Criação de um CRUD em PHP com bons recursos

A proposta aqui é a de criação de 4 aplicativos inteiramente "na unha", ou seja, criados inteiramente com código nativo PHP, sem o uso de um Framework e passo a passo. Sei que atualmente, em geral, é mais produtivo criar com um Framework, mas lembro pelo menos de duas situações em que é mais vantajoso criar "na unha": o aprendizado do PHP e a grande potencial de customização.

Recursos:

- PDO
- Paginação
- Busca
- BootStrap
- Suporte ao MySQL e PostgreSQL testados

CRUD – Significa Create, Read, Update e Delete. É uma forma de se criar um aplicativo muito prática, com praticamente todos os recursos de um aplicativo para gerenciar uma tabela, todos numa mesma tela.

Este tutorial é o primeiro de quatro.
Este trata da criação de um CRUD com PHP estruturado.
O segundo da criação de um CRUD com PHP orientado a objetos.
O terceiro da criação de um aplicativo para gerenciar duas tabelas com PHP estruturado.
O quarto e último para criar um aplicativo com PHP orientado a objetos.

Importante - Estes aplicativos, para serem compreendidos facilmente, requerem conhecimento de PHP, de Bootstrap, ou CSS e JavaScript.

Testando exemplos de paginação encontrados via internet

Exemplo simples usando o plugin do jquery bootpag:

https://www.kodingmadesimple.com/2017/01/simple-ajax-pagination-in-jquery-php-pdo-mysql.html

Instalado localmente

A estrutura da arquivos é esta:

/css/
/fonts/
/js/
/db.sql
/db_connect.php
/fetch_data.php
/index.php

Vejamos o código dos arquivos básicos, já com algumas alterações nos nomes de campos: db_connect.php, index.php e fetch_data.php:

db_connect.php

```
<?php
$hostname = "localhost";
$username = "root";
$password = "root";
$database = "paginacao";
$row_limit = 5;

// connect to mysql
try {
   $pdo = new PDO("mysql:host=$hostname;dbname=$database", $username, $password);
   $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $err) {
   die("Error! " . $err->getMessage());
}
?>
```

index.php

```
<?php
include_once("db_connect.php");

$stmt = $pdo->prepare("SELECT COUNT(*) FROM clientes");
$stmt->execute();
$rows = $stmt->fetch();

// get total no. of pages
$total_pages = ceil($rows[0]/$row_limit);
?>

<!DOCTYPE html>
<html lang="en">
<head>
   <title>AJAX Pagination using PHP & MySQL</title>
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css" />
   <style type="text/css">
   .panel-footer {
```

```
        padding: 0;
        background: none;
    }
    </style>
</head>
<body>
<br/>
<div class="container">
    <div class="panel panel-default">
        <div class="panel-heading text-center"><h3>jQuery PHP Pagination Demo</h3></div>

        <table class="table table-bordered table-hover">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Nome</th>
                    <th>Email</th>
                    <th>Nascimento</th>
                    <th>CPF</th>
                </tr>
            </thead>
            <tbody id="pg-results">
            </tbody>
        </table>
        <div class="panel-footer text-center">
            <div class="pagination"></div>
        </div>
    </div>
</div>

<script src="js/jquery-1.10.2.min.js" type="text/javascript"></script>
<script src="js/bootstrap.min.js" type="text/javascript"></script>
<script src="js/jquery.bootpag.min.js" type="text/javascript"></script>

<script type="text/javascript">
$(document).ready(function() {
    $("#pg-results").load("fetch_data.php");
    $(".pagination").bootpag({
        total: <?php echo $total_pages; ?>,
        page: 1,
        maxVisible: 15// páginas da paginação
    }).on("page", function(e, page_num){
        e.preventDefault();
        $("#results").prepend('<div class="loading-indication"><img src="ajax-loader.gif" />
Loading...</div>');
        $("#pg-results").load("fetch_data.php", {"page": page_num});
    });
});
</script>

</body>
</html>
```

fetch_data.php

```
<?php
include_once("db_connect.php");

if (isset($_POST["page"])) {
   $page_no = filter_var($_POST["page"], FILTER_SANITIZE_NUMBER_INT,
FILTER_FLAG_STRIP_HIGH);
   if(!is_numeric($page_no))
      die("Error fetching data! Invalid page number!!!");
} else {
   $page_no = 1;
}

// get record starting position
$start = (($page_no-1) * $row_limit);

$results = $pdo->prepare("SELECT * FROM clientes ORDER BY id LIMIT $start, $row_limit");
$results->execute();

while($row = $results->fetch(PDO::FETCH_ASSOC)) {
   echo "<tr>" .
   "<td>" . $row['id'] . "</td>" .
   "<td>" . $row['nome'] . "</td>" .
   "<td>" . $row['email'] . "</td>" .
   "<td>" . $row['nascimento'] . "</td>" .
   "<td>" . $row['cpf'] . "</td>" .
   "</tr>";
}
?>
```

Com esta estrutura básica, onde temos apenas a listagem dos registros com a paginação, agora iremos adicionar novos recursos.

Já vem com alguns bons recursos como PDO e BootStrap.

Agora irei adicionar:

- Ações para cada registro: Editar e Excluir e um botão para inserir um novo registro, tornando nosso exemplo em um CRUD

Começarei por trocar a tabela por uma outra, a clientes, que criarei usando o datagenerator:

https://github.com/benkeen/generatedata/zipball/master

Com 100 registros.

Veja como ficou

Neste caso eu alterei o index.php, na linha 60
maxVisible: 15

Para mostrar uma quantidade maior de páginas. Veja que não alterei a quantidade de registros por página. Vou fazer isso quando mudar a tabela para a de um milhão de registros, no arquivo db_connect.php

Depois testarei com uma tabela com a mesma estrutura da anterior, mas com um milhão de registros para ver como se comporta a paginação com bootpag com muitos registros.

Veja agora com um milhão de registros, com 10 registros por página e 20 páginas por vez

Agora quero customizar a paginação para que mostre os botões Primeiro e Último, além dos existentes.

Alterei o código do bootpag na index.php para este:

```
<script type="text/javascript">
$(document).ready(function() {
    $("#pg-results").load("fetch_data.php");
    $(".pagination").bootpag({
        total: <?php echo $total_pages; ?>,
        page: 1,
        maxVisible: 20,// páginas da paginação
        leaps: true,
        firstLastUse: true,
        first: 'Primeiro',//←
        last: 'Último',//→
        wrapClass: 'pagination',
        activeClass: 'active',
        disabledClass: 'disabled',
        nextClass: 'next',
        prevClass: 'prev',
        lastClass: 'last',
        firstClass: 'first'
    }).on("page", function(e, page_num){
        //e.preventDefault();
        $("#results").prepend('<div class="loading-indication"><img src="ajax-loader.gif" />
Loading...</div>');
        $("#pg-results").load("fetch_data.php", {"page": page_num});
    });
});
</script>
```

Agora a página fica assim:

Veja na última página.

