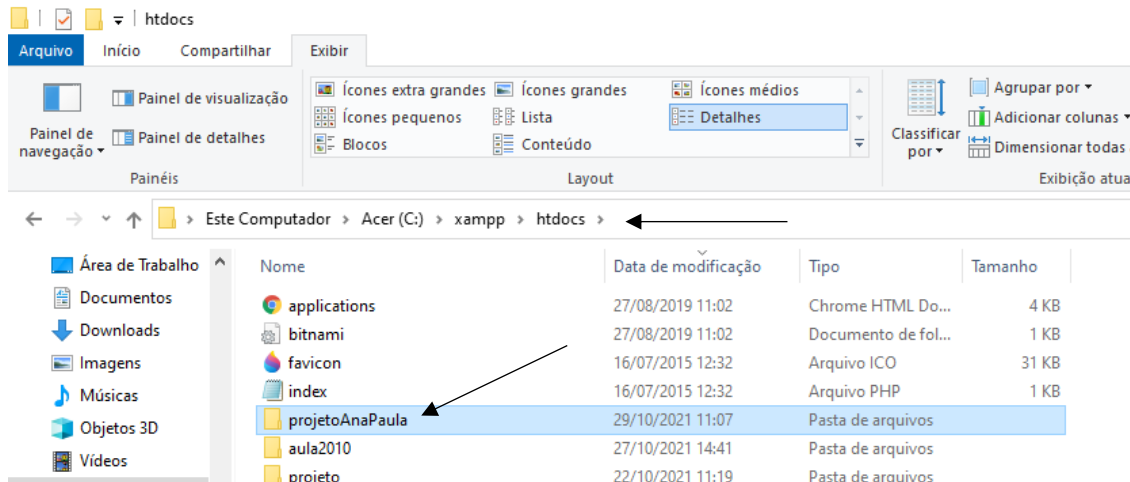


Projeto Biblioteca - PW

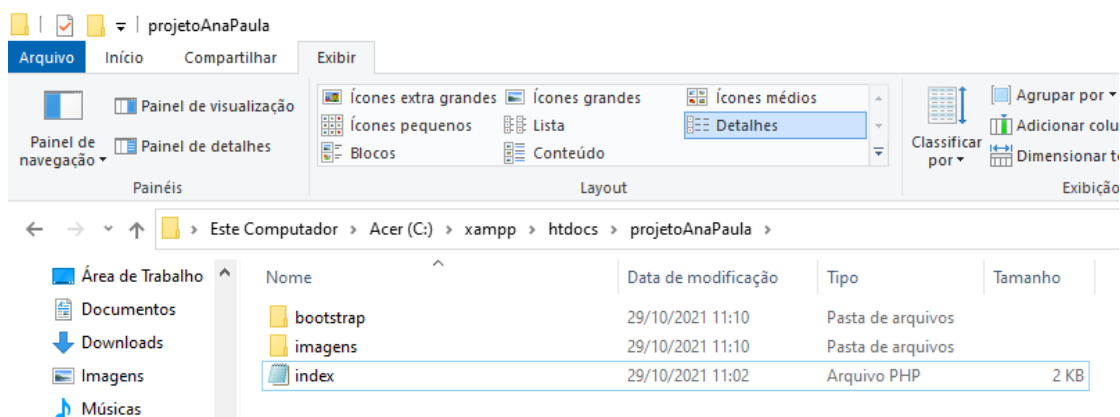
PHP com PDO e Maria DB

Para iniciar esse projeto, você precisa criar uma pasta com o seu nome ou nome da dupla em C:\xampp\htdocs



Na pasta criada, você precisa adicionar os seguintes arquivos:

- Pasta do Bootstrap
- Pasta com as Imagens
- Arquivos HTML



Agora vamos programar: Esse projeto consiste no desenvolvimento de um sistema simples de cadastro de livros de uma biblioteca com PHP e PDO, implementando o uso do framework Bootstrap e armazenamento das informações no SQL Maria DB.

Página index.php

Nessa página teremos um menu desenvolvido no Bootstrap, utilizaremos ele em todas as demais páginas para que o usuário possa navegar entre as opções:

```

C:\xampp\htdocs\projeto\index.php - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index.php  conexao.php  cadastrar.php  sistema.php

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>LivrariaCultura</title>
6   <link rel="stylesheet" type="text/css" href="bootstrap/css/
  bootstrap.css">
7 </head>
8 <body>
9   <div class="container">
10    <div class="row">
11      <div class="col-sm-12">
12        
13      </div>
14    </div>
15
16    <hr>
17    <div class="row">
18      <div class="col-sm-12">
19        <nav class="navbar navbar-expand-lg navbar-light bg-light">
20          <div class="container-fluid">
21
22            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
23              <li class="nav-item">
24                <a class="nav-link active" aria-current="page"
25                  href="index.php">Home</a>
26              <li class="nav-item">
27                <a class="nav-link" href="cadastro.php">Cadastrar
28              </a>
29              <li class="nav-item">
30                <a class="nav-link" href="Consultar.php">Consultar
31              </a>
32            </li>
33          </ul>
34
35          <form class="d-flex">
36            <input class="form-control me-2" type="search" placeholder="
37              Pesquisar" aria-label="Search">
38            <button class="btn btn-outline-success" type="submit">
39              Pesquisar</button>
40          </form>
41        </div>
42      </div>
43    </nav>
44    <div>
45      <?php
46        if(isset($_GET["msg"])){
47          echo "<h3><i>".$_GET["msg"]."</i></h3>";
48        }
49      <?>
50    </div>
51  </div>
52 </body>
53 </html>

