉCNICOS**

**DESENVOLVIMENTO DE
SISTEMAS WEB II**

Eixo Informática para Internet

UNIDADE 3

SUMÁRIO

1.1.	PHP e HTML.....	3
1.2	WEB Servers	4
1.3	Renderização dinâmica de conteúdo na prática.....	5
1.3.1	Renderizar HTML com base em uma lista PHP.....	5
1.3.2	Definimos a estrutura padrão do HTML para renderizar os elementos da lista.....	5
1.3.3	Renderizar HTML com base em uma condicional PHP.....	6
2.	Utilizando Formulários no PHP.....	7
2.1	Revisando a tag <form>.....	7
2.2	Recuperando a entrada de dados do usuário.....	8
2.3	URL Parameters.....	11
2.4	Redirecionamentos.....	12
2.4.1	Envio de Search Params na URL.....	13
3.	Persistindo dados no cliente (SESSION).....	13
4.	Fixando conhecimento: Desafio.....	15
5.	Referências.....	16

UNIDADE 3

1. PHP WEB

Nesta unidade, vamos nos concentrar na dinâmica do PHP WEB e HTML.

1.1. PHP e HTML

Em uma aplicação web, o HTML é usado para criar a estrutura e o conteúdo básico de uma página da web, enquanto o PHP é usado para adicionar funcionalidades dinâmicas e interativas a essa página. Eles, muitas vezes, trabalham juntos para criar aplicativos web completos e personalizados.

Na unidade 1, vimos um exemplo de como utilizar o PHP embutido no HTML para escrever um texto em tela.

Vamos colocar isso em prática agora!

1. Crie uma pasta do projeto e abra-o no VsCode.
2. Crie um arquivo index.php.
3. Desenvolva o código abaixo, neste mesmo arquivo.

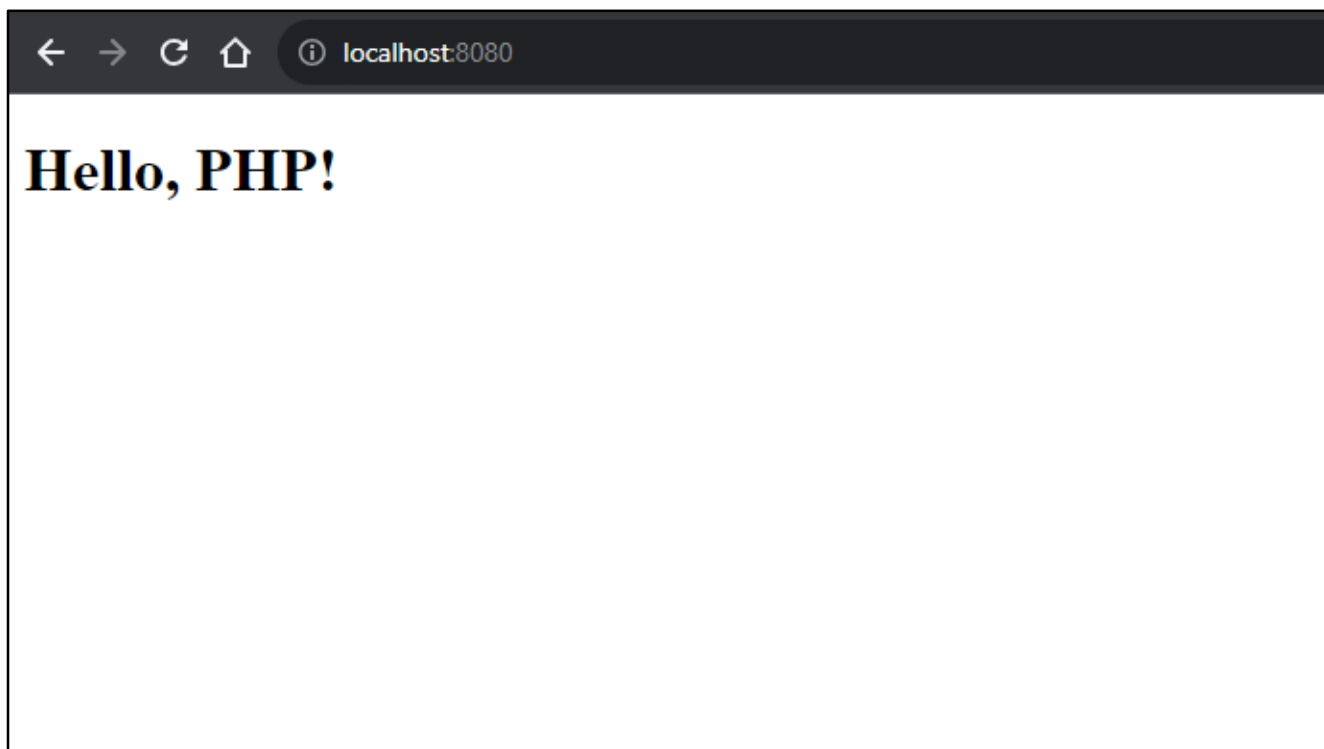
```

1  <?php
2  $texto = "Hello, PHP!";
3  ?>
4
5  <!DOCTYPE html>
6  <html lang="pt-br">
7  <head>
8    <meta charset="UTF-8">
9    <meta name="viewport" content="width=device-width, initial-scale=1.0">
10   <title>Hello PHP</title>
11 </head>
12 <body>
13   <h1><?php echo $texto; ?></h1>
14 </body>
15 </html>
  