As informações extras foram colhidas no site do bootpag:
https://botmonster.com/jquery-bootpag/ *no* Full example

Mudei o título na tag <title> e na <h3> da <body> para
CRUD com Paginação, PDO, BootStrap e Busca
Páginas por vez

Para facilitar a alteração eu mudei a linha

maxVisible: 15,// páginas da paginação

para
maxVisible: <?php echo $max_visible; ?>,// páginas da paginação

E adicionei a variável max_visible logo no início do arquivo, abaixo de $total_pages …

$max_visible = 15;

Adicionar form para busca de registros

Mudei o index.php adicionando isso:

```
<div class="container">
   <div class="panel panel-default">
      <div class="panel-heading text-center"><h3>CRUD com Paginação, PDO, BootStrap e
Busca</h3></div>
      <div class="row">
         <!-- Form de busca-->
         <div class="col-md-12">
            <form action="search.php" method="get" >
            <div class="pull-right topo" style="padding-left: 0;"  >
              <span class="pull-right">
                <label class="control-label" for="palavra" style="padding-right: 5px;">
                  <input type="text" value="" placeholder="Nome ou parte" class="form-control"
name="keyword">
                </label>
                </span>
              <button class="btn btn-primary">Busca</button> 
            </div>
            </form>
          </div>
         </div>

      <table class="table table-bordered table-hover">
```

Vamos organizar um pouco o código, separando em arquivos: header.php, … footer.php

Criei um arquivo chamado functions.php para guardar algumas variáveis e funções.

header.php

```
<?php
include_once("db_connect.php");
include_once("functions.php");

$stmt = $pdo->prepare("SELECT COUNT(*) FROM clientes");
$stmt->execute();
$rows = $stmt->fetch();

// get total no. of pages
$total_pages = ceil($rows[0]/$row_limit);
?>

<!DOCTYPE html>
<html lang="en">
<head>
   <title><?=$title?></title>
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css" />
   <link href="css/style.css" rel="stylesheet" type="text/css" />
   <style type="text/css">
   .panel-footer {
      padding: 0;
      background: none;
   }
   </style>
</head>
```

Este trecho veio do início do index.php, onde eu adicionei o include:

```
<?php
include_once("header.php");
?>
```

Ao final adicionei um footer.php assim:

```
<div class="container rodape" align="center">
   <i>"Adaptação de <a href="https://ribafs.org" target="_blank">Ribamar FS</a></i>
</div>
<br>
</body>
</html>
```

E adicionei um include ao final do index.php assim:

```
</script>
<?php include_once("footer.php"); ?>
</body>
</html>
```

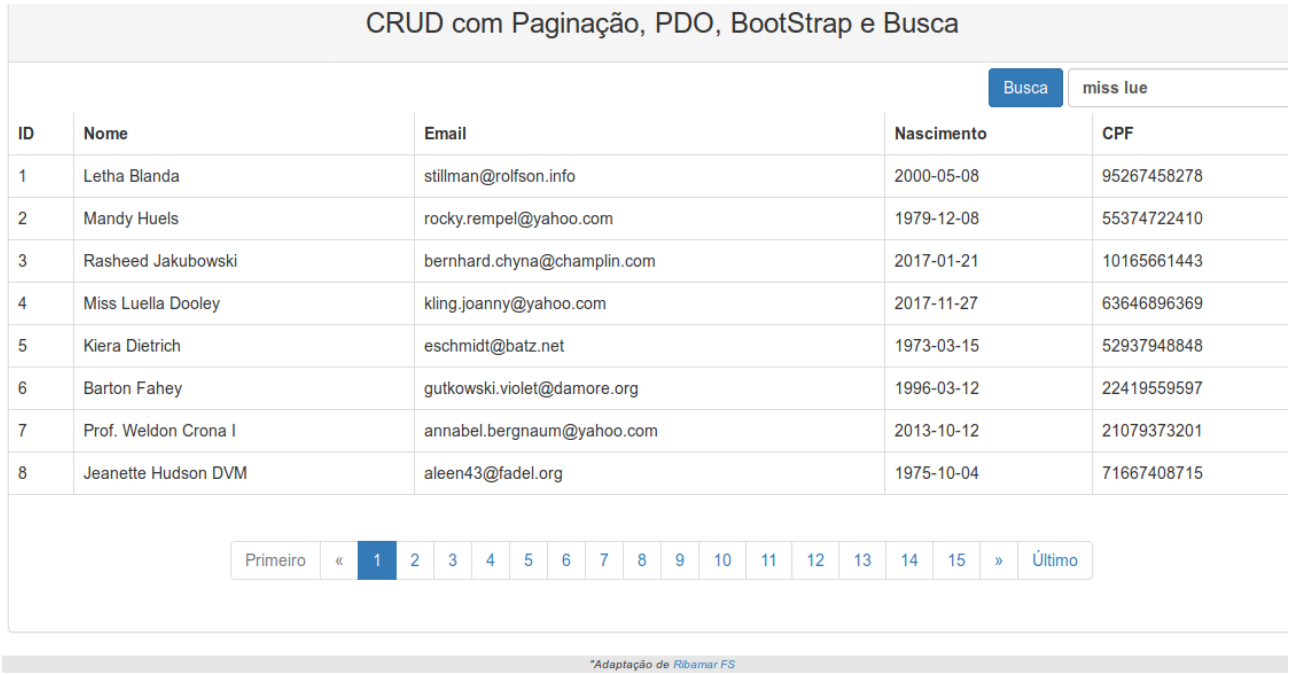
O functions.php está assim:

```
<?php

// Variáveis
$title = 'CRUD com Paginação, PDO, BootStrap e Busca';
$max_visible = 15;
```

E o código continua funcionando como antes.