```

Explicando os códigos acima:

Utilizamos um container para o desenvolvimento dessa página. Na primeira row, entre as linhas 10 e 14 inserimos uma imagem com a imagem de topo

Entre as linhas 17 e 41 utilizamos comandos do framework Bootstrap para criar o menu de navegação para o usuário (estudaremos mais detalhadamente esses comandos na aula de DDW)

Entre as linhas 42 e 46 inserimos um comando em PHP, esse comando verifica se a variável \$msg possui algum valor, caso ela possua algo armazenado nela, nesse espaço da página o conteúdo da variável será exibido, caso contrário o PHP deixa esse espaço sem conteúdo e executa os demais conteúdos:

isset — Informa se a variável foi iniciada

Página cadastrar.php

Nessa página vamos criar um pequeno formulário para inserir as informações dos livros que precisamos armazenar no Banco de Dados.

```
C:\xampp\htdocs\projeto\cadastrar.php - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>Cadastrar</title>
6 </head>
7 <body>
8     <?php
9         include_once "index.php";
10    ?>
11    <div class="container">
12        <div class="row">
13            <div class="col">
14                <form action="sistema.php" method="post">
15                    <h1>Cadastro de Livros</h1>
16
17                    <input type="hidden" name="tipo" value="
18                        incluir">
19                <div class="mb-3">
20                    <label for="titulo" class="form-label">Título</label>
21                    <input type="text" class="form-control" id="autor"
22                        name="titulo">
23                <div class="mb-3">
24                    <label for="autor" class="form-label">Autor</label>
25                    <input type="text" class="form-control" id="autor"
26                        name="autor">
27                </div>
28                <div class="mb-3">
29                    <label for="pagina" class="form-label">Páginas</label>
30                    <input type="number" class="form-control" id="
31                        paginas" name="paginas" min="1">
32                </div>
33                <input type="submit" class="btn btn-primary" value="
34                    Cadastrar">
35            </div>
36        </div>
37    </body>
38 </html>
```

Explicando os códigos acima:

Os programas que fizemos até agora, possuíam apenas dois arquivos de código. Trabalhamos dessa maneira para simplificar o estudo e maximizar a visualização do código que estávamos implementando. Porém, o comum, é que uma aplicação escrita em PHP possua vários, senão, centenas de arquivos com código PHP.

Dessa forma, temos de ser capazes de dizer que um arquivo em específico deve ser aberto antes que a execução do arquivo que estamos desenvolvendo seja interpretado. Por exemplo, entre as linhas 8 e 10 com o comando `include_once` podemos inserir o menu feito na página `index.php`, sem precisar copiar e colar o código diversas vezes.

O PHP disponibiliza 4 funções para importação de arquivos e cada uma desempenha uma tarefa específica, logo, temos que conhecer todas as maneiras disponíveis de importação:

`include()`: inclui o arquivo passado como parâmetro. Se o arquivo não for encontrado, o PHP irá lançar um "warning", mas dará continuidade na execução.

`include_once()`: o funcionamento dessa função é igual ao da função `include()`, porém, o arquivo só será importado caso o mesmo ainda não tenha sido.

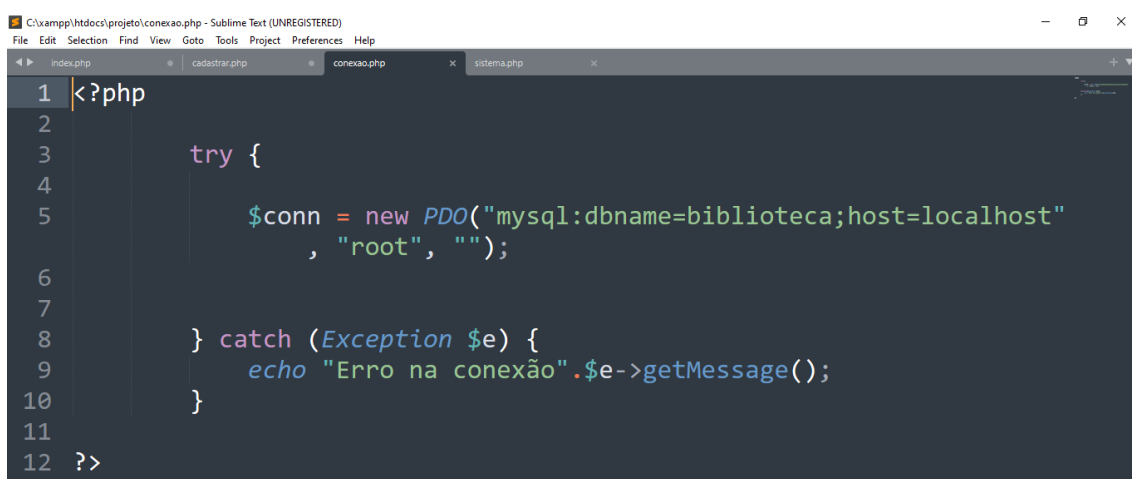
`require()`: a função `require()` importa arquivos, porém, caso o mesmo não seja encontrado, será levantado uma exceção e a execução é finalizada. Essa é uma maneira de interrompermos a execução dos scripts caso alguma anomalia ocorra.

