PHP e MySQL – Introdução

Uma das maiores vantagens em utilizarmos a linguagem PHP em páginas de internet é a facilidade que ela oferece para acessar bancos de dados, em especial, o banco MySQL.

Utilizaremos aqui um banco de dados MySQL para simular um livro de visitas de uma página qualquer.

Conexão com o Banco de Dados

Crie a estrutura abaixo em um banco de dados. Em nosso modelo, criaremos um banco chamado Visitas com as seguintes tabelas:

Tabela Livro de assinaturas (Livro)

CREATE TABLE `livro` (
`cod` INT(5) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`nome` VARCHAR(255) NOT NULL ,
`localizacao` VARCHAR(50) NOT NULL ,
`mensagem` TEXT NOT NULL ,
`data` DATETIME NOT NULL
);

Vamos inserir pelos menos cinco registros para fazermos nossos testes com o PHP.

Com o banco de dados e a tabela criados e alguns registros inseridos passaremos agora à conexão do banco com a página php.

Em primeiro lugar, criaremos uma página que irá fazer a conexão com o banco e exibir os dados existentes na tabela.

Iremos mostrar o código da página mostra.php e em seguida explicaremos passo a passo os comandos utilizados:

```
<?php
// Mensagens de Erro
$msg[0] = "Conexão com o banco falhou!";
$msg[1] = "Não foi possível selecionar o banco de dados!";

// Fazendo a conexão com o servidor MySQL
$conexao = mysqli_connect("localhost","root","") or die($msg[0]);
mysqli_select_db($conexao, "visitas") or die($msg[1]);
// Colocando o Início da tabela
?>
<table border="1"><tr>
   <td><b>Cód</b></td>
   <td><b>Nome</b></td>
   <td><b>Localização</b></td>
</tr>
<?php

// Fazendo uma consulta SQL e retornando os resultados em uma tabela HTML
$query = "SELECT cod,nome,localizacao FROM livro ORDER BY nome";
$resultado = mysqli_query($conexao,$query);
while ($linha = mysqli_fetch_array($resultado)) {
   ?>
   <tr>
```

```
      <td><?php echo $linha['cod']; ?></td>
      <td><?php echo $linha['nome']; ?></td>
      <td><?php echo $linha['localizacao']; ?></td>
   </tr>
   <?php
}
?>
</table>
```

As primeiras linhas do código criam um vetor com duas strings que utilizaremos para dar mensagens de erro na hora de conexão ou seleção do banco de dados.

Em seguida, utilizamos a função **mysqli_connect()** que é responsável pela abertura da conexão com o banco de dados mysql. Como parâmetros, enviamos o endereço do banco (nesse caso foi localhost, mas poderia ser o IP ou o host do servidor por exemplo), o nome do usuário (nesse caso o usuário foi o root), e enviamos a senha (que nesse caso é vazia, por isso “”). Na linha de comando terminamos com a chamada *or die,* que mostrará a mensagem que estiver em seguida, nesse caso usamos o vetor da primeira linha na posição zero.

Por que usar **mysqli_connect()** e não usar **mysql_pconnect()**? Depende o que você irá fazer em sua página. A letra **p** antes do **connect** é abreviação de conexão persistente. O que é isso? Acesse o manual do PHP para termos uma noção clicando em http://www.php.net/manual/en/features.persistent-connections.php

Em seguida acionamos o comando **mysqli_select_db()** que fará a seleção do banco de dados a ser utilizado. Passamos como parâmetros nesse caso o banco visitas que foi criado no mysql e indicamos em qual conexão. Mais uma vez encerramos com o comando *or die* que exibirá uma mensagem caso falhe, nesse segundo caso utilizamos o mesmo vetor com a posição um.

Em seguida vem a demostração da flexibilidade do PHP, você pode sair e entrar da programação PHP a qualquer momento, bastando utilizar as tags <?PHP e ?>. Neste ponto, a gente fecha a programação PHP e começa a desenhar uma tabela html simples. Após o desenho do cabeçalho da tabela, acionamos mais uma vez o modo PHP para trazermos os dados do banco de dados para a página.

Vamos na seqüência usar a função **mysqli_query()** que é responsável pela consulta no banco de dados. Basta inserir uma instrução SQL válida para termos o resultado. A função tem como parâmetros uma sql (ou variável string que contenha uma sql) e a conexão que a sql será executada.