Veja



Criei um pequeno style.css na pasta css e adicionei ao index.php:

```
.top{
    padding-top:5px;
}

.footer{
    background-color:#e6e6e6;
    width:85%;
    font-size:10px;
    text-align:center;
}

.header{
    text-align:center;
    background-color:#8ad3f7;
        padding-top: 10px;
        padding-bottom: 10px;
}
```

Vamos adicionar o arquivo de Busca

Após entrar o código no index.php, vou adicionar o arquivo search.php:

```
<?php
require_once('./header.php');
require_once('./db_connect.php');

// Busca
if(isset($_GET['keyword'])){
   $keyword=$_GET['keyword'];

   $sql = "select * from clientes WHERE nome LIKE :keyword order by id";
   $sth = $pdo->prepare($sql);
   $sth->bindValue(":keyword", $keyword."%");
   $sth->execute();
   //$nr = $sth->rowCount();
   $rows =$sth->fetchAll(PDO::FETCH_ASSOC);
}
?>
<div class="container">
   <div class="panel panel-default">
      <div class="panel-heading text-center"><h3><?=$title?></h3></div>
<?php
print '<div class="text-center"><h4>Registro(s) encontrado(s): '.count($rows).' com
'.$keyword.'</h4></div>';

if(count($rows) > 0){
?>
   <table class="table table-hover">
         <thead>
            <tr>
               <th>ID</th>
               <th>Nome</th>
               <th>Email</th>
               <th>Nascimento</th>
               <th>CPF</th>
            </tr>
         </thead>

<?php
   // Loop através dos registros recebidos
   foreach ($rows as $row){
      echo "<tr>" .
      "<td>" . $row['id'] . "</td>" .
      "<td>" . $row['nome'] . "</td>" .
      "<td>" . $row['email'] . "</td>" .
      "<td>" . $row['nascimento'] . "</td>" .
      "<td>" . $row['cpf'] . "</td>" .
      "</tr>";
   }
   echo "</table>";
```

```
}else{
   print '<h3>Nenhum Registro encontrado!</h3>
</div>';
}
?>

<div class="text-center"><input name="send" class="btn btn-warning" type="button"
onclick="location='index.php'" value="Voltar"></div>
</div>
<br>
<?php require_once('./footer.php'); ?>
```

Agora podemos localizar os clientes pelo nome ou parte do nome.

Veja a busca em funcionamento

Na index.php entre com um nome ou parte de um nome e tecle enter

Adicionar botão para adicionar novo registro

Este botão ficará na index, à esquerda do form de busca.

Adicionei o seguinte código para o index.php:

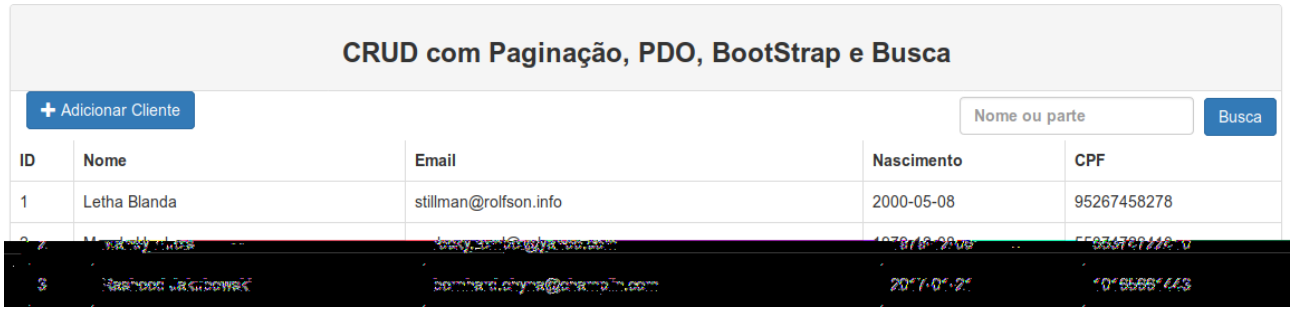
```
<div class="row">

<!-- create new register button -->
<div class="text-left col-md-2 top">
<a href="add.php" class="btn btn-primary pull-left">
```

```
<span class="glyphicon glyphicon-plus"></span> Adicionar Cliente
</a>
</div>

<!-- Form de busca-->
<div class="col-md-10">
```

Que gerou o seguinte botão:



Veja que mudei o botão da busca para a direita da caixa de texto.
Falta agora criar o arquivo add.php.

Adicionar os botões para Editar e Excluir

Mudei i final de fetch_data.php para:

```
   "<td>" . $row['cpf'] . "</td>";
            ?>
         <td><a href="update.php?id=<?=$row['id']?>"><i class="glyphicon glyphicon-edit"
title="Editar"></a></td>
         <td><a onclick="return confirm('Tem certeza de que deseja excluir este registro ?')"
href="delete.php?id=<?=$row['id']?>"><i class="glyphicon glyphicon-remove-circle"
title="Excluir"></a></td>
<?php
print "
   </tr>";
}
```

E resultou nisso:

A ação de exclusão é muito importante, portanto gosto de pedir confirmação, como mostrado.
Agora falta criar os arquivos update.php, delete.php e add.php.

Adicionar add.php

Este código abrirá o formulário para cadastrar um novo cliente e quando for clicado no submit ele adicionará os dados no banco de dados:

```
<?php
require_once('header.php');
require_once('functions.php');
?>

<div class="container">
   <div class="panel panel-default">
      <div class="panel-heading text-center"><h3><b><?=$title?> <br> Adicionar</h3></b></div>
      <div class="row">

      <div class="col-md-3"></div>
      <div class="col-md-6">

      <table class="table table-bordered table-responsive table-hover">
         <form method="post" action="add.php">
         <tr><td>Nome</td><td><input type="text" name="nome" autofocus></td></tr>
         <tr><td>E-mail</td><td><input type="text" name="email"></td></tr>
         <tr><td>Nascimento</td><td><input type="text" name="nascimento"></td></tr>
         <tr><td>CPF</td><td><input type="text" name="cpf"></td></tr>
         <tr><td></td><td><input class="btn btn-primary" name="enviar" type="submit"
value="Cadastrar">   
         <input class="btn btn-warning" name="enviar" type="button"
onclick="location='index.php'" value="Voltar"></td></tr>
         </form>
      </table>
      </div>
   </div>
```

```
</div>

<?php
require_once('db_connect.php');

if(isset($_POST['enviar'])){
$nome = $_POST['nome'];
$email = $_POST['email'];
$nascimento = $_POST['nascimento'];
$cpf = $_POST['cpf'];

  try{
     $stmte = $pdo->prepare("INSERT INTO clientes(nome,email,nascimento,cpf) VALUES
(?, ?, ?, ?)");
     $stmte->bindParam(1, $nome , PDO::PARAM_STR);
     $stmte->bindParam(2, $email , PDO::PARAM_STR);
     $stmte->bindParam(3, $nascimento , PDO::PARAM_STR);
     $stmte->bindParam(4, $cpf , PDO::PARAM_INT);
     $executa = $stmte->execute();

     if($executa){
        echo 'Dados inseridos com sucesso';
            header('location: index.php');
     }
     else{
        echo 'Erro ao inserir os dados';
     }
  }
  catch(PDOException $e){
    echo $e->getMessage();
  }
}
require_once('footer.php');
?>
```