```

Perceba que podemos escrever APENAS códigos PHP separadamente, assim como, é possível escrever código PHP dentro do HTML, note que o código PHP escrito dentro da tag **h1** no HTML está envolto das tags de abertura `<?php` e fechamento `?>`.

4. Abra o seu terminal no VSCode e digite o comando `php -S localhost:8080`. Isto inicia um servidor interno PHP para renderizar HTML com PHP no navegador, de acordo com o endereço que foi passado após o **-S**.

5. Acesse no seu navegador a URL: `https://localhost:8080` (que é a mesma que você colocou após o **-S** no comando anterior).



6. Confira o resultado.

No navegador foi renderizado o texto fornecido na variável `$texto`. Tente trocar o valor de `$texto` e pressione **F5** (atualiza a página) no navegador com o servidor ainda rodando para você observar o resultado.

1.2 WEB Servers

Servidores web, também conhecidos como web servers, são programas de software ou hardware que fornecem serviços de hospedagem de páginas da web na internet.

Eles desempenham um papel fundamental na entrega de conteúdo da web aos navegadores dos usuários finais, quando uma solicitação é feita por meio de um URL (Uniform Resource Locator).

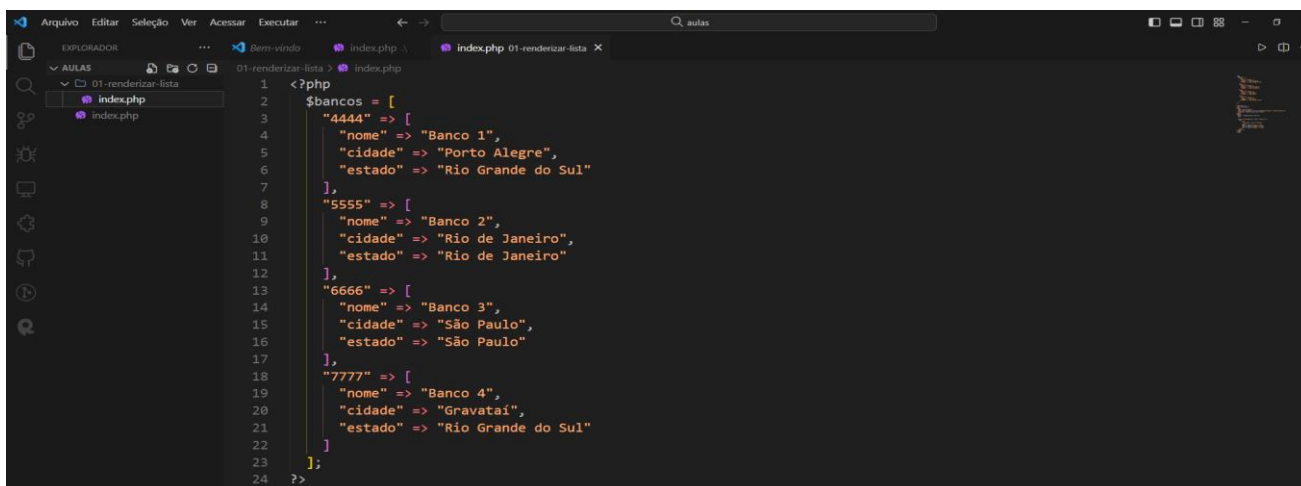
Um exemplo de WEB server foi o que vimos no índice anterior (3.1), onde subimos um **web server local** com o comando `php -S`, para interpretar o nosso código PHP e devolver uma página HTML construída, hospedada localmente em localhost:8080.

1.3 Renderização dinâmica de conteúdo na prática

Vamos criar algumas situações e ver como se comporta na escrita de código PHP + HTML para gerar um conteúdo dinâmico.

1.3.1 Renderizar HTML com base em uma lista PHP.