Em seguida vem um pequeno truque para mostrarmos todos os dados retornados na consulta. Faremos um laço **While** ler linha a linha o resultado da mysql_query (que neste caso foi atribuída à variável $resultado) juntamente com a função **mysqli_fecth_array()**, que transformará $resultado em um vetor. Como temos 5 linhas em nosso banco de dados, o laço repetirá 5 vezes apenas.

Observe que os dados de cada linha da tabela serão atribuídos ao vetor $linha criado na instrução, e acessaremos um a um com a chamada de seu nome como índice (**$linha[´coluna´]**), como podemos ver em $linha[´nome´], $linha[´cod´], etc.

Depois de passarmos as 5 linhas, novamente fechamos o modo PHP e acionamos o HTML para fechar a tabela.

Basicamente essas são as instruções necessárias para acessarmos os dados em uma tabela de um banco de dados mysql.

GRAVAR DADOS NO NOSSO LIVRO DE VISITAS

Para oferecermos a opção de salvar os dados no livro de visitas, utilizaremos um formulário html solicitando os dados a serem preenchidos e enviando-os para o php. Veja o código da página assinar.php

```
<?PHP
//vamos verificar se o método da página é POST
if (getenv("REQUEST_METHOD") == "POST") {
   // Configura as variáveis do método POST para virarem variáveis
   $nome = $_POST['nome'];
   $localizacao = $_POST['localizacao'];
   $mensagem = $_POST['mensagem'];

   // Caso todos os campos forem preenchidos, inclui a mensagem no
   // banco de dados. Caso isso não aconteça, gera uma mensagem de
   // erro que será impressa no browser mais a frente.
   if ($nome and $localizacao and $mensagem) {
      $conexao = mysqli_connect("localhost","root","");
      mysqli_select_db($conexao, “visitas”);
$query = "INSERT INTO livro
VALUES('00000','$nome','$localizacao','$mensagem',NOW())";
      mysqli_query($conexao,$query);
      header("Location: ler.php");
   } else {
      $err = "Preencha todos os campos!";
   }
}

?>
<html>
<head>
   <title>Livro de Visitas: Assinar</title>
</head>
<body bgcolor="white">

<h1>Assine o Livro de Visitas</h1>

<?PHP
// Se ocorreu algo de errado, então vai existir uma variável $err
// contendo a mensagem. Imprime-se então em fonte vermelha esta
// mensagem.
if ($err) {
   ?>
   <ul><font color="red"><?PHP echo $err; ?></font></ul>
   <?PHP
}
?>

<form method="post" action="assinar.php">
<table border="0"
<tr>
   <td>Nome: </td>
   <td><input type="text" size="15" name="nome" maxlength="250"></td>
</tr>
<tr>
   <td>Localização: </td>
   <td><input type="text" size="15" name="localizacao" maxlength="45"></td>
</tr>
<tr>
   <td colspan="2">
   <textarea cols="60" rows="10" name="mensagem">Digite aqui sua
mensagem!</textarea>
   </td>
```

```
</tr>
</table>
<input type="submit" value="Assinar">
</form>

</body>
</html>
```

Aqui montamos um formulário padrão em HTML que receberá os dados do usuário. Quando pressionar o botão Assinar o formulário envia os dados com o método escolhido (no caso o POST para que os dados não sejam visíveis para o usuário) para a própria página, como se ela fosse chamada novamente. Nada impede que mandemos para outra página que fará apenas a validação e gravação dos dados.

Ao entrar na página, ela verifica se os dados estão chegando através de POST. Caso afirmativo, quer dizer que o usuário pressionou o botão de assinar. Porém não podemos garantir que os dados foram preenchidos. Por isso nossa primeira função é identificar isso.

Teremos então que criar variáveis com os dados que estão chegando via POST. Para isso, basta que criemos variáveis (no exemplo $nome) e igualemos aos dados que estão chegando, um a um (por exemplo $nome = $_POST[´nome´]). Os dados chegam no vetor $_POST, bastando identificarmos o ítem desejado.

Uma vez com os dados checamos se os mesmos estão preenchidos (if $nome and $mensagem and $localizacao). Nesta instrução, caso algum deles estiver vazio retornará falso, não deixando o programa abrir a conexão com o banco nem tentando gravar dados inválidos.