Adicionar delete.php

```
<?php
require_once('header.php');
require_once('db_connect.php');

$id=$_GET['id'];

$sth = $pdo->prepare("SELECT id, nome,email,nascimento,cpf from clientes WHERE id = :id");
$sth->bindValue(':id', $id, PDO::PARAM_STR);
$sth->execute();

$reg = $sth->fetch(PDO::FETCH_OBJ);
$nome = $reg->nome;
$email = $reg->email;
$nascimento = $reg->nascimento;
$cpf = $reg->cpf;
```

```
require_once('header.php');
?>

<div class="container">
   <div class="panel panel-default">
      <div class="panel-heading text-center"><h3><b><?=$title?> <br> Excluir</h3></b></div>
      <div class="row">
      <div class="col-md-3"></div>
      <div class="col-md-6">
         <h3>Realmente excluir o registro abaixo?</h3>
         <br>
         <b>ID:</b> <?=$id?><br>
         <b>Nome:</b> <?=$nome?><br>
         <b>E-mail:</b> <?=$email?><br>
         <b>Nascimento:</b> <?=$nascimento?><br>
         <b>CPF:</b> <?=$cpf?><br>
         <br>
         <form method="post" action="">
         <input name="id" type="hidden" value="<?=$id?>">
         <input name="enviar" class="btn btn-danger" type="submit"
value="Excluir!">   
         <input name="enviar" class="btn btn-warning" type="button"
onclick="location='index.php'" value="Voltar">
         </form>
         <br><br><br>
      <?php require_once('footer.php'); ?>
      </div>
   <div>
</div>

<?php

if(isset($_POST['enviar'])){
$id = $_POST['id'];
   $sql = "DELETE FROM  $tabela WHERE id = :id";
   $sth = $pdo->prepare($sql);
   $sth->bindParam(':id', $id, PDO::PARAM_INT);
   if( $sth->execute()){
      print "<script>alert('Registro excluído com sucesso!');location='index.php';</script>";
   }else{
      print "Erro ao exclur o registro!<br><br>";
   }
}
?>
```

Adicionar update.php

```
<?php
require_once('header.php');
require_once('db_connect.php');
```

```
$id=$_GET['id'];

$sth = $pdo->prepare("SELECT id, nome,email,nascimento,cpf from clientes WHERE id = :id");
$sth->bindValue(':id', $id, PDO::PARAM_STR); // No select e no delete basta um bindValue
$sth->execute();

$reg = $sth->fetch(PDO::FETCH_OBJ);
$nome = $reg->nome;
$email = $reg->email;
$nascimento = $reg->nascimento;
$cpf = $reg->cpf;

require_once('header.php');
?>
<div class="container">
   <div class="panel panel-default">
      <div class="panel-heading text-center"><h3><b><?=$title?> <br>Atualizar</h3></b></div>
      <div class="row">
      <div class="col-md-3"></div>
      <div class="col-md-6">
         <form method="post" action="">
            <table class="table table-bordered table-responsive table-hover">
            <tr><td><b>Nome</td><td><input type="text" name="nome"
value="<?=$nome?>"></td></tr>
            <tr><td><b>E-mail</td><td><input type="text" name="email" value="<?=$email?>"></
td></tr>
            <tr><td><b>Nascimento</td><td><input type="text" name="nascimento" value="<?
=$nascimento?>"></td></tr>
            <tr><td><b>CPF</td><td><input type="text" name="cpf"
value="<?=$cpf?>"></td></tr>
            <input name="id" type="hidden" value="<?=$id?>">
            <tr><td></td><td><input name="enviar" class="btn btn-primary" type="submit"
value="Editar">       
            <input name="enviar" class="btn btn-warning" type="button"
onclick="location='index.php'" value="Voltar"></td></tr>
            </table>
         </form>
         <?php require_once('footer.php'); ?>
      </div>
   <div>
</div>

<?php

if(isset($_POST['enviar'])){
   $id = $_POST['id'];
   $nome = $_POST['nome'];
   $email = $_POST['email'];
   $nascimento = $_POST['nascimento'];
   $cpf = $_POST['cpf'];
```

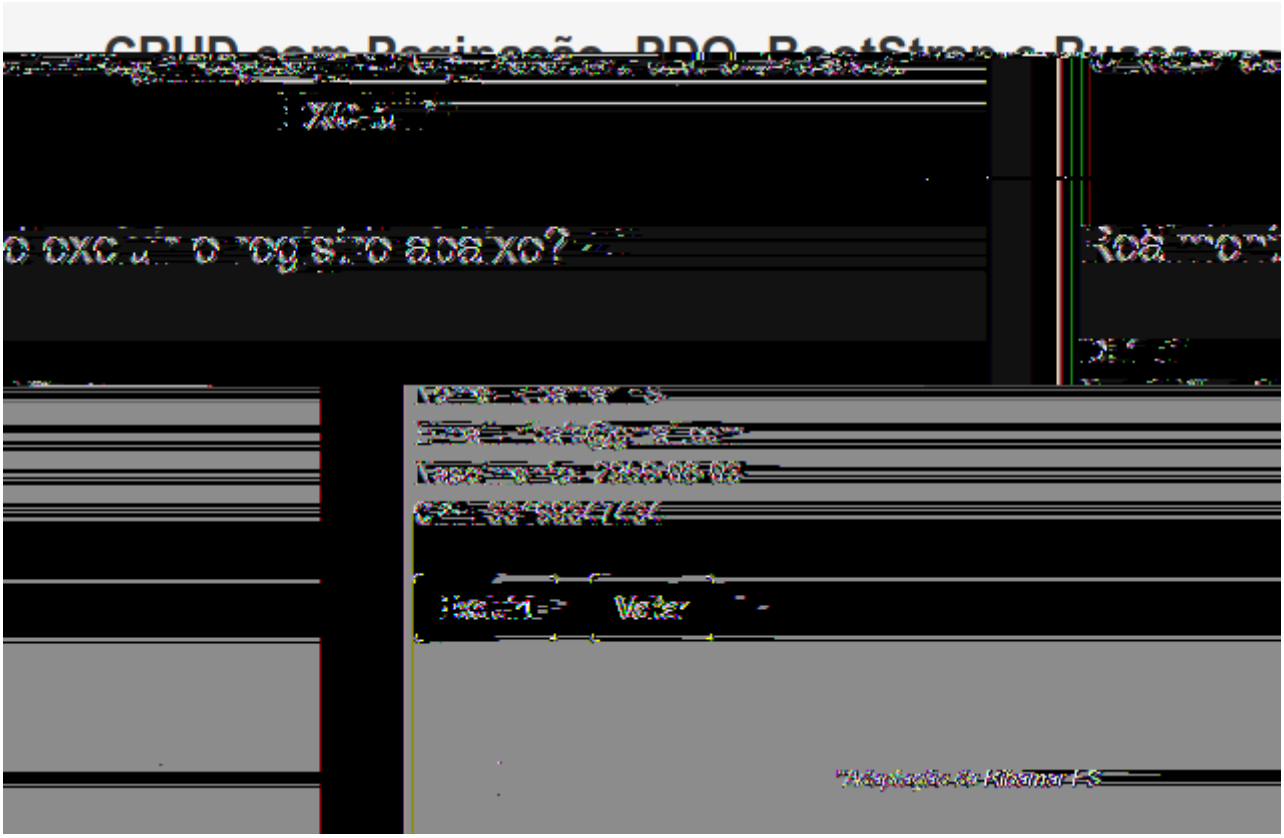
```
    $sql = "UPDATE $tabela SET nome = :nome, email=:email, nascimento=:nascimento, cpf=:cpf
WHERE id = :id";
    $sth = $pdo->prepare($sql);
    $sth->bindParam(':id', $_POST['id'], PDO::PARAM_INT);
    $sth->bindParam(':nome', $_POST['nome'], PDO::PARAM_STR);
    $sth->bindParam(':email', $_POST['email'], PDO::PARAM_STR);
    $sth->bindParam(':nascimento', $_POST['nascimento'], PDO::PARAM_STR);
    $sth->bindParam(':cpf', $_POST['cpf'], PDO::PARAM_INT);

   if($sth->execute()){
        print "<script>alert('Registro alterado com sucesso!');location='index.php';</script>";
    }else{
        print "Erro ao alterar  o registro!<br><br>";
    }
}
?>
```