1. Iremos utilizar a lista de bancos do desafio II.



```

1 <?php
2 $bancos = [
3     "4444" => [
4         "nome" => "Banco 1",
5         "cidade" => "Porto Alegre",
6         "estado" => "Rio Grande do Sul"
7     ],
8     "5555" => [
9         "nome" => "Banco 2",
10        "cidade" => "Rio de Janeiro",
11        "estado" => "Rio de Janeiro"
12    ],
13    "6666" => [
14        "nome" => "Banco 3",
15        "cidade" => "São Paulo",
16        "estado" => "São Paulo"
17    ],
18    "7777" => [
19        "nome" => "Banco 4",
20        "cidade" => "Gravataí",
21        "estado" => "Rio Grande do Sul"
22    ]
23 ];
24 ?>
    
```

1.3.2 Definimos a estrutura padrão do HTML para renderizar os elementos da lista.

```

38 <dl>
39     <dt>
40         <strong>4444</strong>
41     </dt>
42     <dd>Banco 1</dd>
43     <dd>Gravataí</dd>
44     <dd>Rio Grande do Sul</dd>
45 </dl>
    
```

2. Após mapeada a estrutura padrão que irá renderizar as informações desejadas, então montamos o código PHP para deixar esta estrutura dinâmica.

```
<?php foreach($bancos as $ag => $banco) {?> Abertura do foreach
  <dl>
    <dt>
      <strong><?= $ag ?></strong>
    </dt>
    <dd><?= $banco["nome"] ?></dd>
    <dd><?= $banco["cidade"] ?></dd>
    <dd><?= $banco["estado"] ?></dd>
  </dl>
<?php } ?> Fechamento do foreach
```

4. O resultado no navegador será:

```
4444 Banco 1
      Porto Alegre
      Rio Grande do Sul

5555 Banco 2
      Rio de Janeiro
      Rio de Janeiro

6666 Banco 3
      São Paulo
      São Paulo

7777 Banco 4
      Gravataí
      Rio Grande do Sul
```

Perceba que a estrutura HTML foi repetida para cada item na lista `$bancos`.