Uma vez que todos estão preenchidos, deveremos abrir a conexão com o banco novamente (como explicado acima) e proceder a gravação, com uma SQL de Insert (inserção).

Com a instrução preenchida, usamos mysql_select_db para executar nosso SQL.

Em seguida, encontramos a instrução header. Essa instrução é muito utilizada para inserirmos dados no cabeçalho (head) do html. Uma limitação do HTML é que nenhuma linha de página deve ter sido escrita antes de sua utilização. Nesse caso, como não escrevemos nada, enviamos a instrução Location, que irá desviar a página para outro arquivo (no caso, ler.php)

EXIBINDO OS DADOS DO NOSSO LIVRO DE VISITAS

Agora mostraremos as assinaturas de nosso livro de visitas aos nossos visitantes. Para isso criaremos o arquivo chamado ler.php que está mostrado abaixo:

```
<html>
<head>
   <title>Livro de Visitas: Ler</title>
</head>
<body bgcolor="white">

<h1>Livro de Visitas</h1>

<?PHP
// Verifica se existe a variável $begin, que vai indicar a número
// da mensagem que vai aparecer no começo. Se não existir, assume-se
// que vai ser o começo, ou seja, o valor 0.
$begin = $_GET['begin'];
if (!$begin) { $begin = 0; }
```

```
// Conecta ao servidor e seleciona o banco de dados
$conexao = mysqli_connect("localhost","root","");
mysqli_select_db($conexao, “visitas”);

// Coloca na variável $total o número total de mensagens no Guestbook
$query = "SELECT count(*) FROM livro";
$query = mysqli_query($conexao,$query);
$query = mysql_fetch_array($query);
$total = $query[0];

?>
<p>
Total de mensagens postadas: <b><?PHP echo $total; ?></b>
(<a href="assinar.php">Assine você também!</a>)<br>
Exibindo <b>20</b> mensagens por página, mostrando mensagens de
<b><?PHP echo $begin+1; ?></b> a <b><?PHP echo $begin+20; ?></b>.
</p>

<?PHP
// Calcula os links para as próximas mensagens e as anteriores, de
// acordo com o número total de mensagens
if (($begin > 0) and ($begin <= 20)) {
   $anteriores = '<a href="ler.php?begin=0">Anteriores</a>';
} elseif (($begin > 0) and ($begin > 20)) {
   $anteriores = '<a href="ler.php?begin=' . ($begin-20) . '">Anteriores</a>';
} else {
   $anteriores = 'Anteriores';
}

if (($begin < $total) and (($begin+20) >= $total)) {
   $proximas = 'Próximas';
} else {
   $proximas = '<a href="ler.php?begin=' . ($begin+20) . '">Próximas</a>';
}

echo $anteriores . " | " . $proximas;

// Faz uma consulta SQL trazendo as linhas das 20 ultimas mensagens
// que foram colocadas no livro de visitas.
$query = "SELECT * FROM livro ORDER BY data DESC LIMIT $begin,20";
$query = mysqli_query($conexao,$query);

// Gera uma tabela para cada assinatura no livro de visitas (loop)
while ($linha = mysql_fetch_array($query)) {
   // Organiza a mostragem da data, já que no campo do MySQL, a data
   // se encontra em uma forma não tão legal.
   $var = $linha['data'];
   $var = explode(" ",$var);
   $dia = $var[0];
   $hora = $var[1];
   $dia = explode("-",$dia);
   $data = "$dia[2]/$dia[1]/$dia[0] às $hora";
   ?>

   <table border="0" width="70%">
   <tr><td bgcolor="navy" colspan="2"> </td></tr>
   <tr>
      <td><b>Data:</b></td>
      <td width="100%"><?PHP echo $data; ?></td>
   </tr>
   <tr>
      <td><b>Nome:</b></td>
      <td><?PHP echo $linha['nome']; ?></td>
   </tr>
```

```
   <tr>
      <td><b>Localização:</b></td>
      <td><?PHP echo $linha['localizacao']; ?></td>
   </tr>
   <tr>
      <td><b>Mensagem:</b></td>
      <td><?PHP echo $linha['mensagem']; ?></td>
   </tr>
   </table>
   <?PHP
}

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

O código em si já possui diversos comentários, mas vamos destacar algumas funções e métodos utilizados aqui:

De início identificamos uma variável que está chegando (ou não) à página pelo método GET (que estará no vetor $_GET). Essa variável será utilizada para a paginação das mensagens. Esta página está programada para mostrar páginas de mensagens de 20 em 20. Quando o livro tiver mais de 20 assinaturas, elas serão separadas em páginas controladas pela variável $begin conseguida pelo método get ($begin=$_GET[´begin´]). Ela indicará o número da mensagem que iniciará a página. Se ela não existe, então colocaremos o valor zero para indicar que está na primeira página.

Em seguida, acessamos o banco de dados (como já descrito anteriormente) e realizamos uma consulta (select count(*) from livro) para descobrirmos a quantidade de registros cadastrados na tabela. Atribuímos esse valor à variável $total, que é exibida em seguida.

Na seqüência utilizamos a variável $begin para que possamos realizar a paginação, descobrindo quais mensagens exibir, se ativa botões próximos e anteriores, etc.

Depois de paginarmos tudo, finalmente vem a seção onde iremos mostrar de fato as mensagens. A primeira coisa a reparar é a instrução DESC e LIMIT. A instrução DESC apenas indica que a ordem de exibição será inversa, ou seja, da maior para a menor (no caso, foi ordenado por data). A instrução LIMIT, como o próprio nome diz, “limita” a quantidade de linhas que será retornada, iniciando a partir do valor de $begin. Seria como se disséssemos começando de 0 até 0+20.

Em seguida, basta que mostremos os dados em seus devidos lugares, montando a formatação desejada.

Existe também um exemplo em como adaptar a exibição da data, já que no MySQL a data é armazenada em um formato diferente do que o que estamos acostumados a observar.

Alteração de registros

Para fazermos alterações nos registros, o primeiro passo é organizar uma forma de selecionarmos o registro que desejamos manipular. Existem diversas formas. Neste caso, optamos em realizar a busca com o auxílio de uma combo contendo os nomes das pessoas que assinaram nosso livro, mas vamos utilizar como parâmetro de pesquisa, o código do registro que contém aquele nome. Vejamos o código da página procurar.php que realiza a listagem dos nomes na combo:

```
<html>
      <head><title>Procuras</title></head>
<body><center>
   <form method=post action=exibe.php>
```

```
   <table    border=1    bordercolor=blue><tr><td    colspan=2><center><b><font
face=arial      size=4      color=blue>Alteração      ou      exclusão      de
registros</font></b></center></td></tr>
   <tr><td width=150><font face=arial>Selecione um nome: </td><td>
   <select name=procura size=1>
   <?PHP
            include("conecta.php");
            $query = "SELECT nome, cod FROM livro ORDER BY data DESC";
            $query = mysqli_query($conexao,$query);
            while ($linha = mysql_fetch_array($query)) {
             echo '<option value='.$linha['cod'].'>'.$linha['nome'].'</option>';


            }
   ?>
   </select></td></tr>
   <tr><td><center><input    type=reset    value=Limpar></td><td><center><input
type=submit value=Enviar></td></tr>
   </table>
   </form>
   <br><br><center><a href=index.php>Voltar ao menu principal</a></center>
   </body>
</html>
```

O código começa com a declaração de um formulário, simples, em html. Utilizaremos o objeto select para criarmos a combo, que será preenchida dinamicamente com o PHP a seguir.

A primeira novidade aparece na função **include()**. Essa função tem a capacidade de inserir um arquivo inteiro dentro do arquivo que estamos trabalhando.

Nesse caso, optamos em criar um arquivo padrão com as instruções de conexão com o banco de dados e seleção do banco padrão. Criamos o arquivo conecta.php com as seguintes instruções:

```
<?PHP
// Mensagens de Erro
$msg[0] = "Conexão com o banco falhou!";
$msg[1] = "Não foi possível selecionar o banco de dados!";
// Fazendo a conexão com o servidor MySQL
$conexao = mysqli_connect("localhost","root","") or die($msg[0]);
mysqli_select_db($conexao, "visitas") or die($msg[1]);
?>
```

E quando a página procurar.php insere a linha **include("conecta.php")** seria como se copiássemos todo o conteúdo do arquivo conecta.php para procurar.php naquela posição. Dessa forma, em todos os próximos arquivos aqui descritos, sempre que precisarmos abrir uma conexão com o banco de dados livros chamaremos a instrução include("conecta.php") ao invés de digitarmos todas essas instruções.

Feita a conexão preparamos uma consulta listando todos os nomes e códigos em ordem de entrada inversa e criamos a instrução para que todos os nomes sejam listados na select através do comando Option do HTML. Note que no option temos dois parâmetros. O primeiro identifica qual o dado que será enviado quando o select estiver selecionado e o segundo qual dado será exibido na página para que o usuário possa procurar a opção desejada.

Nesse caso colocamos em value do option a coluna cod que será enviada como post para a próxima página e colocamos a coluna nome para ser exibido na tela com essa instrução

```
echo '<option value='.$linha['cod'].'>'.$linha['nome'].'</option>';
```

Feito isso já temos todos os nomes listados prontos para enviar o código para a próxima página pelo método Post, bastando o usuário clicar no botão enviar

(submit do form).

Será acionada a página exibe.php, com o seguinte código:

```
<?PHP
//vamos verificar se o método da página é POST
if (getenv("REQUEST_METHOD") == "POST") {
   // Configura as variáveis do método POST para virarem variáveis
   $cod = $_POST['procura'];
   //Faz a busca pelo código solicitado de acordo com o nome pesquisado na
página anterior
      include("conecta.php");
      $query = "Select * from livro where cod = ".$cod;
      $query = mysqli_query($conexao,$query);
}
?>
<html>
<head>
   <title>Livro de Visitas: Alterar</title>
</head>
<body bgcolor="white">

<h1>Alterações no Livro de Visitas</h1>

<?PHP
// Se ocorreu algo de errado, então vai existir uma variável $err
// contendo a mensagem. Imprime-se então em fonte vermelha esta
// mensagem.
if ($err) {
   ?>
   <ul><font color="red"><?PHP echo $err; ?></font></ul>
   <?PHP
   } else {
$linha = mysql_fetch_array($query);
   }
?>
<form method="post" action="alterar.php">
<table border="0"
<tr>
   <td>Nome: </td>
   <td><input   type="text"   size="150"   name="nome"   value   ="<?PHP   echo
$linha['nome']; ?>" maxlength="250"></td>
</tr>
<tr>
   <td>Localização: </td>
   <td><input  type="text"  size="150"  name="localizacao"  value  ="<?PHP  echo
$linha['localizacao']; ?>" maxlength="45"></td>
</tr>
<tr>
   <td>Data: </td>
   <td><input   type="text"   size="20"   name="data"   value   ="<?PHP   echo
$linha['data']; ?>" maxlength="19"></td>
</tr>
<tr>
   <td colspan="2">
   <textarea cols="60" rows="10" name="mensagem"><?PHP echo $linha['mensagem'];
?></textarea>
   </td>
</tr>
</table>
<input type=hidden name=cod value = "<?PHP echo $linha['cod']; ?>">
<input type=reset value="Recarregar"> <input type="submit" value="Alterar">
</form>
<form method=post action=deletar.php>
```

```
<input type=hidden name=cod value = "<?PHP echo $linha['cod']; ?>">
<br><input type=submit value="Apagar"></form>
<br><br><center><a href=index.php>Voltar ao menu principal</a></center>
</body>
</html>
```

Esta página receberá o código de acordo com o nome selecionado em procurar.php. Ela inicia checando se os dados estão chegando via POST para pode realizar a busca do registro. Mais uma vez é chamada a função include para trazer o arquivo conecta.php para que possamos abrir a conexão com a tabela livro.

Assim que é feita a consulta, utilizamos um formulário semelhante ao formulário utilizado na hora de assinar o livro de visitas (da página assinar.php), porém com a diferença de mostrarmos os dados nos campos, relativos ao nome selecionado na página anterior. Mais uma vez é criado um formulário com os dados preenchidos que aponta para outra página que, depois de fazer as alterações e pressionar o botão alterar, cuidará de realizar as alterações no banco de dados.

O final do formulário será utilizado na hora de apagar um registro. Será comentado adiante neste texto.

A página responsável pela alteração é a alterar.php, descrita abaixo:

```
<?PHP
//vamos verificar se o método da página é POST
if (getenv("REQUEST_METHOD") == "POST") {
   // Configura as variáveis do método POST para virarem variáveis
   $nome = $_POST['nome'];
   $localizacao = $_POST['localizacao'];
   $mensagem = $_POST['mensagem'];
   $data = $_POST['data'];
   $cod = $_POST['cod'];
   // Caso todos os campos forem preenchidos, inclui a mensagem no
   // banco de dados. Caso isso não aconteça, gera uma mensagem de
   // erro que será impressa no browser mais a frente.
   if ($nome and $localizacao and $mensagem and $data) {
      include("conecta.php");
  $query = "UPDATE `livro` SET `nome` = '".$nome."',`localizacao` =
'".$localizacao."',`mensagem` = '".$mensagem."',`data` = '".$data."' WHERE `cod`
=".$cod;
  mysqli_query($conexao,$query);
      header("Location: procurar.php");
   } else {
      $err = "Preencha todos os campos!";
echo err."<br><a href=javascript:history.go(-1)>Voltar</a>";
   }
}
?>
```

Essa página praticamente não terá nenhum conteúdo, apenas se o usuário entrar direto nela sem enviar dados via POST mostrará uma mensagem para retornar ou se algum campo for deixado em branco, que no caso receberá uma opção para voltar e corrigir antes de salvar as alterações.

A página começa com a checagem se os dados vieram via POST. Se sim, ela trata de converter as variáveis do vetor POST para variáveis locais e em seguida efetua a SQL que irá fazer a alteração.

Logo depois a função header (“location: “) irá desviar a página para a página procurar.php novamente. Se o nome for alterado já estará visível a alteração.

Exclusão de registros

Na página procurar.php observamos a existência também de um formulário à

parte no final, que será responsável pelo acionamento de uma página (deletar.php) específica para que possamos apagar o registro.

Uma vez pressionado o botão, ele acionará essa página (deletar.php) com o seguinte código:

```
<?PHP
//vamos verificar se o método da página é POST
if (getenv("REQUEST_METHOD") == "POST") {
   // Configura as variáveis do método POST para virarem variáveis
   $cod = $_POST['cod'];
      include("conecta.php");
        $query = "DELETE FROM `livro` WHERE `cod` =".$cod;
        mysqli_query($conexao,$query);
      header("Location: procurar.php");
   } else {
      $err = "Selecione o registro antes de excluir!";
echo $err."<br><a href=javascript:history.go(-1)>Voltar</a>";
   }
?>
```

Assim como na página alterar.php essa página também não produz nenhum resultado em tela. Ela apenas abre a conexão com o banco e executa uma SQL com o código recebido pelo POST da página procurar.php e apaga o registro. Tendo sucesso ela redireciona pelo header para a página procurar.php novamente e se ela não for acionada corretamente dará uma mensagem de erro, solicitando que o usuário selecione o registro antes.

Procura de registros

Criamos aqui duas formas simples de pesquisa de registros pelo nome. Uma delas irá solicitar do usuário um nome para buscas e apenas exibirá os dados na tela. A outra solicitará o nome também, mas quando mostrar o resultado mostrará dois botões para que o usuário possa, à partir da consulta, alterar ou apagar os registros encontrados.

Vamos ao código da página que recebe os nomes (consultar.php):

```
<html>
      <head><title>Procuras</title></head>
<body><center>
   <form method=post action=resultados.php>
   <table    border=1    bordercolor=blue><tr><td    colspan=2><center><b><font
face=arial size=4 color=blue>Consulta de registros</font></b></center></td></tr>
   <tr><td width=150><font face=arial>Digite um nome: </td><td>
   <input type=text name=nome width=150></td></tr>
   <tr><td><center><input    type=reset    value=Limpar></td><td><center><input
type=submit value=Procurar></td></tr>
   </table>
   </form>

   <hr color=red>
   <form method=post action=resultados2.php>
   <table    border=1    bordercolor=blue><tr><td    colspan=2><center><b><font
face=arial size=4 color=blue>Consulta com opções</font></b></center></td></tr>
   <tr><td width=150><font face=arial>Digite um nome: </td><td>
   <input type=text name=nome width=150></td></tr>
   <tr><td><center><input    type=reset    value=Limpar></td><td><center><input
type=submit value=Procurar></td></tr>
   </table>
   </form>
   <br><br><center><a href=index.php>Voltar ao menu principal</a></center>
   </body>
```

```
</html>
```

Observe que temos praticamente duas vezes o mesmo código nesta página, porém uma delas o formulário envia os dados para a página resultados.php e na outra para resultados2.php.

Essa página é puramente html e não contém nenhum código php. Ela apenas é usada para criar os formulários e enviar os dados via POST

Em seguida vamos analisar o código da página resultados.php

```
<html><head><title>Resultados da pesquisa</title></head>
<body>
<table border="1"><tr>
   <td><b>Cód</b></td>
   <td><b>Nome</b></td>
   <td><b>Localização</b></td>
   <td><b>Data</b></td>
   <td><b>Mensagem</b></td>
</tr>
<?php
//chama a conexão com o banco de dados
include("conecta.php");
//pega a variável POST
$nome = $_POST['nome'];
// Fazendo uma consulta SQL e retornando os resultados em uma tabela HTML
$query =  "SELECT * FROM livro WHERE nome = '$nome'";
$resultado = mysqli_query($conexao,$query);
//mostrando os resultados
while ($linha = mysqli_fetch_array($resultado)) {
   ?>
   <tr>
      <td><?php echo $linha['cod']; ?></td>
      <td><?php echo $linha['nome']; ?></td>
      <td><?php echo $linha['localizacao']; ?></td>
        <td><?php echo $linha['data']; ?></td>
        <td><?php echo $linha['mensagem']; ?></td>
   </tr>
   <?php
}
?>
      <br><br><center><a href=index.php>Voltar ao menu principal</a></center>
</table>
</body>
</html>
```

Aqui vamos receber os dados via POST e fazer uma consulta no banco de dados com o nome (ou pedaço dele) que foi digitado pelo usuário.

Note na consulta a utilização de LIKE na SQL e a adição de % antes e depois do nome recebido no POST. Isso é para que caso o usuário digite apenas Silva, por exemplo, a página procure no banco nomes que contenham Silva em qualquer posição (início, meio ou fim do nome).

Uma vez localizados os nomes que contenham a procura do usuário, os dados são mostrados através de uma tabela semelhante à página mostrar.php, primeiro exemplo do manual.

Já a página resultados2.php descrita abaixo, realiza as mesmas funções, porém exemplifica como podemos criar dois botões para que, com os resultados encontrados, possamos alterar ou apagar os registros desejados.

```
<html><head><title>Resultados da pesquisa</title></head>
<body>
<table border="1"><tr>
```

```
   <td><b>Opção 1</b></td>
   <td><b>Opção 2</b></td>
   <td><b>Cód</b></td>
   <td><b>Nome</b></td>
   <td><b>Localização</b></td>
   <td><b>Data</b></td>
   <td><b>Mensagem</b></td>
</tr>
<?php
//chama a conexão com o banco de dados
include("conecta.php");
//pega a variável POST
$nome = $_POST['nome'];
// Fazendo uma consulta SQL e retornando os resultados em uma tabela HTML
$query =  "SELECT * FROM livro WHERE nome like '%".$nome."%'";
$resultado = mysqli_query($conexao,$query);
//mostrando os resultados
while ($linha = mysqli_fetch_array($resultado)) {
   ?>
   <tr>
      <td><form action="exibe.php" method=post><input type=hidden name=procura
value=<?PHP echo $linha['cod'];?>><input type=submit value=Alterar></form></td>
      <td><form action="deletar.php" method=post><input type=hidden name=cod
value=<?PHP echo $linha['cod'];?>><input type=submit value=Apagar></form></td>
      <td><?php echo $linha['cod']; ?></td>
      <td><?php echo $linha['nome']; ?></td>
      <td><?php echo $linha['localizacao']; ?></td>
        <td><?php echo $linha['data']; ?></td>
        <td><?php echo $linha['mensagem']; ?></td>
   </tr>
   <?php
}
?>
</table>
<center><a href=index.php>Voltar ao menu principal</a></center>
</body>
</html>
```

Esses botões são criados em duas células da tabela e pertencem a um formulário padrão html, mas o único parâmetro visível é o submit. Observamos que os dados necessários estão ocultos (input type=hidden) e tem os mesmos nomes das páginas que irão recebê-los (no caso alterar.php e deletar.php).