`require_once()`: o funcionamento é igual ao da função `require()`, porém, um arquivo só será importado caso o mesmo ainda não tenha sido.

Linha 17: nessa linha colocamos um `input type="hidden"`, esse tipo de input serve para entrada de um campo oculto. Como vocês sabem em um sistema podemos desenvolver várias ações, como inserir, atualizar, excluir e consultar, então nesse projeto utilizamos o campo oculto para que o sistema possa identificar qual ação está sendo realizada nesse momento.

Página `conexao.php`

Essa página realizará a conexão entre as páginas em PHP e o Banco de Dados.



```
1 <?php
2
3     try {
4
5         $conn = new PDO("mysql:dbname=biblioteca;host=localhost"
6             , "root", "");
7
8     } catch (Exception $e) {
9         echo "Erro na conexão". $e->getMessage();
10    }
11
12    ?>
```

Explicando os códigos acima:

A linha 5 já estudamos na aula anterior, é responsável por armazenar na variável `$conn` os dados de conexão com o Banco de Dados Biblioteca. Não se esqueça da última aspas, no exemplo acima elas estão sem conteúdo porque o meu SGBD não possui senha de acesso, na escola a maioria dos computadores tem a senha "minas", então é necessário inserir essa informação:

```
$conn = new PDO("mysql:dbname=biblioteca;host=localhost", "root", "minas");
```

Bloco try/catch: serve para tratamento de exceções, tratamento de códigos que podem não ser totalmente atendidos e gerarem alguma exceção/erro. O try consegue recuperar erros que possam ocorrer no código fornecido em seu bloco. O catch por sua vez faz o tratamento dos erros que aconteceram. Nesse nosso projeto o try verifica se a conexão é realizada, caso a conexão não aconteça o catch exibe uma mensagem de erro.

Página sistema.php

A página sistema.php é responsável por gerenciar as ações do projeto que está sendo desenvolvido.



```
1 <!DOCTYPE html>
2 <html lang="pt-br">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,
6       initial-scale=1.0">
7     <meta http-equiv="X-UA-Compatible" content="ie=edge">
8     <title>sistema</title>
9 </head>
10 <body>
11     <?php
12         include_once "conexao.php";
13
14         $titulo = $_POST['titulo'];
15         $autor = $_POST['autor'];
16         $paginas = $_POST['paginas'];
17
18         $op = $_POST['tipo'];
19
20         switch ($op) {
21             case 'incluir':
22                 $stmt=$conn->prepare("insert into livro(titulo, autor,
23                   paginas) values(:TITULO, :AUTOR, :PAGINAS)");
24
25                 $stmt->bindParam(":TITULO", $titulo);
26                 $stmt->bindParam(":AUTOR", $autor);
27                 $stmt->bindParam(":PAGINAS", $paginas);
28
29                 $msg = $stmt->execute()?"Livro Cadastrado com sucesso!":
30                   "Erro ao alterar!";
31
32                 header("location: index.php?msg=".urlencode($msg));
33                 break;
34             default:
35                 echo "Opção não encontrada";
36                 break;
37         }
38     ?>
39 </body>
40 </html>
```

Explicando os códigos acima:

Na linha 20 pegamos a variável da conexão e acessamos um método prepare que serve para preparar um comando que você quer enviar para o banco de dados, então você escreve dentro das aspas o comando SQL que você quer que seja executado no seu banco de dados

```
$stmt=$conn->prepare("INSERT INTO teste VALUES()")
```

Agora usando a variável statement(interface utilizada para executar instruções SQL, que significa comando).

Até agora você sabia como passar os valores colocando os mesmos dentro do parenteses depois do values, mas o PDO consegue detectar onde existem parâmetros e ele coloca até as aspas quando é um texto e deixa sem aspas quando é um número, então para passar esse parâmetro você vai colocar dois pontos e fazer como se fosse um id(identificador) daquele parâmetro em caixa alta

```
$stmt=$conn->prepare("insert into livro(titulo, autor, paginas) values(:TITULO, :AUTOR, :PAGINAS)");
```

Pelo stmt você pode enviar qualquer comando, não apenas comandos básicos, e depois do ponto e vírgula poderiam ser enviados mais comandos sql para serem executados juntos