Vejamos agora as novas telas:

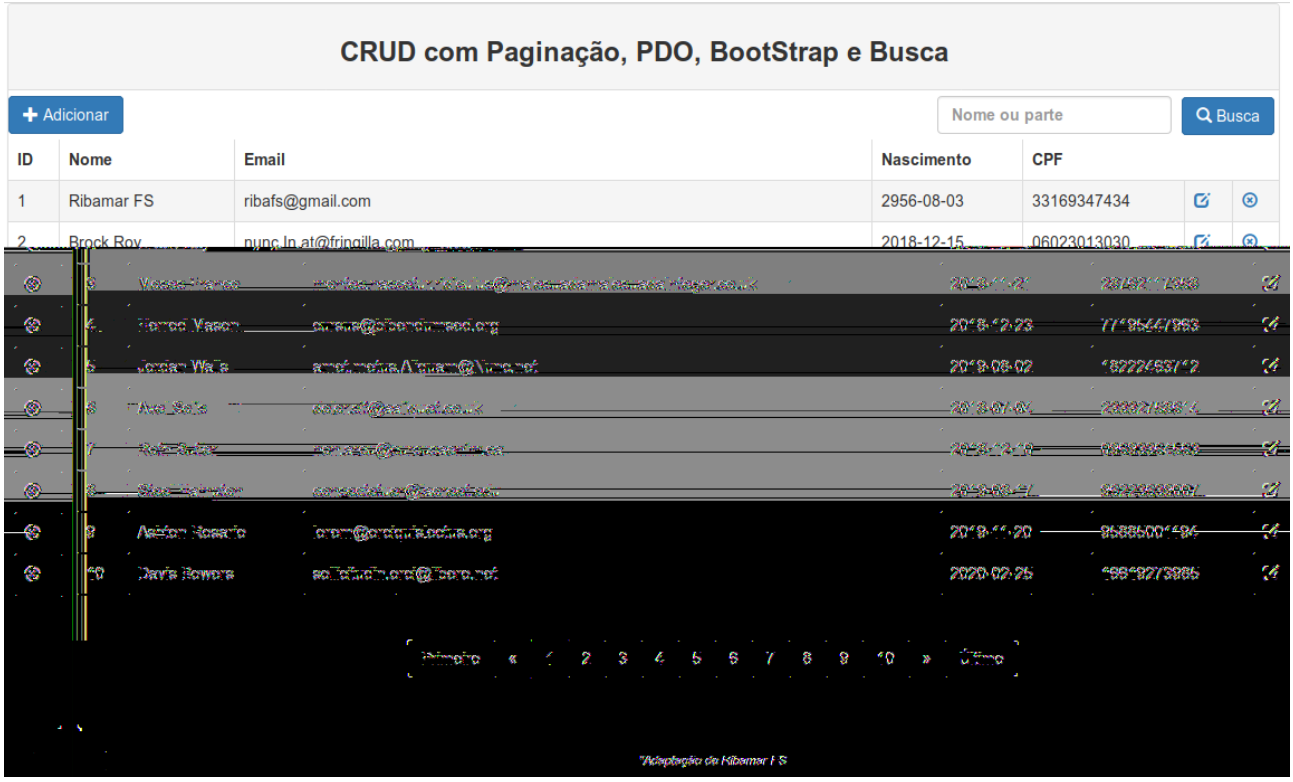
Adicionar

Excluir



Atualizar

Veja como está a tela agora



Testes

Aplicativo testado e funcionando normalmente no Linux e no Windows, com PHP 7.2 em ambos. Agora testarei com o PostgreSQL.

Criação de Funções

É bom lembrar que a criação de funções ajuda a não repetir código. Uma boa opção seria criar funções para o add, delete e update na parte de manipulação do banco.

Código Orientado a Objetos

Veja que até agora praticamente não usamos código orientado a objetos, mas sim estruturado.

Iremos criar uma versão deste aplicativo usando código orientado a objetos, com classes, propriedades e métodos.

Criação de um CRUD em PHPOO com bons recursos

Recursos:

- PDO
- Paginação usando o plugin do jquery bootpag
- Busca
- BootStrap
- Suporte ao MySQL e PostgreSQL testados

Testando exemplos de paginação encontrados via internet

Partindo deste exemplo de paginação:
https://www.kodingmadesimple.com/2017/01/simple-ajax-pagination-in-jquery-php-pdo-mysql.html

Usarei o CRUD criado anteriormente como ponto de partida.