1.3.3 Renderizar HTML com base em uma condicional PHP.

1. Iremos utilizar novamente lista de bancos do desafio II
2. Definimos a condição verificando se a lista possui itens. Caso não tenha, definimos um aviso “Nenhum banco foi encontrado”.

```
37 <?php
38 unset($bancos); //Retirando os itens da lista o aviso aparecerá.
39 if(isset($bancos)) { ?>
40   <?php foreach($bancos as $ag => $banco) { ?>
41     <dl>
42       <dt>
43         <strong><?= $ag ?></strong>
44       </dt>
45       <dd><?= $banco["nome"] ?></dd>
46       <dd><?= $banco["cidade"] ?></dd>
47       <dd><?= $banco["estado"] ?></dd>
48     </dl>
49   <?php } ?>
50   <?php } else { ?>
51     <strong>Nenhum banco foi encontrado</strong>
52   <?php } ?>
```

Perceba que com o **unset** na linha 38 a lista de bancos fica indefinida, ou seja, a mensagem “nenhum banco foi encontrado” aparecerá em tela, neste caso.

Confira:

02 - Renderização Condicional

Nenhum banco foi encontrado

Se comentarmos a **linha 38** onde o **unset** é declarado, então a lista será renderizada normalmente. Faça o teste! **Não esqueça de atualizar a página com F5.**

2. Utilizando Formulários no PHP.

Formulários desempenham um papel fundamental na interação entre os usuários e aplicativos da web. No PHP, é possível criar, processar e validar formulários de várias maneiras.

2.1 Revisando a tag <form>.

Utilizamos a tag <form> para criar um formulário HTML que receberá entradas de um usuário. A tag de formulário pode possuir um ou mais dos elementos abaixo declarados dentro dela:

<input> | <label> | <textarea> | <button> | <select> | <option> | <fieldset>

Além disso, a tag <form> possui **atributos** que são de extrema importância para o correto funcionamento de HTML + PHP. Por exemplo:

Atributo	Valor	Descrição
action	URL/<caminho do arquivo>	Especifica para onde os dados do formulário devem ser enviados
method	GET ou POST	Especifica qual o método HTTP que deve ser utilizado no envio dos dados do formulário
enctype	application/x-www-form-urlencoded multipart/form-data text/plain	Especifica como os dados do formulário devem ser codificados no envio (disponível apenas no method POST) Pode ser útil no envio de imagens e arquivos

Estes 3 atributos serão os que teremos contato em outros exemplos da apostila, porém existem muitos outros. Para saber mais, acesse a documentação da [W3schools - Tag form](#).

Vamos explorar como recuperar dados da entrada de um usuário com PHP via método **GET** ou **POST**, no próximo item.

2.2 Recuperando a entrada de dados do usuário.

Via GET

1. Criar um formulário com os campos que precisamos receber, declarando o **método que vamos usar (GET)**, o **arquivo que executará a ação para com os dados do form (action)**, e também, **um nome (name) para cada input do formulário**. Segue exemplo:

formulario.html

```

13
14 <form action="/cadastro-banco.php" method="GET" style="display: flex; flex-direction: column; gap: 0.5rem;">
15   <label for="agencia">Agência</label>
16   <input type="number" id="agencia" name="agencia" placeholder="Agência" required>
17
18   <label for="nome">Nome banco</label>
19   <input id="nome" name="nome" placeholder="Nome Banco" required>
20
21   <label for="cidade">Cidade</label>
22   <input id="cidade" name="cidade" placeholder="Cidade" required>
23
24   <label for="estado">Estado</label>
25   <input id="estado" name="estado" placeholder="UF" required>
26
27   <input name="cadastrado" type="submit" value="Cadastrar">
28 </form>
  
```

É possível observar que, precisaremos de um arquivo .php para lidar com os dados do formulário. A estrutura do projeto será da seguinte forma:

sua-pasta-com-formularios/

| - - formulario.html

| - - cadastro-banco.php

cadastro-banco.php

```

v AULAS 03-formularios > cadastro-banco.php
> 01-renderizar-lista
> 02-renderizacao-condicio...
v 03-formularios
  cadastro-banco.php
  formulario.html
1 <?php
2   var_dump($_GET);
3 ?>
4
5
  
```

No arquivo, **cadastro-banco.php** no momento, vamos apenas observar o comportamento da variável superglobal **\$_GET**, no envio do formulário.