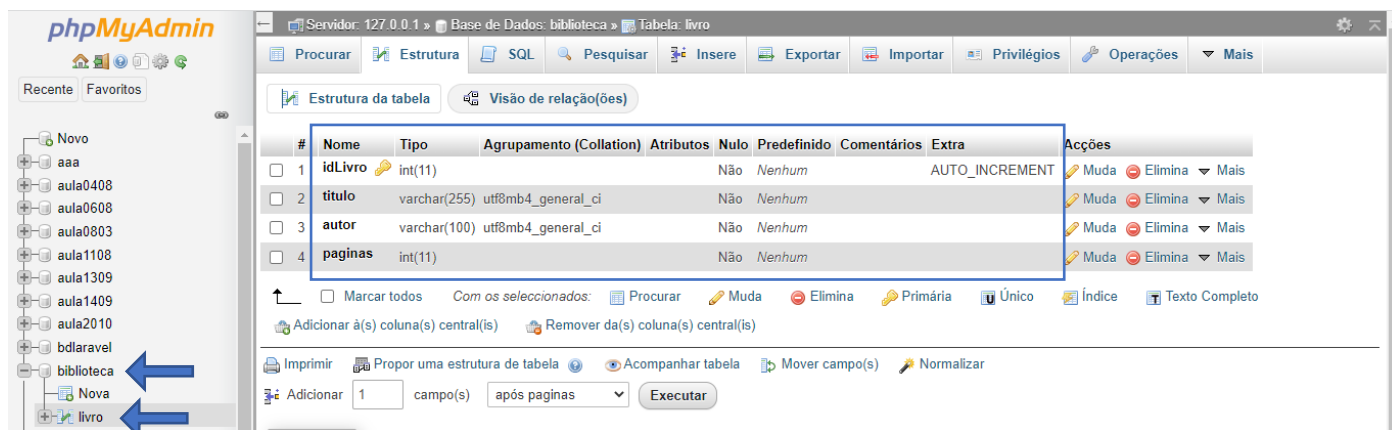
Nas linhas 22 a 24 utilizamos o bind para unir e conectar esses parâmetros com o nosso query com nosso comando sql. Para isso vamos usar o método bindParam para cada parâmetro que nós temos, onde o bind vai ligar o parâmetro ao valor, por exemplo :TITULO ao \$titulo

Depois de tudo isso, na linha 26 você ainda não executou o comando, apenas preparou e para executar usamos o método execute()

Lembra do espaço que separamos na página index para receber possíveis mensagens? A variável \$msg da linha 26 vai armazenar a mensagem “Livro Cadastrado com Sucesso” caso a execução dos comandos aconteça.

Banco de Dados

Antes de testar suas páginas acima é necessário que você crie um Banco de Dados que seja o responsável pelo armazenamento e consultas às informações que forem cadastradas nas páginas. Para isso, como já fizemos nas aulas anteriores, acesse o PHPMyAdmin, crie o BD biblioteca e a tabela livro com os campos:



Realize um teste: Abra as páginas PHP criadas, cadastre um livro e verifique se os dados foram salvos no Banco de Dados.

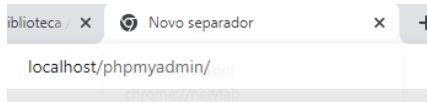
Projeto Biblioteca – Parte II

Criação do Banco de Dados Biblioteca e Exibição dos Dados Cadastrados na aba Consulta

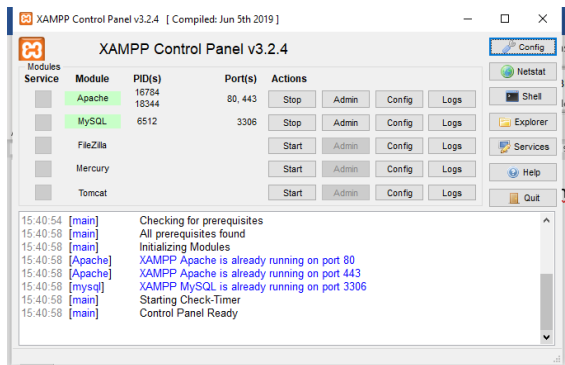
Para aqueles que não conseguiram criar o Banco de Dados Biblioteca, segue abaixo o passo a passo. Se você já conseguiu, pode ir direto para a criação do arquivo `exibe.php` (página 10)

Passo 1 – Acesse o PHPMYAdmin

Abra seu navegador web e digite: `localhost/phpmyadmin/`



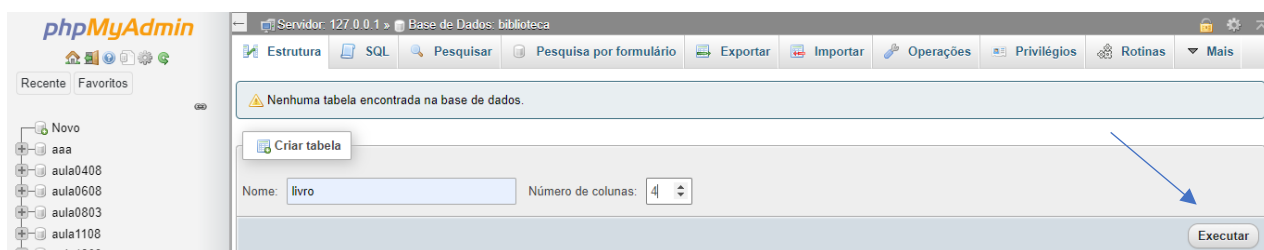
Caso dê algum erro, tente startar o Apache e o MySQL no painel de controle do XAMPP



Após acessar o PHPMYAdmin, clique em Novo, escolha o nome do seu Banco de Dados, nesse caso o nosso BD chama biblioteca e por fim clique no botão CRIAR:



Em seguida, você será direcionado para a página onde vai criar a tabela LIVRO



O número de colunas é 4, pois teremos que armazenar o título do livro, o autor, quantidade de páginas e uma chave primária para identificar os registros. Não esqueça de clicar no botão executar.