Começarei criando uma classe para conexão ao banco de dados

Criarei uma pasta para as classes

classes/connection.php

Aqui o código da classe Connection():

```
<?php

class Connection
{
      private $host = 'localhost';
      private $db   = 'crud_phpoo';
      private $user = 'root';
      private $pass = 'root';
      public  $sgbd = 'mysql';// Testados: mysql, pgsql e sqlite
      public  $pdo;
      private $port = 3306; // pgsql - 5432

      public $regsPerPage = 10; // Registros por página
   public $linksPerPage = 15;
   public $appName = 'Cadastro de Clientes';

   public function __construct(){

          switch ($this->sgbd){
               case 'mysql':
                    try {
                         $dsn = $this->sgbd.':host='.$this->host.';dbname='.$this->db.';port='.$this-
>port;
                         $this->pdo = new PDO($dsn, $this->user, $this->pass);
                         // Boa exibição de erros
                         $this->pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES,false);
```

```
                $this->pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

                $this->pdo->query('SET NAMES utf8');
                return $this->pdo;

            }catch(PDOException $e){
            // Usar estas linhas no catch apenas em ambiente de testes/desenvolvimento. Em produção apenas o exit()
                echo '<br><br><b>Código</b>: '.$e->getCode().'<hr><br>';
                echo '<b>Mensagem</b>: '. $e->getMessage().'<br>';
                echo '<b>Arquivo</b>: '.$e->getFile().'<br>';
                echo '<b>Linha</b>: '.$e->getLine().'<br>';
                exit();
            }
            break;

        case 'pgsql':
            try {
                $dsn = $this->sgbd.':host='.$this->host.';dbname='.$this->db.';port='.$this->port;
                $this->pdo = new PDO($dsn, $this->user, $this->pass);

                // Boa exibição de erros
                $this->pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES,false);
                $this->pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

                return $this->pdo;

            }catch(PDOException $e){
                echo '<br><br><b>Código</b>: '.$e->getCode().'<hr><br>';
                echo '<b>Mensagem</b>: '. $e->getMessage().'<br>';
                echo '<b>Arquivo</b>: '.$e->getFile().'<br>';
                echo '<b>Linha</b>: '.$e->getLine().'<br>';
                exit();
            }
            break;

        case 'sqlite':
            try {
                $this->pdo = new PDO('sqlite:/home/ribafs/estoque.db'); // Caminho do banco em sqlite

                // Boa exibição de erros
                $this->pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES,false);
                $this->pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

                return $this->pdo;

            }catch(PDOException $e){
```

```
                    echo '<br><br><b>Código</b>: '.$e->getCode().'<hr><br>';
                    echo '<b>Mensagem</b>: '. $e->getMessage().'<br>';
                    echo '<b>Arquivo</b>: '.$e->getFile().'<br>';
                    echo '<b>Linha</b>: '.$e->getLine().'<br>';
                    exit();
                }
                break;
            case 'default':
                break;
        }
    }
}
```

Código da classe Crud

```
<?php

require_once 'connection.php';
$conn = new Connection();
$pdo = $conn->pdo;

/* Classe que trabalha com um crud, lidando com uma tabela por vez, que é fornecida a cada
instância, desde a conexão com o banco */

class Crud extends Connection
{

    public $pdo;

    public function __construct($pdo){
        $this->pdo = $pdo;
    }

  public function insert(){
    $nome = $_POST['nome'];
    $email = $_POST['email'];
    $nascimento = $_POST['nascimento'];
    $cpf = $_POST['cpf'];

      try{
        $stmte = $this->pdo->prepare("INSERT INTO clientes(nome,email,nascimento,cpf)
VALUES (?, ?, ?, ?)");
        $stmte->bindParam(1, $nome , PDO::PARAM_STR);
        $stmte->bindParam(2, $email , PDO::PARAM_STR);
        $stmte->bindParam(3, $nascimento , PDO::PARAM_STR);
        $stmte->bindParam(4, $cpf , PDO::PARAM_INT);
        $executa = $stmte->execute();

        if($executa){
          //echo 'Dados inseridos com sucesso';
          return true;
```

```
            }else{
                //echo 'Erro ao inserir os dados';
                return false;
            }
        }
        catch(PDOException $e){
          echo $e->getMessage();
        }
    }

    public function delete($id){
        $sql = "DELETE FROM clientes WHERE id = :id";
        $sth = $this->pdo->prepare($sql);
        $sth->bindParam(':id', $id, PDO::PARAM_INT);

        if( $sth->execute()){
            //Registro excluído com sucesso!
            return true;
        }else{
            //Erro ao exclur o registro!
            return false;
        }

    }

    public function update(){
        $id = $_POST['id'];
        $nome = $_POST['nome'];
        $email = $_POST['email'];
        $nascimento = $_POST['nascimento'];
        $cpf = $_POST['cpf'];

        $sql = "UPDATE clientes SET nome = :nome, email = :email, nascimento = :nascimento, cpf =
:cpf WHERE id = :id";

        $sth = $this->pdo->prepare($sql);
        $sth->bindParam(':id', $_POST['id'], PDO::PARAM_INT);
        $sth->bindParam(':nome', $_POST['nome'], PDO::PARAM_STR);
        $sth->bindParam(':email', $_POST['email'], PDO::PARAM_STR);
        $sth->bindParam(':nascimento', $_POST['nascimento'], PDO::PARAM_STR);
        $sth->bindParam(':cpf', $_POST['cpf'], PDO::PARAM_INT);

       if($sth->execute()){
            //Registro alterado com sucesso!
            return true;
        }else{
            //Erro ao alterar o registro!
            return false;
        }

    }
}
```

Agora vejamos como ficaram os demais arquivos.

db_connection.php foi removido, pois a classe classes/connection.php o substituiu.

header.php ficou assim:

```
<?php
include_once("classes/crud.php");
$crud = new Crud($pdo);
// $conn para referir a classe Connection()

$stmt = $crud->pdo->prepare("SELECT COUNT(*) FROM clientes");
$stmt->execute();
$rows = $stmt->fetch();

// get total no. of pages
$totalPages = ceil($rows[0]/$conn->regsPerPage);
?>

<!DOCTYPE html>
<html lang="pt">
<head>
   <title><?=$conn->appName?></title>
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css" />
   <link href="css/style.css" rel="stylesheet" type="text/css" />
   <style type="text/css">
   .panel-footer {
      padding: 0;
      background: none;
   }
   </style>
</head>
```

add.php

```
<?php
require_once('header.php');
?>

<div class="container">
   <div class="panel panel-default">
      <div class="panel-heading text-center"><h3><b><?=$conn->appName?></b>
<br>(Adicionar)</h3></div>
      <div class="row">

      <div class="col-md-3"></div>
      <div class="col-md-6">

      <table class="table table-bordered table-responsive table-hover">
         <form method="post" action="add.php">
```

```
        <tr><td>Nome</td><td><input type="text" name="nome" autofocus></td></tr>
        <tr><td>E-mail</td><td><input type="text" name="email"></td></tr>
        <tr><td>Nascimento</td><td><input type="text" name="nascimento"></td></tr>
        <tr><td>CPF</td><td><input type="text" name="cpf"></td></tr>
        <tr><td></td><td><input class="btn btn-primary" name="enviar" type="submit"
value="Cadastrar">   
        <input class="btn btn-warning" name="enviar" type="button"
onclick="location='index.php'" value="Voltar"></td></tr>
        </form>
      </table>
      </div>
   </div>
</div>

<?php
require_once('classes/crud.php');
$crud = new Crud($pdo);

   if(isset($_POST['enviar'])){
      if($crud->insert()){
        // 'Dados inseridos com sucesso';
            header('location: index.php');
      }else{
        echo 'Erro ao inserir os dados';
        exit();
      }
   }

require_once('footer.php');
?>
```