Superglobais são variáveis nativas do PHP que estão disponíveis em todo o escopo do projeto. Saiba mais em: [Doc. Oficial PHP - Superglobais](#).

Preenchimento dos dados do formulário e envio:

Cadastro de dados nos campos

var_dump no arquivo cadastro-banco.php

Após preenchidos os campos no formulário e realizado clique em **cadastrear**, então o arquivo **cadastro-banco.php** é requisitado e os códigos contidos neste arquivo executam. Neste caso, **var_dump()** mostrou em tela todos os itens contidos no superglobal **\$_GET**, com base no atributo **name** definido no arquivo HTML do formulário.

Confira o vídeo do processo completo:

https://www.youtube.com/watch?v=224Yx_mz6jg

Agora que sabemos que os dados do formulário ficam dentro do superglobal **\$_GET** para acessar estes dados podemos utilizar o seguinte código:

```

03-formularios > cadastro-banco.php
1  <?php                                Verifica se o form foi enviado através do name="cadastrado" no botão de submit
2      if (isset($_GET["cadastrado"])) {
3          echo "Agência: {$_GET["agencia"]}" . "<br/>";
4          echo "Nome: {$_GET["nome"]}" . "<br/>";
5          echo "Cidade: {$_GET["cidade"]}" . "<br/>";
6          echo "Estado: {$_GET["estado"]}" . "<br/>";
7      }
8  ?>
9
    
```

Basicamente, acessamos os campos disponíveis dentro de **\$_GET[name]** e mostramos em tela. Confira o resultado:

< > ↻ 🏠 ⓘ localhost:8080/03-formularios/cadastro-banco.php?agencia=01010&nome=Banco+6&cidade=Gravataí&estado=Rio+Grande+do+Sul&cadastrado=Cadastrar

Agência: 01010
 Nome: Banco 6
 Cidade: Gravataí
 Estado: Rio Grande do Sul

Via POST

O **método POST** é frequentemente usado para enviar informações confidenciais ou dados que podem ser longos, como envio de formulários de login, envio de dados de formulários de registro e etc.

A grande vantagem deste método, com relação ao GET é que: os dados do usuário não ficam expostos na URL do navegador. Assim, garantimos que ninguém enxergue a senha de um usuário, enquanto ele preenche o formulário de login, por exemplo.

formulario.html

```

<!-- Usamos o GET ou POST no método do form --> usando POST agora
<form action="/cadastro-banco.php" method="POST" style="display: flex; flex-direction:
  <label for="agencia">Agência</label>
  <input type="number" id="agencia" name="agencia" placeholder="Agência" required>

  <label for="nome">Nome banco</label>
  <input id="nome" name="nome" placeholder="Nome Banco" required>

  <label for="cidade">Cidade</label>
  <input id="cidade" name="cidade" placeholder="Cidade" required>

  <label for="estado">Estado</label>
  <input id="estado" name="estado" placeholder="UF" required>

  <input name="cadastrado" type="submit" value="Cadastrar">
</form>
  
```

cadastro-banco.php

```

2 //VIA POST
3 if (isset($_POST["cadastrado"])) {
4   echo "Agência: {$_POST["agencia"]}" . "<br/>";
5   echo "Nome: {$_POST["nome"]}" . "<br/>";
6   echo "Cidade: {$_POST["cidade"]}" . "<br/>";
7   echo "Estado: {$_POST["estado"]}" . "<br/>";
8 }
  
```

Desta forma, obtemos o mesmo resultado que usando GET, porém agora as informações do usuário não são mais expostas na URL. Normalmente, **utilizamos o GET mais para pesquisas e redirecionamentos**, enquanto o **POST para envio de dados de usuário**.