Nome da Tabela: livro Adicionar 1 coluna(s) Executar

Nome	Tipo	Tamanho/Valores	Predefinido	Agrupamento (Collation)	Atributos	Nulo	Índice	A_I
idLivro	INT		Nenhum			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>
titulo	VARCHAR	255	Nenhum			<input type="checkbox"/>		<input type="checkbox"/>
autor	VARCHAR	100	Nenhum			<input type="checkbox"/>		<input type="checkbox"/>
paginas	INT		Nenhum			<input type="checkbox"/>		<input type="checkbox"/>

Estrutura

Na tela acima você deve colocar os campos da tabela e seus respectivos tipos de dados. Não esqueça de colocar o idLivro como chave primária e selecionar a opção A_I (auto_increment).

Role até o final da página e clique no botão Guardar

autor VARCHAR 100 Nenhum

paginas INT Nenhum

Estrutura

Comentários da tabela: Colação: Motor de armazenamento: InnoDB

Definição da PARTIÇÃO:

Partição por: (Expressão ou lista de colun)

Partições:

Pré-visualizar SQL Guarda

Agora seu BD e sua tabela estão aptas a receberem e armazenarem as informações que forem digitadas na página cadastro.php

phpMyAdmin

Recente Favoritos

Novo

aaa

aula0408

aula0608

aula0803

aula1108

aula1309

aula1409

aula2010

bdlaravel

biblioteca

Nova

livro

Servidor: 127.0.0.1 » Base de Dados: biblioteca » Tabela: livro

Procurar Estrutura SQL Pesquisar Inserir Exportar Importar Privilegios Operações Rastreamento Acionadores

Estrutura da tabela Visão de relações

#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Comentários	Extra	Acções
1	idLivro	int(11)			Não	Nenhum		AUTO_INCREMENT	Muda Elimina Mais
2	titulo	varchar(255)	utf8mb4_general_ci		Não	Nenhum			Muda Elimina Mais
3	autor	varchar(100)	utf8mb4_general_ci		Não	Nenhum			Muda Elimina Mais
4	paginas	int(11)			Não	Nenhum			Muda Elimina Mais

Marcar todos Com os seleccionados: Procurar Muda Elimina Primária Único Índice Texto Completo Adicionar a(s) coluna(s) central(is)

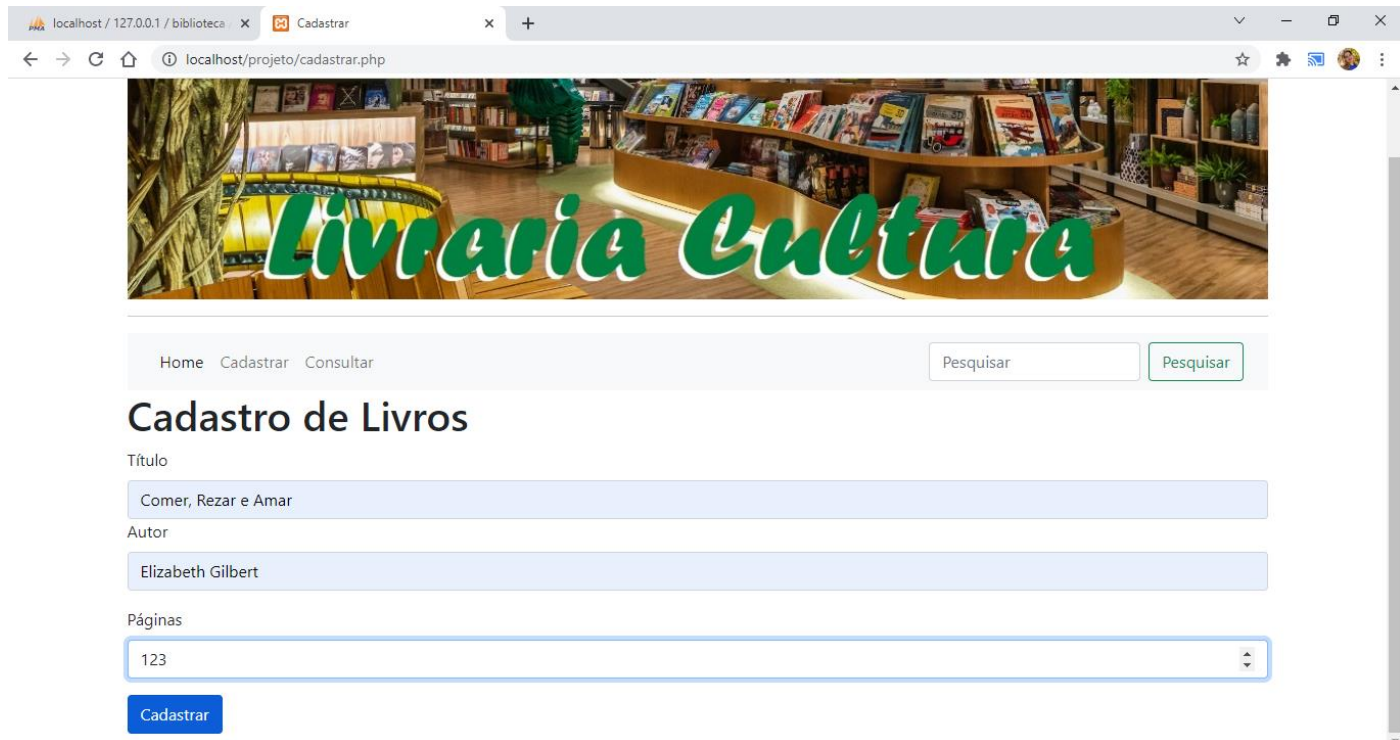
Remover da(s) coluna(s) central(is)