delete.php

```
<?php
require_once('header.php');
require_once('classes/crud.php');
$crud = new Crud($pdo);

$id=$_GET['id'];

$sth = $crud->pdo->prepare("SELECT id, nome,email,nascimento,cpf from clientes WHERE id
= :id");
$sth->bindValue(':id', $id, PDO::PARAM_STR);
$sth->execute();

$reg = $sth->fetch(PDO::FETCH_OBJ);
$nome = $reg->nome;
$email = $reg->email;
$nascimento = $reg->nascimento;
$cpf = $reg->cpf;
```

```
require_once('header.php');
?>

<div class="container">
   <div class="panel panel-default">
      <div class="panel-heading text-center"><h3><b><?=$conn->appName?> <br>
Excluir</h3></b></div>
      <div class="row">
      <div class="col-md-3"></div>
      <div class="col-md-6">
         <h3>Realmente excluir o registro abaixo?</h3>
         <br>
         <b>ID:</b> <?=$id?><br>
         <b>Nome:</b> <?=$nome?><br>
         <b>E-mail:</b> <?=$email?><br>
         <b>Nascimento:</b> <?=$nascimento?><br>
         <b>CPF:</b> <?=$cpf?><br>
         <br>
         <form method="post" action="">
         <input name="id" type="hidden" value="<?=$id?>">
         <input name="enviar" class="btn btn-danger" type="submit"
value="Excluir!">   
         <input name="enviar" class="btn btn-warning" type="button"
onclick="location='index.php'" value="Voltar">
         </form>
         <br><br><br>
      <?php require_once('footer.php'); ?>
      </div>
   <div>
</div>

<?php

if(isset($_POST['enviar'])){
   $id = $_POST['id'];

   if( $crud->delete($id)){
      print "<script>alert('Registro excluído com sucesso!');location='index.php';</script>";
   }else{
      print "Erro ao exclur o registro!<br><br>";
      exit();
   }
}
?>
```

fetch_data.php

```
<?php
include_once("classes/crud.php");
$crud = new Crud($pdo);

if (isset($_POST["page"])) {
   $page_no = filter_var($_POST["page"], FILTER_SANITIZE_NUMBER_INT,
FILTER_FLAG_STRIP_HIGH);
   if(!is_numeric($page_no))
      die("Error fetching data! Invalid page number!!!");
} else {
   $page_no = 1;
}

// get record starting position
$start = (($page_no-1) * $conn->regsPerPage);

$results = $conn->pdo->prepare("SELECT * FROM clientes ORDER BY id LIMIT $start, $conn-
>regsPerPage");
$results->execute();

while($row = $results->fetch(PDO::FETCH_ASSOC)) {
   echo "<tr>" .
   "<td>" . $row['id'] . "</td>" .
   "<td>" . $row['nome'] . "</td>" .
   "<td>" . $row['email'] . "</td>" .
   "<td>" . $row['nascimento'] . "</td>" .
   "<td>" . $row['cpf'] . "</td>";
          ?>
        <td><a href="update.php?id=<?=$row['id']?>"><i class="glyphicon glyphicon-edit"
title="Editar"></a></td>
        <td><a href="delete.php?id=<?=$row['id']?>"><i class="glyphicon glyphicon-remove-
circle" title="Excluir"></a></td>
      <!-- onclick="return confirm('Tem certeza de que deseja excluir este registro ?')" -->
<?php
print "
   </tr>";
}
```

footer.php não mudou

Uma mudança geral foi que resolvi adotar a convenção para nomes de variáveis, camelCase e mudei $total_pages para $totalPages e todas seguindo esta convenção.

Resolvi também criar variáveis, melhor, propriedades na classe Connection para facilitar seu acesso:

```
    public $regsPerPage = 10;
    public $linksPerPage = 15;  // Esta era $max_visible
    public $appName = 'Cadastro de Clientes';
```

index.php – neste fiz apenas algumas pequenas alterações:

A variável $title mudei para $appName e como está em Connection a referência é assim:

```
$conn->appName
```

search.php – neste mudei apenas o começo para adaptar à classe crud:

```
<?php
require_once('./header.php');
require_once('classes/crud.php');
$crud = new Crud($pdo);

// Busca
if(isset($_GET['keyword'])){
    $keyword=$_GET['keyword'];

    $sql = "select * from clientes WHERE nome LIKE :keyword order by id";
    $sth = $crud->pdo->prepare($sql);
```

update.php

```
<?php
require_once('header.php');
require_once('classes/crud.php');
$crud = new Crud($pdo);

$id=$_GET['id'];

$sth = $crud->pdo->prepare("SELECT id, nome,email,nascimento,cpf from clientes WHERE id
= :id");
$sth->bindValue(':id', $id, PDO::PARAM_STR); // No select e no delete basta um bindValue
$sth->execute();

$reg = $sth->fetch(PDO::FETCH_OBJ);
$nome = $reg->nome;
$email = $reg->email;
$nascimento = $reg->nascimento;
$cpf = $reg->cpf;

require_once('header.php');
?>
```

```
<div class="container">
   <div class="panel panel-default">
      <div class="panel-heading text-center"><h3><b><?=$conn->appName?>
<br>Atualizar</h3></b></div>
      <div class="row">
      <div class="col-md-3"></div>
      <div class="col-md-6">
         <form method="post" action="">
            <table class="table table-bordered table-responsive table-hover">
            <tr><td><b>Nome</td><td><input type="text" name="nome"
value="<?=$nome?>"></td></tr>
            <tr><td><b>E-mail</td><td><input type="text" name="email" value="<?=$email?>"></
td></tr>
            <tr><td><b>Nascimento</td><td><input type="text" name="nascimento" value="<?
=$nascimento?>"></td></tr>
            <tr><td><b>CPF</td><td><input type="text" name="cpf"
value="<?=$cpf?>"></td></tr>
            <input name="id" type="hidden" value="<?=$id?>">
            <tr><td></td><td><input name="enviar" class="btn btn-primary" type="submit"
value="Editar">       
            <input name="enviar" class="btn btn-warning" type="button"
onclick="location='index.php'" value="Voltar"></td></tr>
            </table>
         </form>
         <?php require_once('footer.php'); ?>
      </div>
   <div>
</div>

<?php

if(isset($_POST['enviar'])){
require_once('classes/crud.php');
$crud = new Crud($pdo);

  if($crud->update()){
      print "<script>alert('Registro alterado com sucesso!');location='index.php';</script>";
   }else{
      print "Erro ao alterar o registro!<br><br>";
      exit();
   }
}
?>
```

Quanto às telas não mudou nada.