Caso queira conferir os códigos desenvolvidos até o momento:

→ Repositório: <https://github.com/RafaelR4mos/php-exemplos-I>

2.3 URL Parameters

Ao submeter um formulário com o **method=GET**, então os dados dos campos do formulário aparecem na URL do navegador com um formato bastante característico. Vamos analisar:

Voltando ao exemplo do **formulario.html** de cadastro de bancos, ao enviar temos a seguinte URL:

<http://localhost:8080/03-formularios/cadastro-banco.php?agencia=01010&nome=Banco+6&cidade=Gravata%C3%AD&estado=Rio+Grande+do+Sul&cadastrado=Cadastrar>

Podemos dividir essa URL em **2 partes**:

1. Caminho até arquivo e domínio/servidor: → <http://localhost:8080/03-formularios/cadastro-banco.php>

2. URL Param /parâmetros de URL:

→ [?agencia=01010&nome=Banco+6&cidade=Gravata%C3%AD&estado=Rio+Grande+do+Sul&cadastrado=Cadastrar](http://localhost:8080/03-formularios/cadastro-banco.php?agencia=01010&nome=Banco+6&cidade=Gravata%C3%AD&estado=Rio+Grande+do+Sul&cadastrado=Cadastrar)

Dentro dos *URL Params* podemos destacar alguns caracteres reservados de URL.

?agencia=01010&nome=Banco+6&cidade=Gravata%C3%AD&estado=Rio+Grande+do+Sul&cadastrado=Cadastrar

? → Indica o início dos parâmetros de URL.

& → Indica a separação de uma chave-valor

= → Indica a atribuição de um valor a uma chave

Caracteres como: **“+, %C3%AD”** → São caracteres utilizados para tratar strings onde:

+ → Representa um espaço em branco

%<valor ASCII> → Representa um caractere, de acordo com a tabela ASCII.

Saiba mais sobre a tabela ASCII

Estes caracteres de tratamento de string evitam que ocorram conflitos com caracteres reservados de URL como: “?”, “&”, “=”, “/”...

2.4 Redirecionamentos

Podemos utilizar de redirecionamentos ao submeter um formulário. Ao chamar um arquivo .php via outro arquivo de formulário podemos executar determinada lógica, e após isso, redirecionar o usuário para a página que ele estava.

Esse resultado é obtido através do **header** em PHP. O **header** literalmente envia um cabeçalho HTTP. Neste cabeçalho, podemos enviar uma **location**, para o arquivo anterior.

Utilizando deste código base:

→ Repositório: <https://github.com/RafaelR4mos/php-exemplos-I>

Ao acessar **03-formularios > cadastro-banco.php** podemos redirecionar o usuário novamente para o formulário caso tudo tenha ocorrido como esperado. Basta adicionar o header, neste arquivo.

```
//VIA POST
if (isset($_POST["cadastrado"])) {
    echo "Agência: {$_POST["agencia"]}" . "<br/>";
    echo "Nome: {$_POST["nome"]}" . "<br/>";
    echo "Cidade: {$_POST["cidade"]}" . "<br/>";
    echo "Estado: {$_POST["estado"]}" . "<br/>";

    header("Location: formulario.html");
}
```

Manda o usuário de volta para o formulário após executada a lógica neste arquivo

2.4.1 Envio de Search Params na URL

É possível enviar um search param no submit de um formulário. Basta declararmos no atributo action.

Exemplo:

```
<form action="cadastro-banco.php?id_banco=1" method="POST">
```

...

```
</form>
```

assim a URL vai ficar: **localhost:8080/cadastro-banco.php?id_banco=1**

Para recuperar o valor deste ID podemos usar a superglobal **\$_GET[id_banco]**, pois como este resultado fica na URL do navegador só podemos acessar com o **GET**.

3. Persistindo dados no cliente (SESSION)

Uma sessão no PHP é uma maneira de persistir (manter) dados de usuário entre diferentes requisições HTTP. **\$_SESSION** permite que você armazene informações específicas de um usuário, como variáveis de sessão, para que essas informações possam ser acessadas em várias páginas web durante a visita do usuário ao site.