Imprimir Propor uma estrutura de tabela Acompanhar tabela Mover campo(s) Normalizar

Adicionar 1 campo(s) após paginas Executar

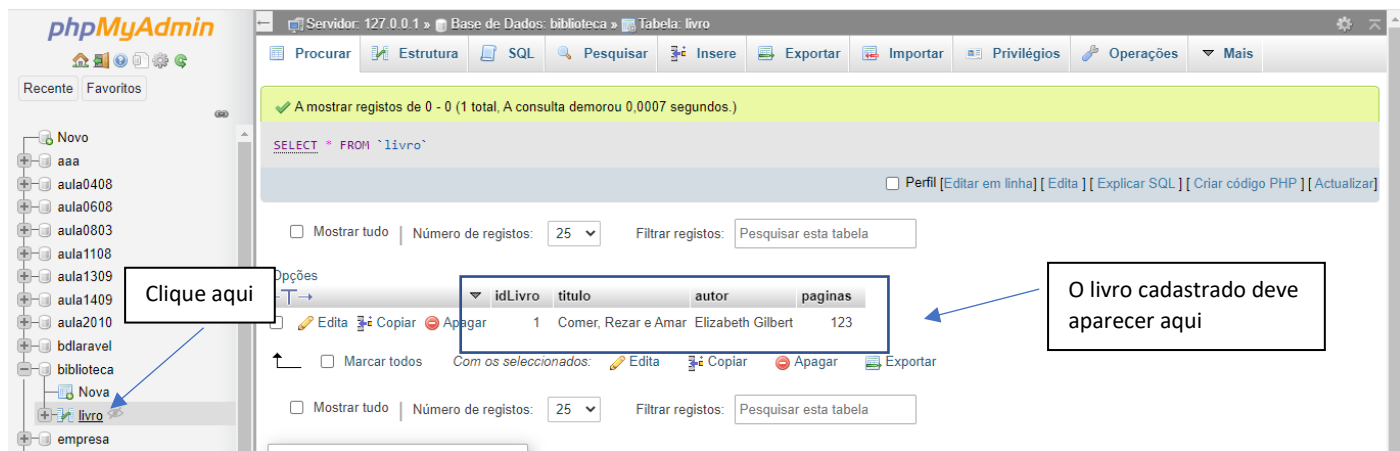
Faça o teste:

1 – Cadastre um livro:



The screenshot shows a web browser window with the address bar displaying 'localhost / 127.0.0.1 / biblioteca' and 'Cadastrar'. The page title is 'Livraria Cultura'. Below the header, there are navigation links: 'Home', 'Cadastrar', and 'Consultar'. There are also search buttons labeled 'Pesquisar'. The main content area is titled 'Cadastro de Livros'. It contains three input fields: 'Título' with the value 'Comer, Rezar e Amar', 'Autor' with the value 'Elizabeth Gilbert', and 'Páginas' with the value '123'. A blue 'Cadastrar' button is at the bottom of the form.

2 – No PHPMYAdmin clique na tabela livro e o livro cadastrado deve aparecer no BD



The screenshot shows the phpMyAdmin interface. On the left, there is a sidebar with a tree view of databases and tables. The 'biblioteca' database is selected, and the 'livro' table is highlighted. A blue arrow points to the 'livro' table with the text 'Clique aqui'. The main area shows the 'Tabela: livro' view. It displays a table with the following data:

idLivro	titulo	autor	paginas
1	Comer, Rezar e Amar	Elizabeth Gilbert	123

A blue arrow points to the table with the text 'O livro cadastrado deve aparecer aqui'. The interface also shows a SQL query: 'SELECT * FROM `livro`' and a status bar indicating 'A mostrar registros de 0 - 0 (1 total, A consulta demorou 0,0007 segundos.)'.

CONTINUAÇÃO...

Agora vamos preparar nosso projeto para exibir ao usuário os livros que estiverem cadastrados no Banco de Dados. Para isso, vamos criar uma nova página chamada `exibe.php`:

exibe.php

```
C:\xampp\htdocs\projeto\exibe.php - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index.php x cadastran.php conexao.php sistema.php exibe.php consultar.php x

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title></title>
6     <link rel="stylesheet" type="text/css" href="bootstrap/css/
    bootstrap.css">
7 </head>
8 <body>
9 <div class="container">
10     <?php
11         include_once "conexao.php";
12
13         try {
14             $consulta = $conn->query("SELECT * FROM Livro");
15
16             echo "<h1>Livros Cadastrados</h1><br>";
17             echo "<table class='table table-striped'>
18                 <tr style='font-weight: bold;'>
19                     <td>Titulo</td>
20                     <td>Autor</td>
21                     <td>Páginas</td>
22                     <td>Ações</td>
23                 </tr>";
24
25             while($linha = $consulta->fetch(PDO::FETCH_ASSOC)){
26                 echo "<tr>
27                     <td>$linha[titulo]</td>
28                     <td>$linha[autor]</td>
29                     <td>$linha[paginas]</td>
30                     <td><a href='editar.php?idLivro=$linha[idLivro]
31                         '>Editar</a> -
32                     <a href='excluir.php?idLivro=$linha[idLivro]
33                         '>Excluir</a></td>
34                 </tr>";
35             }
36             echo "</table>";
37         }
38         catch (PDOException $e) {
39             echo $e->getMessage();
40         }
41     ?>
42 </div>
```