Cadastro de Clientes

+ Adicionar | Nome ou parte | Busca

ID	Nome	Email	Nascimento	CPF		
1	Ribamar FS	diam.Duis@amalesuada.org	2019-05-29	33169347434		
2	Brock Roy	nunc.In.at@fringilla.com	2018-12-15	06023013030		
3	Moses Franco	montes.nascetur.ridiculus@malesuadamalesuadaInteger.co.uk	2018-11-21	28762117358		
4	Herrod Mason	cursus@bibendumsed.org	2018-12-23	77195447963		
5	Jordan Walls	amet.metus.Aliquam@Nunc.net	2019-08-02	18222463712		
6	Axel Solis	dolor.sit@aaliquet.co.uk	2018-07-04	23332758614		
7	Roth Butler	non.arcu@cursusaenim.ca	2018-12-10	09693364563		
8	Steel Hampton	consectetuer@laoreet.edu	2019-08-17	96223639007		
9	Ashton Rosario	lorem@orciquislectus.org	2019-11-20	95885001194		
10	Davis Bowers	sollicitudin.orci@libero.net	2020-02-25	16619273985		

Primeiro « 1 2 3 4 5 6 7 8 9 10 » Último

*Adaptação de Ribamar FS

Criação de um Aplicativo em PHP com bons recursos

Recursos:

- PDO
- Paginação
- Busca
- BootStrap
- Suporte ao MySQL e PostgreSQL testados

Este aplicativo não usará orientação a objetos, com exceção do PDO. O foco ficará apenas na programação estruturada.

Partirei do CRUD criado anteriormente. Farei uma cópia do seu diretório para começar, como app_php. E ele terá a finalidade de gerenciar duas tabelas. Um exemplo com duas tabelas serve para adaptar para qualquer quantidade de tabelas.

Começarei criando a pasta
clientes

Moverei para clientes os arquivos: index.php, fetch_data.php, add.php, delete.php, search.php e update.php.

Então farei uma cópia de **clientes** com todos os seus arquivos para **produtos**

Veja que agora os arquivos da pasta clientes ou da produtos não mais encontrarão db_connect.php nem header.php, footer.php ou functions.php. Para que encotnrem precisaremos alterar seus respectivos paths.

Mas a situação atual é esta:

/clientes/
    add.php
    delete.php
    fetch_data.php
    index.php
    search.php
    update.php
/css
/fonts
/js
/produtos
    add.php
    delete.php
    fetch_data.php
    index.php
    search.php
    update.php
/db_connect.php
/footer.php
/functions.php
/header.php

Veja que os arquivos de clientes e de produtos precisarão voltar um nível para encontrar os includes.

Mudança nos includes

Veja que os includes que antes procuravam os arquivos no mesmo diretório que se encontravam, assim:

require_once('header.php');
require_once('functions.php');

Agora precisam procurar assim:

require_once('../header.php');
require_once('../functions.php');

E assim com todos eles, pois deixei os arquivos que eram usados por vários, como db_connect.php, header.php e footer.php no raiz do aplicativo.

Aplicar esta correção a todos os arquivos de clientes, inclusive para os includes de js no index.php. Depois removeremos produtos e copiaremos novamente clientes para produtos, já com os includes corrigidos, para evitar ter que fazer novamente, entendeu?

Quando fui testar abrindo clientes o css não foi encontrado, ficando tudo sem estilo.
O que aconteceu?
Quando fizemos o include do header.php em clientes/index.php o path do css ficou assim: css/bootstrap.min.css, ou seja, header.php procurava o bootstrap na mesma pasta em que se encontrava e agora ele estava um nível anes.
Solução. Criar dois headers.php, um para o raiz e outro para cada pasta, clientes e produtos. Além disso alterar o path do bootstrap para que procure um nível anes no header.php interno: ../css/bootstrap.min.css

Agora ele encontrou o css.

Criação do index.php do raiz/menu

Algo muito simples, apenas como exemplo:

```
<?php
require_once('./header.php');
require_once('./functions.php');
?>

<div class="container">
   <div class="panel panel-default">
      <div class="panel-heading text-center"><h3><b><?=$title?></b></h3>
      <div class="row">
<pre>
```

</pre>
Clientes   Produtos

<pre>

</pre>

<?php require_once('./footer.php');?>

Veja como ficou:

Vamos agora adaptar o código que está na pasta produtos para a tabela produtos:

produtos:
id
descricao
unidade
data_cadastro

Trocar todas as ocorrências de nomes de campos e de tabela pelos acima.

Vamos agora adicionar um link ao título para voltar ao menu inicial.

Para isso editemos a variável title em functions.php e deixemos assim:

$title = 'CRUD com Paginação, PDO, BootStrap e Busca';

Mas não funcionará no index.php do raiz. Para isso vamos criar um código condicional.

Uma forma simples de resolver isso é criar uma variável $titleMenu exclusiva para o menu.

Finalmente a ela principal de produtos ficou assim:

Projeto de Aplicativo para gerenciar duas tabelas com PHP

Este aplicativo usará os recursos:

- PDO
- Paginação com o plugin do jQuery bootpag
- Busca
- BootStrap
- Suporte ao MySQL e PostgreSQL testados

Mas será construído inteiramente com PHP orientado a objetos

Este aplicativo usará duas tabelas a serem gerenciadas, com dois CRUDs respectivos e um menu inicial. Ele partirá do aplicativo CRUD com PHPOO.

Os procedimentos serão muito semelhantes aos adotados na criação do Aplicativo em PHP procedural com duas tabelas.

Começarei criando uma pasta
clientes

E movendo para ela os arquivos: add.php, delete.php, fetch_data.php, index.php, search.php e update.php

Copiarei também header.php para clientes.

Agora corrigirei o path dos includes.

require_once('classes/crud.php');

para
require_once('../classes/crud.php');

Exceção para header.php que fica no mesmo diretório.

Agora farei as adaptações na pasta produtos para a tabela:

produtos:
id
descricao
unidade
data_cadastro

Criar classe Cliente

Como a classe criada anteriormente chamava-se Crud, mudei seu nome para Cliente, pois teremos também uma para produtos. Após mudar seu nome corrigi as referências de crud para cliente.

Criar a classe Produto

Copiar a classe cliente para produto e adaptar para o nome da tabela e os campos de produtos.

Adicionar os botões Editar e Excluir em cada registro retornado pela Busca.

Veja a Busca

E a tela de produtos

Por Fazer/Todo

Algo importante para implementar agora seria:

- Melhorar a segurança cuidando de filtrar todos os formulários com os filtros do PHP

Vide:
https://www.w3schools.com/PHP/php_filter.asp
https://secure.php.net/manual/en/filter.filters.php
https://secure.php.net/manual/pt_BR/filter.constants.php

- Implementar nos formulários os novos recursos do HTML5. Vide os forms de exemplo do curso de HTML.

- Outro recurso é o uso de expressões regulares. Ver o curso de PHP estruturado.

Colaboração de *Ribamar FS – http://ribafs.org*