As sessões são amplamente utilizadas para **manter o estado** e a **autenticação do usuário** em aplicações da web. É importante ressaltar que **\$_SESSION**, assim como o **\$_GET** e **\$_POST**, **possui escopo global**, pois é uma superglobal do PHP.

Em resumo, as vantagens são:

- **Persistência de dados:** as sessões permitem que você armazene dados do usuário de forma persistente durante a visita do usuário, para que esses dados possam ser usados em várias páginas.
- **Autenticação:** você pode usar sessões para rastrear se um usuário está autenticado, o que é útil para proteger áreas restritas ou protegidas do seu site.
- **Segurança:** as informações da sessão são armazenadas no servidor, o que as torna mais seguras do que armazenar informações sensíveis no lado do cliente (como cookies).

Para utilizar `$_SESSION` deve-se:

1. **Iniciar uma sessão:** para começar a usar sessões em PHP, você precisa chamar `session_start()` no início de cada script onde deseja usar sessões.

Geralmente, isso é feito na parte superior do arquivo.

```

<?php
session_start();
?>
    
```

2. **Armazenar dados na sessão:** você pode armazenar dados na sessão usando a variável superglobal `$_SESSION`. Por exemplo:

```

<?php
$_SESSION['nome'] = 'João';
$_SESSION['email'] = 'joao@example.com';
?>
    
```

Perceba que chamamos a superglobal `$_SESSION["nomeDaChave"]`. A chave que passamos será a mesma que iremos utilizar para atualizar ou consumir a informação armazenada na chave. Confira o exemplo:

```

1  <?php
2  session_start();
3
4  $_SESSION['nome'] = 'João';
5  $_SESSION['email'] = 'joao@example.com';
6
7  echo $_SESSION["nome"]; //João
8  ?>
    
```

A informação que aparecerá em tela no **echo da linha 7** será **“João”**. E esta informação estará disponível em **QUALQUER** arquivo do seu projeto que tenha uma sessão iniciada. Basta acessar `$_SESSION["nome"]`.

4. Fixando conhecimento: Desafio

Construa uma página WEB com PHP, HTML e estilização básica, que contenha um formulário e renderize informações dinamicamente de uma lista com a possibilidade de inserir novas informações, editar e deletar. É ideal que estas informações persistam no estado da aplicação, para isso é recomendado utilizar `$_SESSION`.

Em resumo deve conter:

- Página que renderiza informações de uma lista;
- Página com formulário para editar/criar uma nova informação;
 - Pode ser duas páginas também: Uma para **CRIAR**, outra para **EDITAR**.
- Página para exibir informações detalhadas de um item. Nesta mesma página pode ter a opção de deletar o item em específico.

Você pode aproveitar este desafio para adiantar a produção do seu projeto, caso escolha um tema que tenha relação com o da semana passada.

Um exemplo de projeto pode ser encontrado no repositório abaixo:

→ <https://github.com/RafaelR4mos/php-exemplos-I>

Passo-a-passo:

1. Você pode clonar ou baixar os arquivos do repositório.
2. Logo após, você deve utilizar o servidor integrado do PHP com o comando `php -S localhost:port`. Exemplo: `php -S localhost:8080`
3. Logo após, entrar em seu navegador com a URL: **localhost:8080/04-projeto-exemplo/index.php**

5. Referências

Múltiplos autores: PHP: **Manual do PHP**, 2023. Disponível em:

<https://www.php.net/manual/pt_BR/>. Acesso em: 10 de outubro de 2023.

Múltiplos autores: PHP: **Superglobais**, 2023. Disponível em:

<https://www.php.net/manual/pt_BR/language.variables.superglobals.php>. Acesso em: 11 de outubro de 2023.

Múltiplos autores: **HTML <form> Tag**, 2023. Disponível em:

<https://www.w3schools.com/tags/tag_form.asp>. Acesso em 11 de outubro de 2023.