Explicando os códigos acima:

Linha 14: criamos uma variável \$consulta que acessa o método query que é o responsável por preparar e executar uma instrução SQL, nesse caso a instrução é um SELECT (consulta simples que retorna todos os dados cadastrados no BD)

Linha 17: Como queremos que os livros sejam exibidos dentro de uma tabela, nessa linha preparamos o topo da tabela de exibição, por isso você vê tags já conhecidas, como table, tr, td. O comando class='table table-striped' dentro da table é relacionado com o Bootstrap, estamos adicionando uma classe do Bootstrap que vai formatar nossa tabela com o CSS já predefinido dele.

Linha 25: Esse trecho do código é muito importante e interessante, estamos usando uma estrutura de repetição chamada while, nela adicionamos uma variável chamada \$linha, que será responsável por armazenar todos os dados cadastrados no nosso BD, uma tupla de cada vez.

Graças ao PDOStatement :: fetch que retorna uma linha do conjunto de resultados e o parâmetro PDO::FETCH_ASSOC que informa ao PDO para retornar o resultado como uma matriz associativa, o while consegue fazer um array (uma estrutura de dados que armazena uma coleção de elementos de tal forma que cada um dos elementos possa ser identificado por, pelo menos, um índice ou uma chave) que vai exibindo todos os livros cadastrados.

Linhas 30 e 31: Essas linhas ainda não executam as ações que elas serão responsáveis, por enquanto, apenas exibem que poderemos futuramente alterar ou excluir os livros cadastrados.

Se você acessar a página `exibe.php` já poderá ver os livros cadastrados:

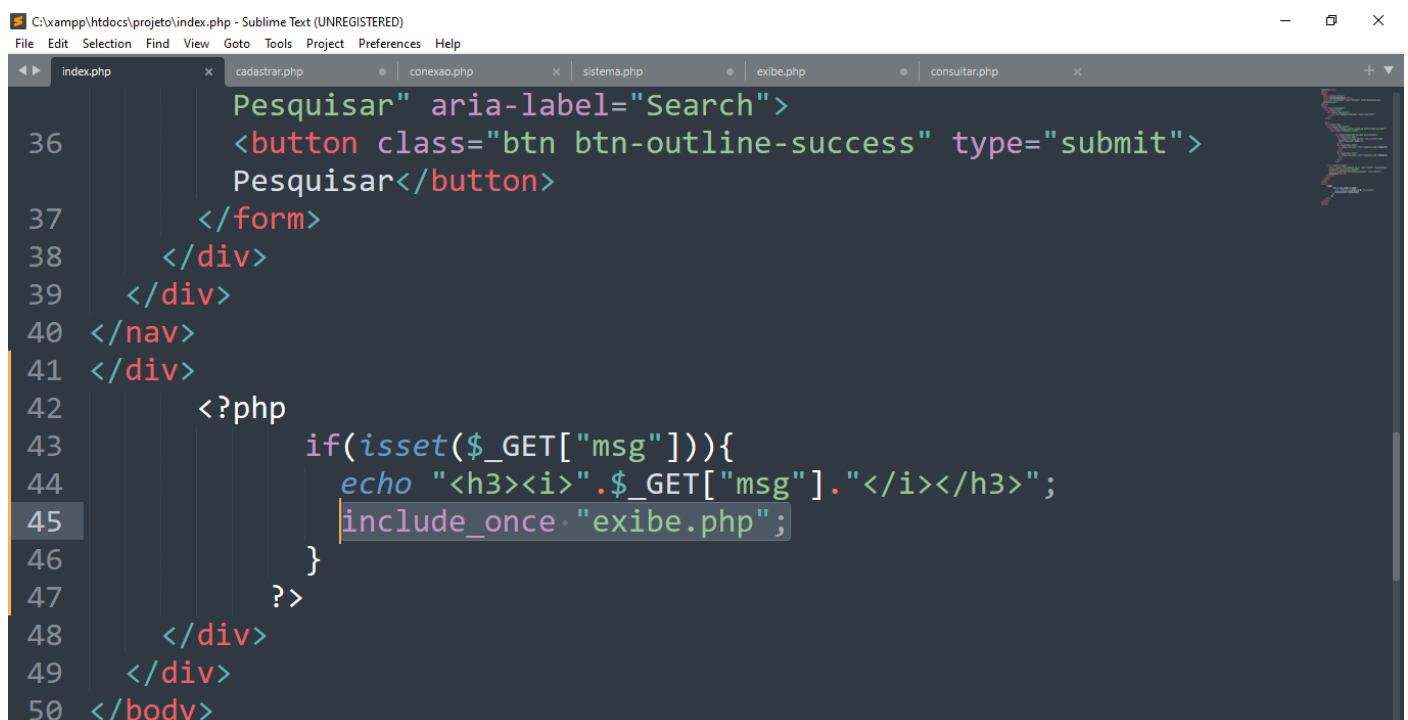


Título	Autor	Páginas	Ações
Comer, Rezar e Amar	Elizabeth Gilbert	123	Editar - Excluir

Mas só dessa forma não é muito interessante, pois o usuário teria que saber o endereço da página para ter acesso. Então vamos fazer algumas poucas modificações para já exibir o livro cadastrado logo após essa ação e também para que, quando o usuário clique na aba Consultar, ele também tenha acesso aos livros cadastrados.

Modificações na página `index.php`

Adicione a linha 45 com o comando `include_once`, para exibir os livros cadastrados abaixo da mensagem Livro Cadastrado com Sucesso!



```
index.php
36     <button class="btn btn-outline-success" type="submit">
37     Pesquisar</button>
38 </form>
39 </div>
40 </nav>
41 </div>
42 <?php
43     if(isset($_GET["msg"])){
44         echo "<h3><i>".$_GET["msg"]."</i></h3>";
45         include_once "exibe.php";
46     }
47     ?>
48 </div>
49 </div>
50 </body>
```

Crie o arquivo consultar.php para consulta e as demais ações que serão desenvolvidas:

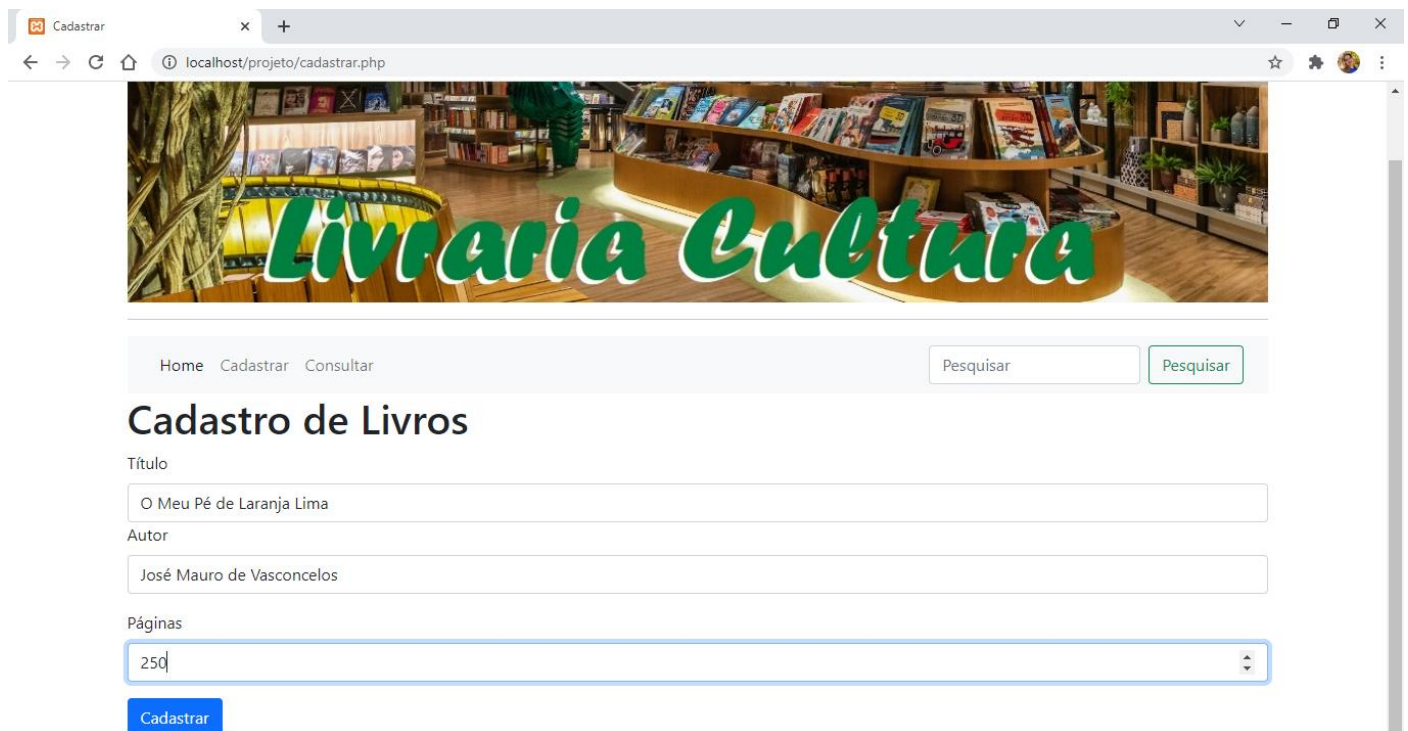
consultar.php

```
C:\xampp\htdocs\projeto\consultar.php (projeto) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

consultar.php
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>Consultar</title>
6 </head>
7 <body>
8     <?php
9         include_once "index.php";
10
11         include_once "exibe.php";
12     ?>
13 </body>
14 </html>
```

Faça testes:

1. Cadastre um novo livro:



2 – Verifique se após o cadastro o livro será exibido na página inicial, como na imagem abaixo:



3 – Se você clicar na aba Consultar, também poderá ver os livros cadastrados:

