

Sistema CRUD com Login e Session em PHP
Prof Anderson Vanin - 2018

1. INTRODUÇÃO

Iremos partir do princípio de que já temos o banco de dados chamado **sistema** criado contendo as seguintes tabelas:

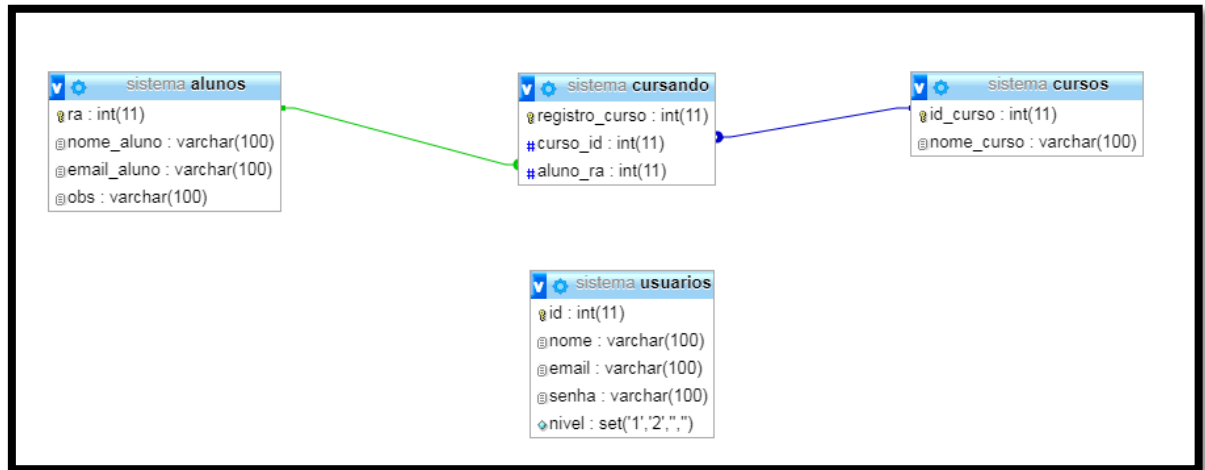


Figura 1 - Relações do Banco de Dados Sistema

Caso você não saiba como criar um banco de dados, sugiro a seguinte [playlist](#) que ensina a modelar e criar um banco de dados.

2. ESTRUTURA DAS TABELAS:

a) Tabela Alunos

Field	Type	Null	Key	Default	Extra
id_ra	int(11)	NO	PRI	NULL	auto_increment
nome_aluno	varchar(100)	NO		NULL	
email_aluno	varchar(100)	NO		NULL	
obs	varchar(100)	NO		NULL	

Figura 2 - Estrutura da tabela Alunos

b) Tabela Cursos

Field	Type	Null	Key	Default	Extra
id_curso	int(11)	NO	PRI	NULL	auto_increment
nome_curso	varchar(100)	NO		NULL	

Figura 3 - Estrutura da tabela Cursos

c) Tabela Cursando

Field	Type	Null	Key	Default	Extra
registro_curso	int(11)	NO	PRI	NULL	auto_increment
curso_id	int(11)	NO	MUL	NULL	
aluno_ra	int(11)	NO	MUL	NULL	

Figura 4 - Estrutura da tabela Cursando

d) Tabela Usuários

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
nome	varchar(100)	NO		NULL	
email	varchar(100)	NO		NULL	
senha	varchar(100)	NO		NULL	
nivel	set('1','2','3','4')	NO		NULL	

Figura 5 - Estrutura da tabela Usuários

3. CONFIGURAÇÕES INICIAIS DOS ARQUIVOS DO TEMPLATE NO SERVIDOR LOCAL

Para esse modelo, utilizaremos um template criado em bootstrap chamado **Initializr**, que pode ser obtido gratuitamente neste [link](#).



Figura 6 - Página de Download do template inicial – Initializr

Selecionando a opção Bootstrap, nos é fornecido mais algumas configurações que podem ser utilizadas. Caso não precise de mais configurações, basta clicar em **Download It** e download irá ser iniciado.

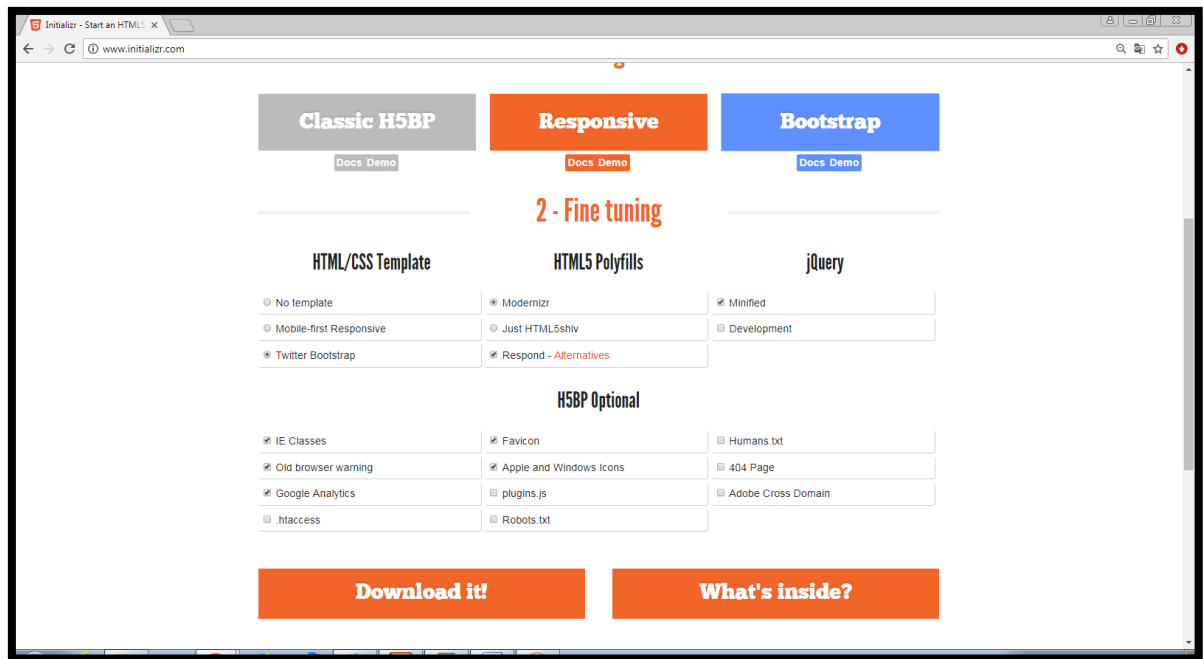


Figura 7 - Configurações adicionais e download do template

Após o download, faça a extração dos arquivos em uma pasta chamada **exemplo**, dentro de seu servidor local.

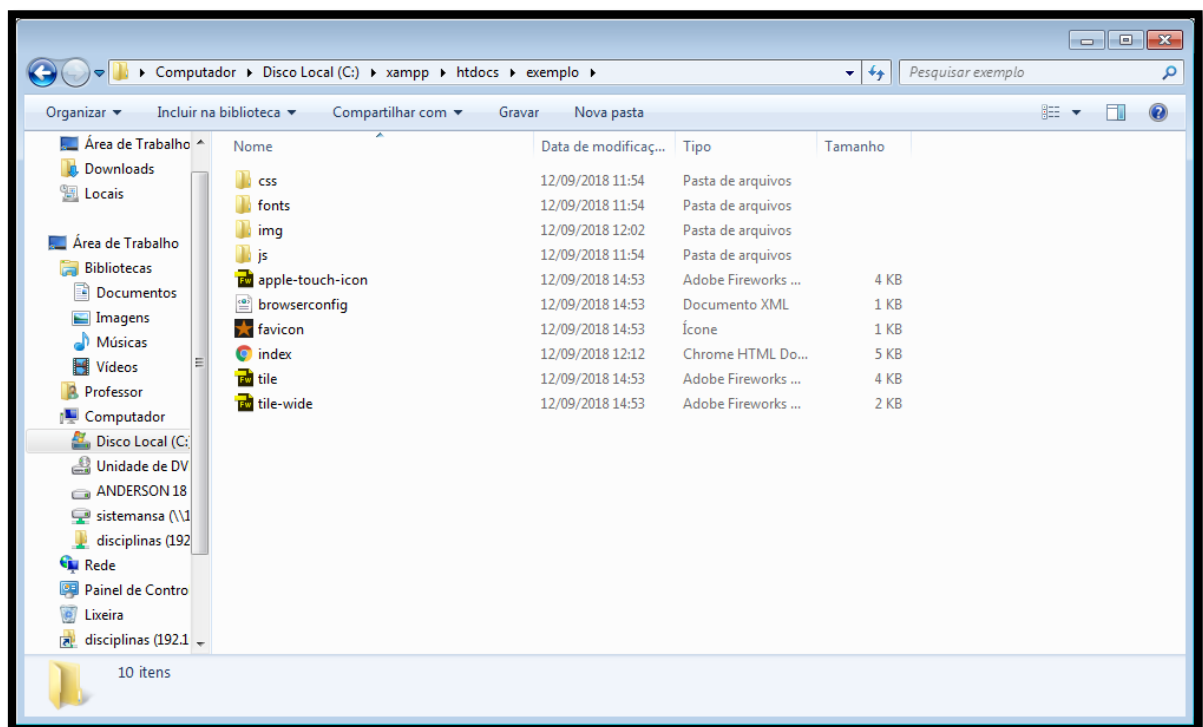


Figura 8 - Extração dos arquivos iniciais do template

4. CONSTRUÇÃO DAS TELAS DE VISUALIZAÇÃO DAS INFORMAÇÕES

Antes de começar a programação propriamente dita, uma boa dica é já criar as telas que irão receber os dados vindos das consultas no banco de dados, para depois fazermos as alterações necessárias.

A partir deste ponto, iremos salvar os arquivos criados todos com a extensão **.php**.

Partimos do princípio que o primeiro a usar o sistema será o ADMINISTRADOR, cujo qual terá permissões totais dentro do sistema, podendo Alterar, Incluir, Selecionar e Excluir os dados das tabelas.

4.1. Tela que exibe todos os alunos

A primeira tela será a que irá exibir todos os alunos que estão cadastrados no sistema.

A criação desta e das demais telas, pode seguir o mesmo que foi feito para o arquivo inicial do sistema, onde primeiramente baixamos um template inicial.

Deixo aqui uma outra sugestão, onde pode ser baixado um template de área administrativa neste [link](https://github.com/puikinh/sufee-admin-dashboard).

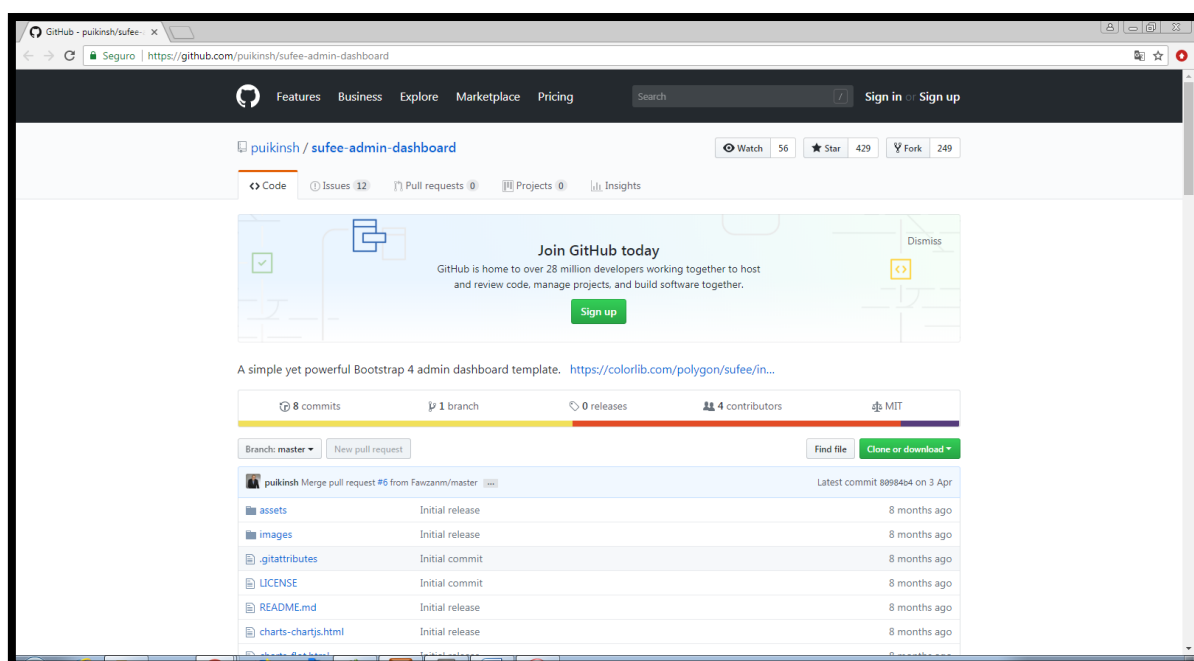


Figura 9 - Template para Área Administrativa do Site

Para baixar, basta clicar em Clone or Download e extrair os arquivos dentro de um novo diretório da sua pasta exemplo chamado restrito.

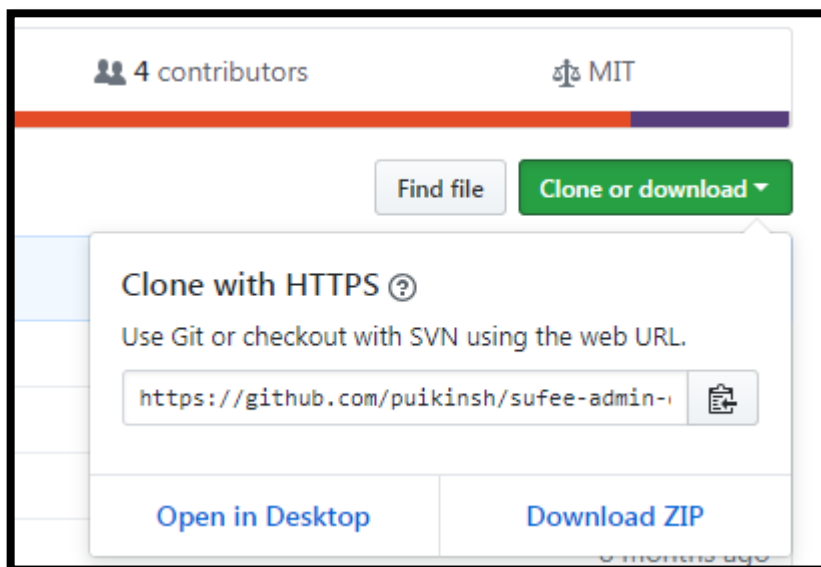


Figura 10 - Download do template de área administrativa

A estrutura de arquivos depois de descompactados deve estar assim:

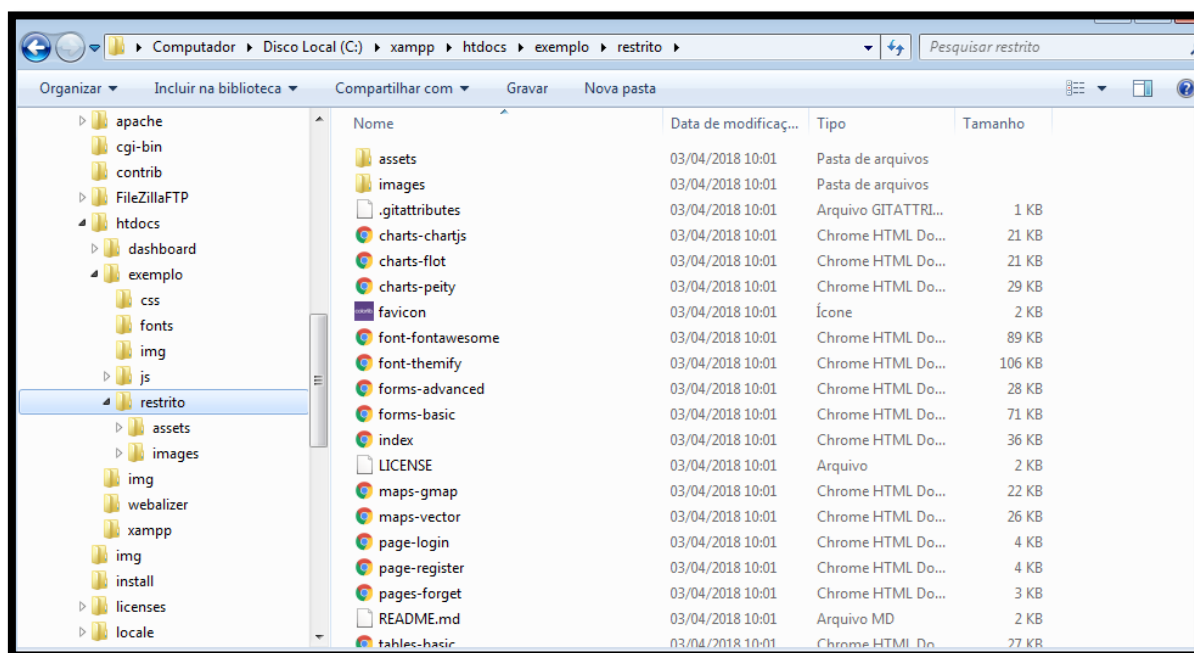


Figura 11 - Estrutura de arquivos template administrativo

Note que a página *index* dessa estrutura é um arquivo *html*. Altere a extensão para *index.php*.

O template vem com várias telas que não são necessárias ao nosso exemplo. Elimine tudo que não irá utilizar deixando somente os arquivos necessários para o

sistema **CRUD** que serão as telas de Atualização e Seleção (Ambos os usuários), Inserção e Exclusão (somente administrador).

Nesse momento você deve imaginar duas áreas administrativas distintas no sistema: uma para o Administrador Geral que tem todos os privilégios e outra para o Aluno que só pode visualizar seus próprios dados.

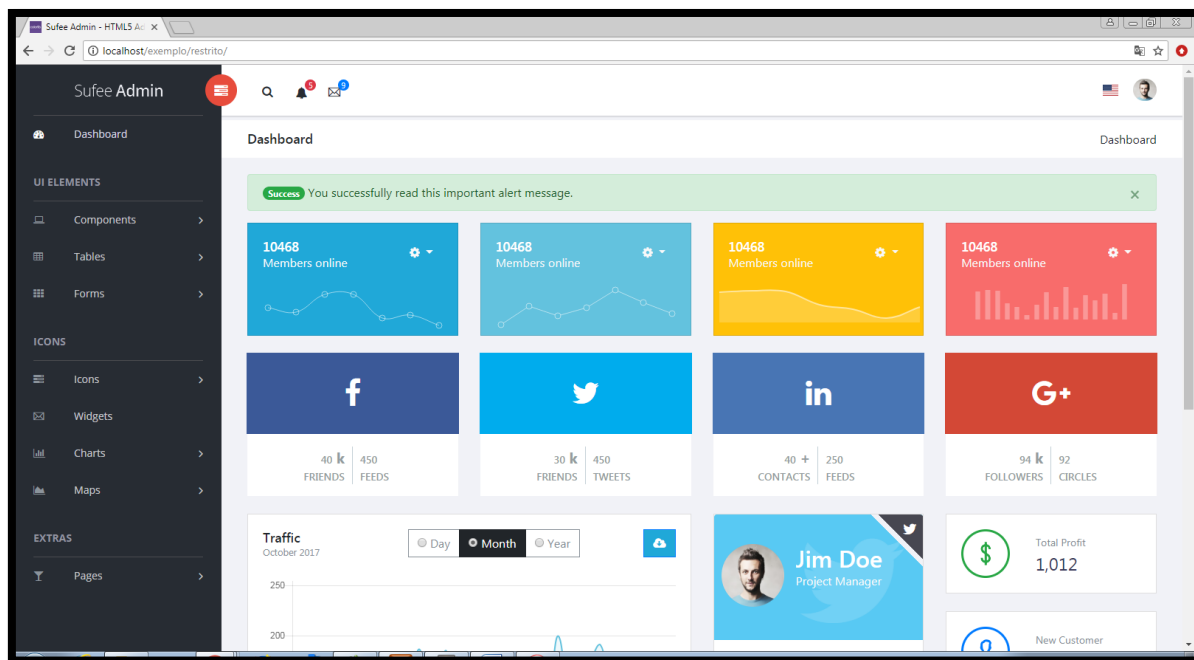


Figura 12 - Template de Área Administrativa

Crie todas as áreas que o Administrador irá ter acesso (uma página para cada área do CRUD).

4.2. Tela Inicial do Administrador

O administrador do sistema será o responsável por ter todos os privilégios como: selecionar, alterar, apagar e inserir novos registros, o que chamamos de **CRUD (Create, Read, Update, Delete)**.

Utilizaremos o *template* visto anteriormente e criaremos uma tela para cada privilégio do administrador.

Em nosso banco de dados temos 4 tabelas: **alunos**, **cursando**, **cursos** e **usuários**. Com base nessas tabelas vamos criar uma página para cada uma delas para permitir a visualização de todas as informações de cada uma das tabelas (uma página **php** para cada uma).

No sistema em questão, optamos por visualizar na página inicial da área restrita, uma lista de alunos cadastrados.

Alterando o html do template deixaremos a visualização da seguinte forma:

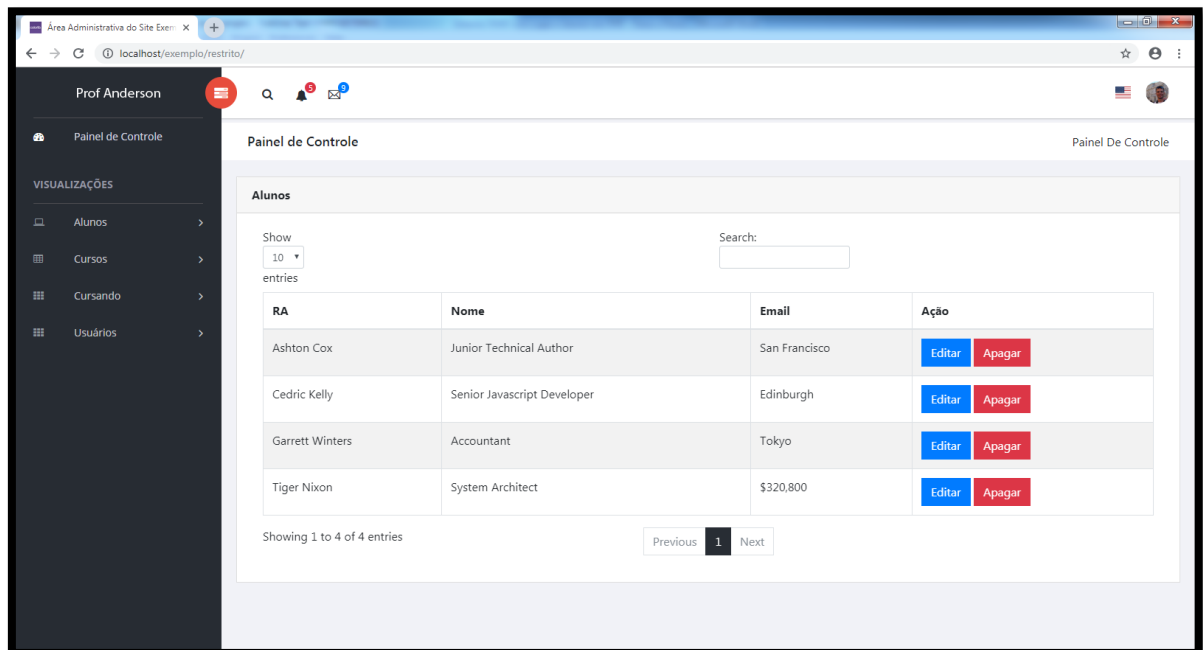


Figura 13 - Preparação do layout

Antes de iniciar o preparo da tela de login, iremos fazer com que o sistema se conecte ao banco de dados e exiba as informações dos alunos.

Iremos criar agora a página de conexão ao banco de dados.

4.3. Página conecta.php

```
1  <?php
2
3  $servidor = 'localhost';
4  $usuario = 'root';
5  $senha = '';
6  $banco = 'sistema';
7
8  // Conecta-se ao banco de dados MySQL
9  $conexao = new mysqli($servidor, $usuario, $senha, $banco);
10
11 // Caso algo tenha dado errado, exibe uma mensagem de erro
12 if (mysqli_connect_errno()){
13
14     echo 'ERRO DE CONEXÃO!';}
15
16 ;
17 ?>
```

Figura 14 - Arquivo conecta.php

4.4. Exibindo os dados da tabela alunos na página index.php da área restrito.

Na página **index.php** precisaremos fazer um *include* do arquivo de conexão ao banco de dados e criar um laço de repetição para que a tabela seja montada de acordo com a quantidade de dados que existem na tabela alunos.

```
1 <?php
2     include 'conecta.php';
3 >
4 <!doctype html>
5 <!--[if lt IE 7]>     <html class="no-js lt-ie9 lt-ie8 lt-ie7" lang=""> <![endif]-->
6 <!--[if IE 7]>       <html class="no-js lt-ie9 lt-ie8" lang=""> <![endif]-->
7 <!--[if IE 8]>       <html class="no-js lt-ie9" lang=""> <![endif]-->
8 <!--[if gt IE 8]><!--> <html class="no-js" lang="pt-br"> <!--<![endif]-->
9 <head>
10     <meta charset="utf-8">
11     <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

Figura 15 - Incluindo arquivo de conexão

Nas primeiras linhas de código da página **index.php** (linhas 1 a 3), adicionamos um trecho de código que irá chamar a página **conecta.php** e realizar a conexão com o banco de dados.

Agora precisamos que os dados sejam exibidos na tabela. Para isso, logo após a construção da tabela em *html*, iremos abrir um trecho de código em *php* para que seja feita a consulta no banco de dados e que em cada linha da tabela seja preenchido com os valores que ali existam.

```

238     </div>
239     <div class="card-body">
240     <table id="bootstrap-data-table" class="table table-striped table-bordered">
241     <thead>
242     <tr>
243     <th>RA</th>
244     <th>Nome</th>
245     <th>Email</th>
246     <th>Ação</th>
247     </tr>
248     </thead>
249     <tbody>
250     <?php
251         // Executa uma consulta que pega todos os registros
252         $sql = "SELECT * FROM ALUNOS";
253         $consulta = $conexao->query($sql);
254         while ($dados = $consulta->fetch_assoc()) {
255
256             <tr>
257                 <td><?php echo $dados['ra'];?></td>
258                 <td><?php echo $dados['nome_aluno'];?></td>
259                 <td><?php echo $dados['email_aluno'];?></td>
260                 <td>
261                     <button type="button" class="btn btn-primary">Editar</button>
262                     <button type="button" class="btn btn-danger">Apagar</button>
263                 </td>
264             </tr>
265             <?php
266                 }
267
268             ?>
269
270
271         </tbody>
272     </table>

```

Figura 16 - Alterando o html para a exibição da consulta em php

Observe que após a linha 249 (após a construção do cabeçalho da tabela) abrimos php para que seja feita a consulta à tabela alunos e após a linha 254 encerramos a instrução php, porém sem fechar a cláusula do laço de repetição while, pois queremos que os resultados sejam inseridos em cada uma das linhas da tabela. Assim em cada linha da tabela (que no nosso caso é uma <td>), abrimos php e colocamos a instrução relativa à um campo da tabela do banco de dados (conforme mostrado nas linhas 257 a 259).

Após o término da tabela fechamos as instruções do php (conforme mostrado nas linhas 265 a 267).

Executando nosso código teremos:

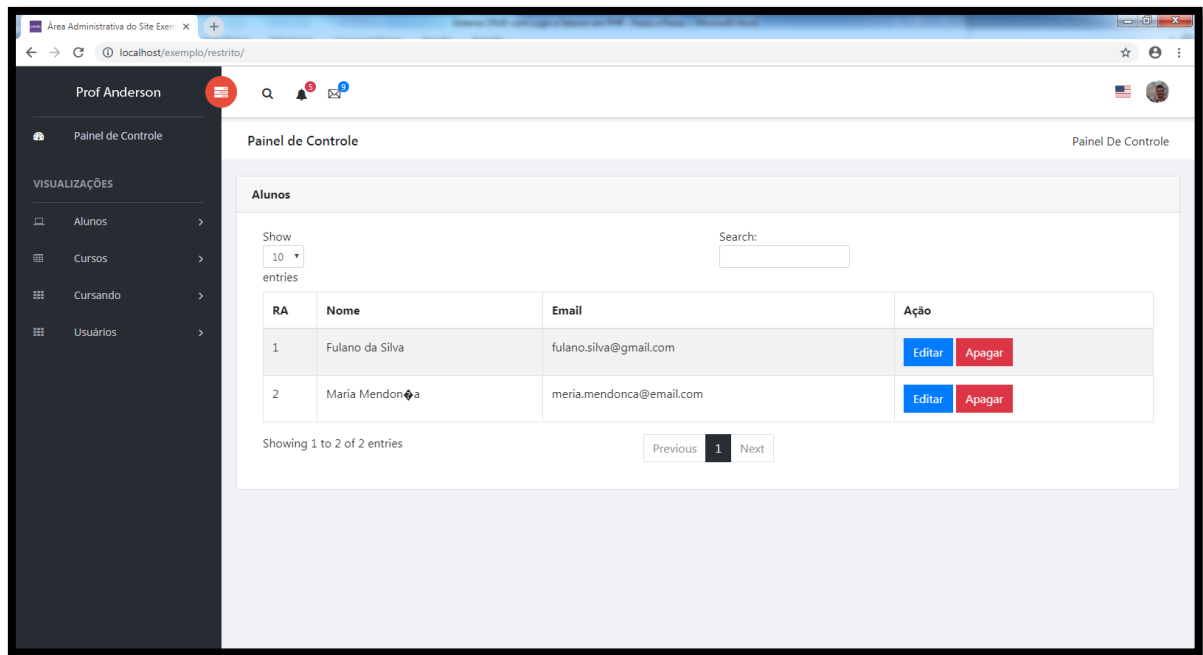


Figura 17 - Resultado da consulta à tabela Alunos

Observe que há alguns erros de acentuação. Vamos corrigir estes erros executando alguns parâmetros tanto no arquivo de configuração da conexão (**conecta.php**) como no banco de dados.

```

1  <?php
2
3  $servidor = 'localhost';
4  $usuario = 'root';
5  $senha = '';
6  $banco = 'sistema';
7
8  // Conecta-se ao banco de dados MySQL
9  $conexao = new mysqli($servidor, $usuario, $senha, $banco);
10
11
12  $sql= "SET NAMES 'utf8'";
13  mysqli_query($conexao, $sql);
14  $sql = 'SET character_set_connection=utf8';
15  mysqli_query($conexao, $sql);
16  $sql = 'SET character_set_client=utf8';
17  mysqli_query($conexao, $sql);
18  $sql = 'SET character_set_results=utf8';
19  mysqli_query($conexao, $sql);
20
21
22
23
24  // Caso algo tenha dado errado, exibe uma mensagem de erro
25  if (mysqli_connect_errno()){
26
27      echo 'ERRO DE CONEXÃO!';}
28
29  ;
30  ?>

```

Figura 18 - Alteração do arquivo conecta.php para exibição de caracteres UTF-8

Essa alteração no arquivo **conecta.php**, garante a correta exibição de caracteres UTF-8 vindos do banco de dados. Entretanto, é necessário que o banco de dados também está apto a receber tal codificação de caracteres. Para resolver isso vamos até o banco de dados e executar uma instrução que irá definir que o banco de dados também irá trabalhar utilizando a mesma codificação.

Abra a tela do phpmyadmin e altere a codificação do banco de dados utilizando o comando:

```

ALTER DATABASE nomedobanco DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci

```

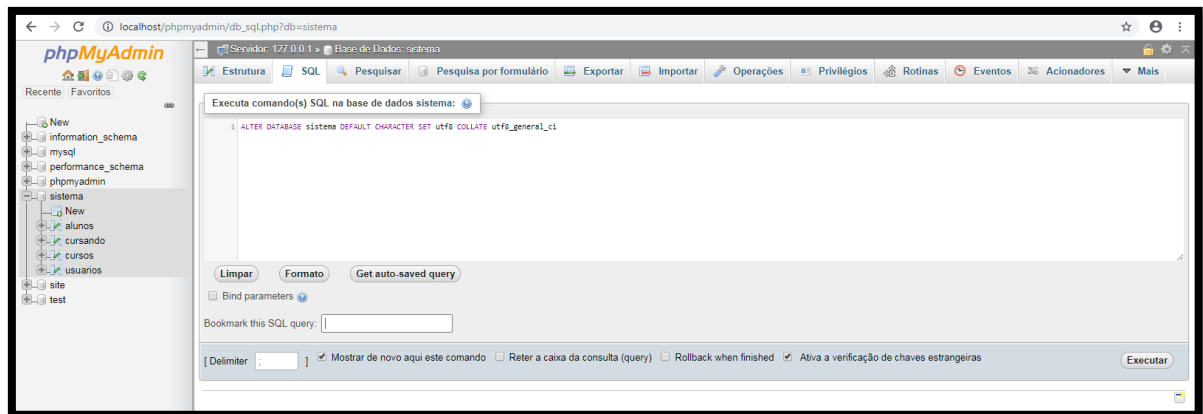


Figura 19 - Mudando a codificação do Banco de Dados para UTF-8

Clique em Executar para a instrução ser executada.

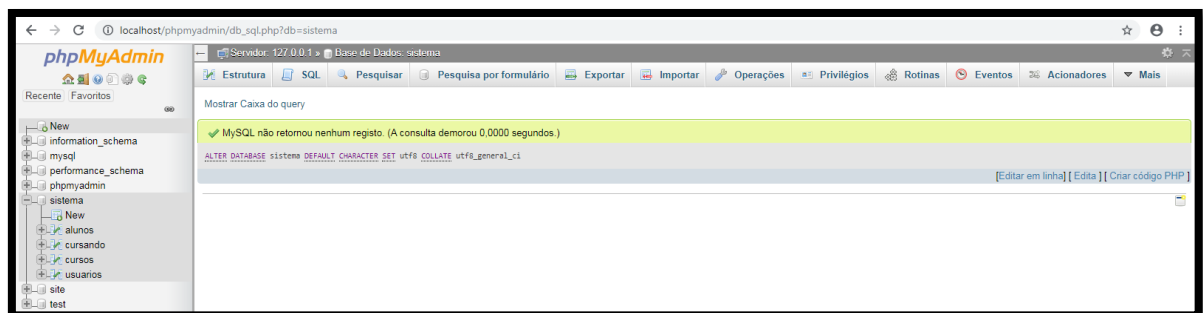


Figura 20 - Confirmação de alteração da codificação

Pronto! Agora o sistema está apto a fornecer e receber informações codificadas em UTF-8.

Podemos agora prosseguir com o restante da programação em php para fazer as telas de inserção, atualização e remoção de registros do banco de dados.

4.5. Inserindo novos registros

Para inserir novos registros, iremos inicialmente criar o formulário de cadastro de novos alunos. Vamos criar o arquivo que irá ter um formulário (***form_cadastra_alunos.php***) que irá passar os dados para uma pagina chamada ***cadastra_aluno.php*** que irá executar a consulta no banco de dados e em seguida retornar para a página principal da área restrita.

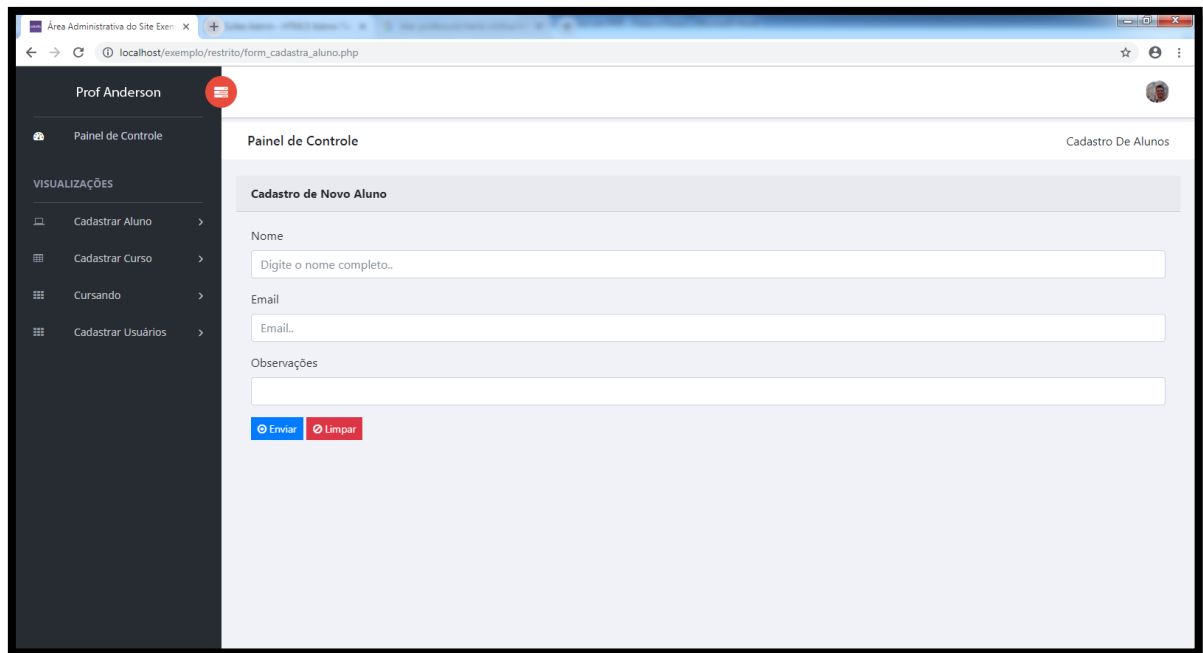


Figura 21 - página form_cadastra_aluno.php

A seguir o trecho do código da página form_cadastra_aluno.php que será responsável por coletar as informações.

```
<form action="cadastra_aluno.php" method="post" class="">
  <div class="form-group"><label for="nome" class=" form-control-label">Nome</label><input type="text" id="nome" name="nome"
  placeholder="Digite o nome completo.." class="form-control"></div>
  <div class="form-group"><label for="email" class=" form-control-label">Email</label><input type="email" id="email" name="email"
  placeholder="Email.." class="form-control"></div>
  <div class="form-group"><label for="obs" class=" form-control-label">Observações</label><input type="text" id="obs" name="obs"
  class="form-control"></div>
  <button type="submit" class="btn btn-primary btn-sm">
    <i class="fa fa-dot-circle-o"></i> Enviar
  </button>
  <button type="reset" class="btn btn-danger btn-sm">
    <i class="fa fa-ban"></i> Limpar
  </button>
</form>
```

Figura 22 - Trecho do formulário que coleta as informações a serem inseridas

Abaixo o código da página ***cadastra_aluno.php*** que irá processar os dados vindos deste formulário.

```
1 <?php
2   include 'conecta.php';
3
4   $nome_aluno = $_REQUEST['nome'];
5   $email_aluno = $_REQUEST['email'];
6   $obs = $_REQUEST['obs'];
7
8
9   $consulta = "INSERT INTO ALUNOS (nome_aluno,email_aluno,obs) VALUES ('$nome_aluno','$email_aluno','$obs')";
10  $conexao->query($consulta);
11
12  header('Location: index.php');
13
14  ?>
```

Figura 23 - Página cadastra_aluno.php

4.6. Atualizando e Apagando informações no Banco de Dados

Na pagina inicial do nosso sistema, já deixamos propositalmente dois botões: Editar e Apagar. Vamos começar programando o botão Editar para que quando o usuário clicar neste botão, o RA, NOME e EMAIL deste aluno sejam enviados à uma outra página (formulário) onde estas informações específicas possam ser editadas e/ou atualizadas e salvas novamente no banco de dados.

RA	Nome	Email	Ação
1	Fulano da Silva	fulano.silva@gmail.com	Editar Apagar
2	Maria Mendonça	meria.mendonca@email.com	Editar Apagar
4	Daniel Gonçalves Abraão	daniel@email.com	Editar Apagar

Figura 24 - Botões EDITAR e APAGAR da página index.php

4.6.1. Alterando o código dos botões EDITAR e APAGAR

Altere o código dos botões Alterar e Apagar:

```
<tr>
  <td><?php echo $dados['ra'];?></td>
  <td><?php echo $dados['nome_aluno'];?></td>
  <td><?php echo $dados['email_aluno'];?></td>
  <td>
    <?php
      echo "<a href='form_atualiza_aluno.php?ra=".$dados["ra"]."' class='btn btn-primary btn-sm'>Editar</a>";
      echo "<a href='apaga_aluno.php?ra=".$dados["ra"]."' class='btn btn-danger btn-sm'>Apagar</a>";
    ?>
  </td>
</tr>
```

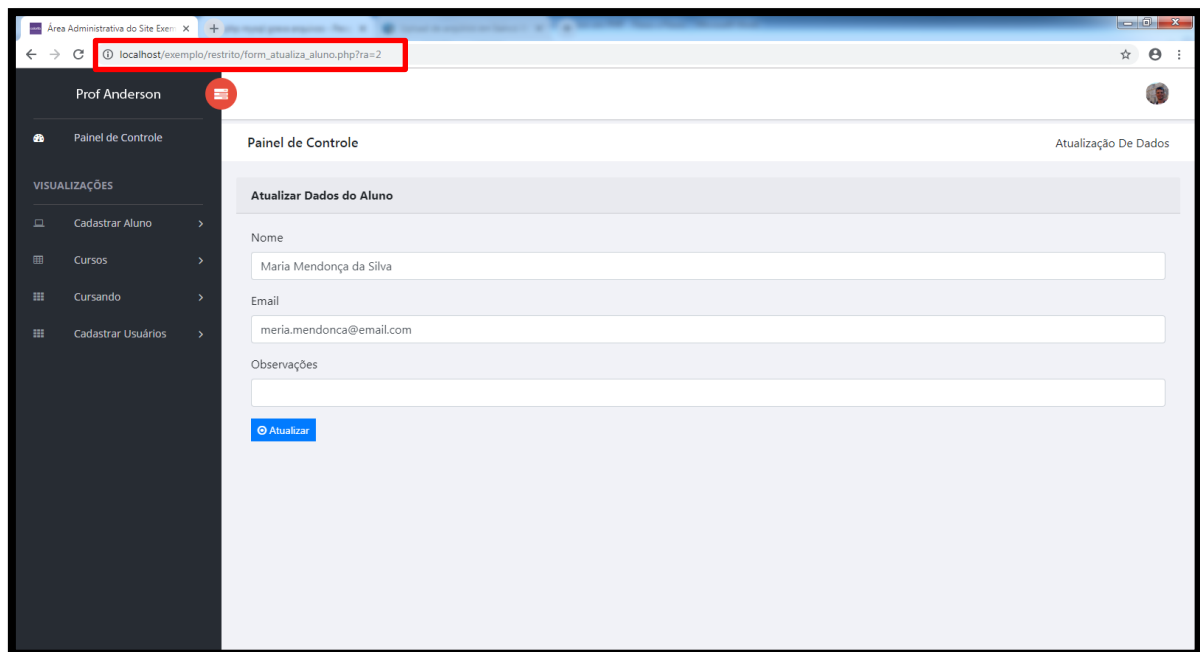
Figura 25 - Código dos botões Alterar e Apagar

Essa alteração faz com que a página **form_atualiza_aluno.php** seja chamada passando por *url* o *ra* do aluno cujo o qual se deseja executar a atualização. A mesma ação é efetuada para o botão Apagar passando via *url* o *ra* do aluno e chamando a página **apaga_aluno.php**.

4.6.2. Criando o formulário para edição dos dados dos alunos (form_atualiza_aluno.php).

Precisamos agora que após clicar no botão **EDITAR**, os dados do aluno selecionado, sejam transferidos para um formulário para que possam ser alterados e subsequentemente atualizados. Para tanto vamos criar a página **form_atualiza_alunos.php**.

Clicando em EDITAR será aberto o formulário para a edição:



The screenshot shows a web browser window with the address bar highlighted in red, displaying the URL `localhost/exemplo/restrito/form_atualiza_aluno.php?ra=2`. The page is titled 'Painel de Controle' and features a sidebar with navigation links: 'Painel de Controle', 'Cadastrar Aluno', 'Cursos', 'Cursando', and 'Cadastrar Usuários'. The main content area is titled 'Atualizar Dados do Aluno' and contains a form with the following fields: 'Nome' (filled with 'Maria Mendonça da Silva'), 'Email' (filled with 'maria.mendonca@email.com'), and 'Observações' (empty). A blue button labeled 'Atualizar' is positioned below the 'Observações' field.

Figura 26 - Formulário de Edição

Na sequência, basta editar as informações nos campos que apresentam os valores e clicar no botão **ATUALIZAR**.

Abaixo o código da página *form_atualiza_alunos.php* que irá receber as informações do aluno selecionado:

```
1 <?php
2     include 'conecta.php';
3     $ra_aluno = $_GET['ra'];
4 >?
5 <!doctype html>
6 <!--[if lt IE 7]>         <html class="no-js lt-ie9 lt-ie8 lt-ie7" lang=""> <![endif]-->
7 <!--[if IE 7]>           <html class="no-js lt-ie9 lt-ie8" lang=""> <![endif]-->
8 <!--[if IE 8]>           <html class="no-js lt-ie9" lang=""> <![endif]-->
9 <!--[if gt IE 8]><!--> <html class="no-js" lang="pt-br"> <!--<![endif]-->
```

Figura 27 - Início do código recebendo o valor do ra do aluno selecionado

```

168 </div>
169 <div class="card-body card-block">
170 <form action="atualiza_aluno.php?ra_aluno=<?php echo $ra_aluno; ?>" method="POST" class="">
171
172 <?php
173 // Executa uma consulta que pega todos os registros
174
175 $sql = "SELECT * FROM ALUNOS WHERE ra = $ra_aluno";
176 $consulta = $conexao->query($sql);
177 while ($dados = $consulta->fetch_assoc()) {
178
179
180 <div class="form-group"><label for="nome_novo" class=" form-control-label">Nome</label><input type="text" id="
181 nome_novo" name="nome_novo" value="<?php echo $dados['nome_aluno']; ?>" class="form-control"></div>
182 <div class="form-group"><label for="email_novo" class=" form-control-label">Email</label><input type="email" id="
183 email" name="email_novo" value="<?php echo $dados['email_aluno']; ?>" class="form-control"></div>
184 <div class="form-group"><label for="obs_novo" class=" form-control-label">Observações</label><input type="text" id="
185 obs" name="obs_novo" value="<?php echo $dados['obs']; ?>" class="form-control"></div>
186
187 <?php
188 }
189
190 <button type="submit" class="btn btn-primary btn-sm">
191 <i class="fa fa-dot-circle-o"></i> Atualizar
192 </button>
193
194 </form>

```

Figura 28 - Alteração do formulário para preencher com os dados selecionados e permitir a edição.

Os dados recebidos neste formulário, serão enviados para a página **atualiza_aluno.php**. Abaixo o código da página:

```

1 <?php
2 include 'conecta.php';
3
4 $ra = $_REQUEST['ra_aluno'];
5 $nome_novo = $_REQUEST['nome_novo'];
6 $email_novo = $_REQUEST['email_novo'];
7 $obs = $_REQUEST['obs_novo'];
8
9
10 $consulta = "UPDATE ALUNOS SET nome_aluno = '$nome_novo', email_aluno = '$email_novo', obs = '$obs_novo' WHERE ra = $ra" ;
11 $conexao->query($consulta);
12
13 header('Location: index.php');
14
15 ?>

```

Figura 29 - Código da página atualiza_aluno.php

4.6.3. Apagando um registro da tabela alunos.

Voltando ao início (página index.php) temos a seguinte tela:

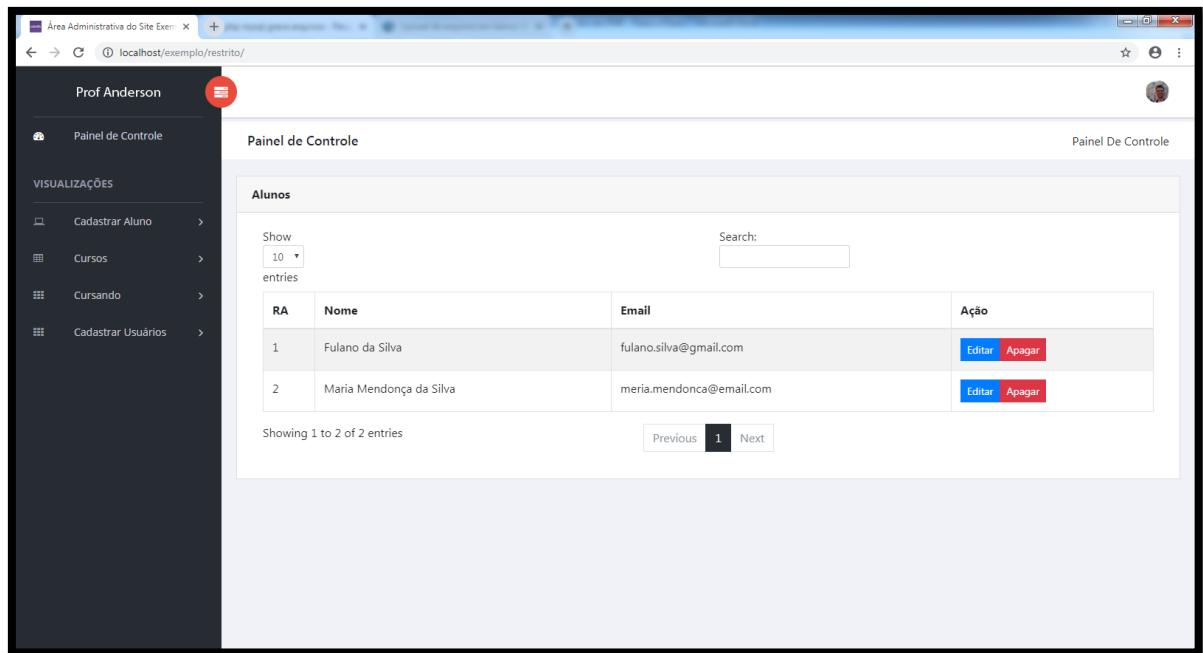


Figura 30 - Tela index.php

O botão APAGAR possui o seguinte código:

```
echo "<a href='apaga_aluno.php?ra=".$dados["ra"]."' class='btn btn-danger btn-sm'>Apagar</a>";
```

Figura 31 - Código do botão APAGAR

Precisamos agora criar a página **apaga_aluno.php** que será um script que irá executar a instrução no banco de dados e retornar para a página inicial **index.php**. Segue o código abaixo:

```
1 <?php
2 include 'conecta.php';
3
4 $ra_aluno = $_REQUEST['ra'];
5
6 // Executa uma consulta que deleta um registro
7 $sql = "DELETE FROM ALUNOS WHERE ra=$ra_aluno";
8
9 $query = $conexao->query($sql);
10
11 header('Location: index.php');
12
13 ?>
```

Figura 32 - Código da página apaga_aluno.php

Até aqui fizemos as quatro operações básicas em um sistema CRUD: Selecionar, Atualizar, Inserir e Apagar. Para as outras telas desse sistema, se repetem as mesmas operações selecionando outras tabelas no banco de dados.

No próximo capítulo vamos criar as telas de login para que usuários que não estejam logados no sistema não possam acessar essas informações. Trabalharemos criando as **SESSIONs** em php.

5. CRIANDO O LOGIN

Retornaremos agora para a página inicial de nosso site a index.html.

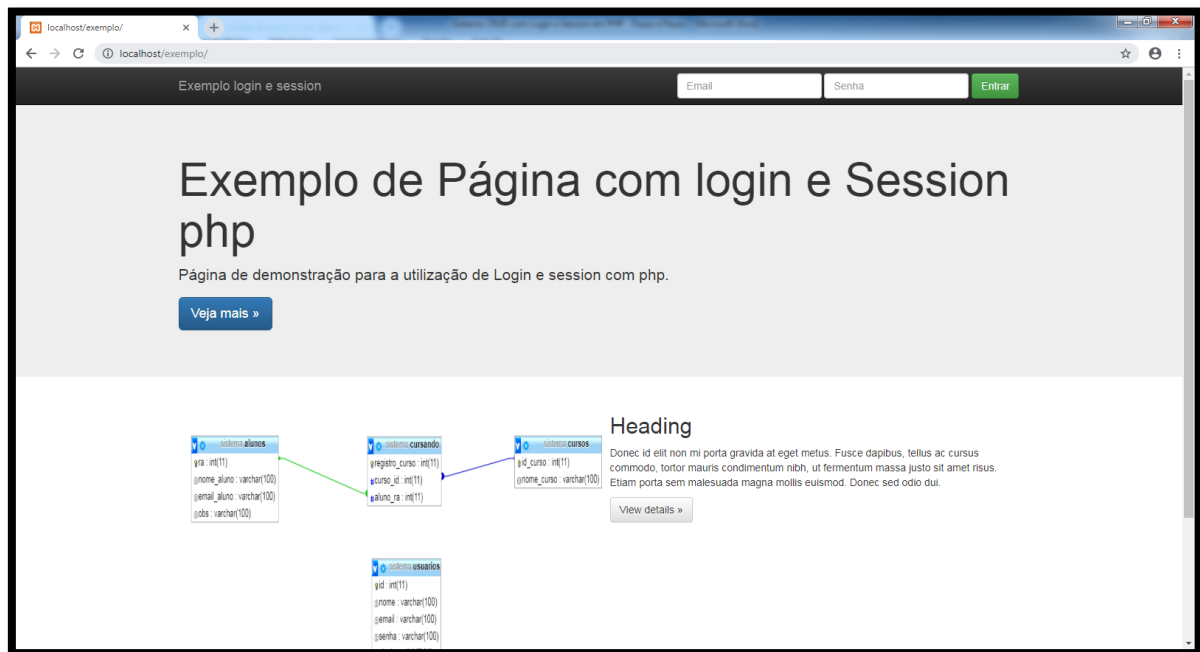


Figura 33 - Página inicial do sistema index.html

Nessa página iremos programar os campos do formulário logo no cabeçalho do site que irá receber o *email* e *senha* dos usuários cadastrados para que possam acessar as informações do banco de dados.

Faça as seguintes alterações no formulário de login:

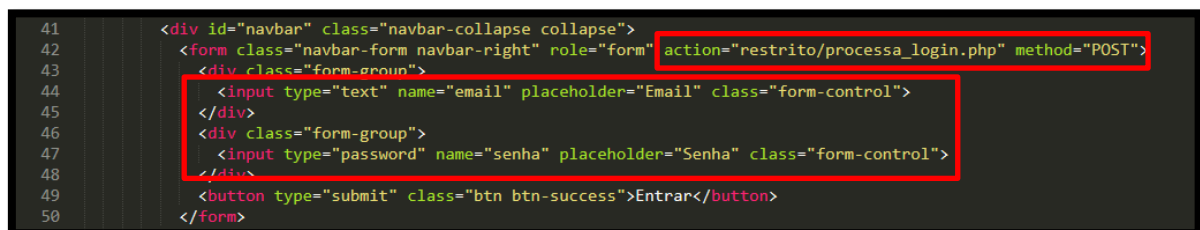


Figura 34 - Formulário da index.html para realizar o Login no sistema.

Agora vamos criar a página que irá conectar ao banco, receber os dados de *login*, verificar se este usuário existe, direcionar para a área restrito e criar uma **SESSION** de modo que somente usuários logados consigam acessar o sistema e visualizar as informações.

5.1. Página processa_login.php

Essa página ficará alocada dentro da pasta restrito e irá verificar as informações referentes ao login do usuário. Segue o código desta página:

```
processa_login.php x
1  <?php
2      session_start();
3      require('conecta.php');
4
5      $email = !empty($_REQUEST['email'])?$_REQUEST['email']:'';
6      $senha = !empty($_REQUEST['senha'])?$_REQUEST['senha']:'';
7
8      /*=====*/
9
10     if($email&&$senha){
11
12         $consulta = "SELECT * FROM usuarios WHERE email = '$email' AND senha = '$senha'";
13         $resultado = $conexao->query($consulta);
14         $registros = $resultado->num_rows;
15         $resultado_usuario = mysqli_fetch_assoc($resultado);
16
17
18         if($registros<>0){
19             //echo "VALIDADO COM SUCESSO!";
20             $_SESSION['id'] = $resultado_usuario['id'];
21             $_SESSION['nome'] = $resultado_usuario['nome'];
22             $_SESSION['email'] = $resultado_usuario['email'];
23
24             header('Location: index.php');
25         }
26         else{
27             //echo "ERRO NO LOGIN!";
28             header('Location: ../index.html');
29         }
30     }
31     else{
32         //echo "ERRO: CAMPOS VAZIOS!";
33         header('Location: ../index.html');
34     }
35
36     ?>
```

Figura 35 - Página processa_login.php

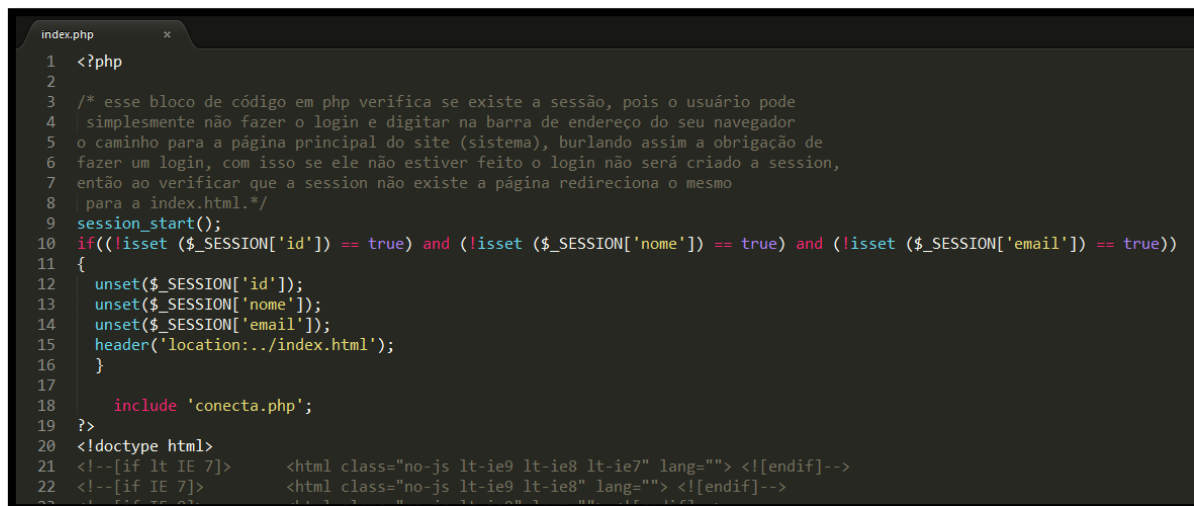
Iniciamos essa página criando uma sessão ativa no php (linha 2), na sequencia fazemos a conexão com o banco de dados (linha 3) e recebemos as variáveis email e senha (linhas 5 e 6) fazendo uma verificação se as mesmas não vieram vazias.

Após verificar que email e senha são válidos e foi encontrado pelo menos 1 registro na tabela usuários (linha 18), setamos as variáveis de Sessão com alguns valores obtidos da tabela usuário (linhas 20 a 22) que serão utilizados para validar as sessões abertas e impedir que outros usuários que não estejam logados acessem dados. Feito isso o sistema é direcionado para a página **index.php** da área restrita. Caso essas informações não possam ser verificadas, o usuário é direcionado para a página inicial de nosso sistema.

5.1.1. Configurando as páginas que são restritas.

Todas as páginas que necessitem que um usuário específico esteja logado no sistema para acessar as informações, necessitam de uma pequena alteração, de modo que, antes que a página seja carregada, o php faça uma verificação que uma sessão está ativa e que os dados estão presentes.

Veja abaixo alteração da página **index.php** da área restrita:



```
1 <?php
2
3 /* esse bloco de código em php verifica se existe a sessão, pois o usuário pode
4 simplesmente não fazer o login e digitar na barra de endereço do seu navegador
5 o caminho para a página principal do site (sistema), burlando assim a obrigação de
6 fazer um login, com isso se ele não estiver feito o login não será criado a session,
7 então ao verificar que a session não existe a página redireciona o mesmo
8 para a index.html.*/
9 session_start();
10 if((!isset($_SESSION['id']) == true) and (!isset($_SESSION['nome']) == true) and (!isset($_SESSION['email']) == true))
11 {
12     unset($_SESSION['id']);
13     unset($_SESSION['nome']);
14     unset($_SESSION['email']);
15     header('location:../index.html');
16 }
17
18 include 'conecta.php';
19 ?>
20 <!doctype html>
21 <!--[if lt IE 7]> <html class="no-js lt-ie9 lt-ie8 lt-ie7" lang=""> <![endif]-->
22 <!--[if IE 7]> <html class="no-js lt-ie9 lt-ie8" lang=""> <![endif]-->
```

Figura 36 - Sessão configurada na página index.php

Dessa forma, mesmo que alguém conheça o endereço desta página, ao digitar na url sem efetuar o login, o sistema irá redirecionar este usuário para a página inicial do sistema (**index.html**).

Lembrando que essa configuração deve estar presente em todas as páginas restritas. Basta copiar este trecho para todas as outras páginas.

5.1.2. Configurando um nome de usuário para exibição.

Com uma sessão ativa, é muito simples utilizar esses parâmetros para a personalização de um site. Um exemplo muito clássico desse tipo de personalização, é uma mensagem de boas vindas para o usuário que está logado exibindo o seu nome em alguma área específica do site. Veja o exemplo abaixo após um administrador logar no sistema:

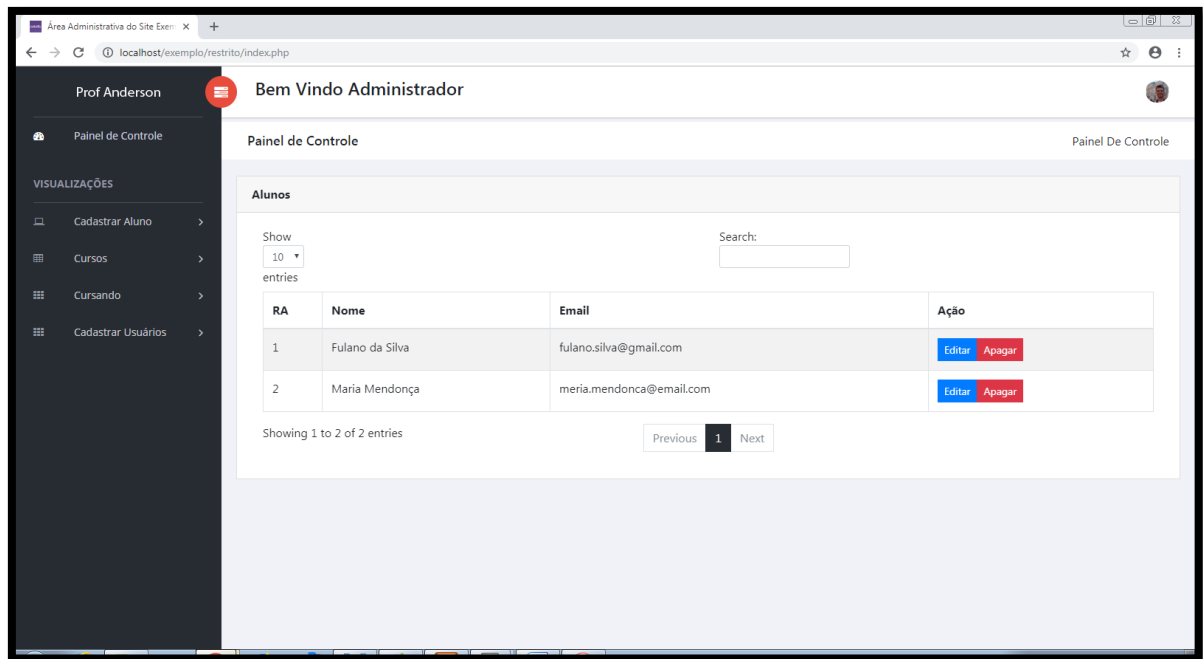


Figura 37 - Personalizando mensagens de boas vindas.

```

120                                     </div>
121
122                                     <div class="dropdown for-message">
123                                     |
124                                     <h4>Bem Vindo <?php echo $_SESSION['nome']; ?></h4>
125                                     |
126                                     </div>
127                                     </div>
128                                     </div>

```

Figura 38 - Editando o código para exibir um valor de Sessão.

5.2. Botão Sair e encerramento das Sessões ativas.

Outra configuração muito importante é a configuração do botão Sair. Ele será o responsável por eliminar todos os dados das sessões que foram iniciadas de modo a garantir a segurança do sistema.

A seguir o código da página **sair.php** que será utilizada por todas as outras que possuam a opção sair.


```

1▼ <?php
2    session_start();
3    //apaga dados da sessão ativa
4▼    unset(
5        $_SESSION['id'],
6        $_SESSION['nome'],
7        $_SESSION['email']
8    );
9
10
11    //redirecionar o usuario para a página de login
12    header("Location: ../index.html");
13 ?>

```

Figura 39 - Página sair.php

Agora basta direcionar o botão sair de cada uma das páginas restritas para esse arquivo. Todas as sessões ativas serão eliminadas e o usuário será redirecionado para a página inicial do sistema (*index.html*).

```
<a class="nav-link" href="sair.php"><i class="fa fa-power -off"></i>Sair</a>
```

Figura 40 - Codificação do botão sair

6. IMPLEMENTANDO A CRIPTOGRAFIA NAS SENHAS.

Criptografia é o ato de codificar dados em informações aparentemente sem sentido, para que pessoas não consigam ter acesso às informações que foram cifradas. Há vários usos para a criptografia em nosso dia-a-dia: proteger documentos secretos, transmitir informações confidenciais pela Internet ou por uma rede local, etc.

A técnica usada em Criptografia envolve pura e simples matemática. O sistema de criptografia usado atualmente é extremamente seguro. Especialistas estimam que para alguém conseguir quebrar uma criptografia usando chaves de 64 bits na base da tentativa e erro, levaria cerca de 100.000 anos usando um PC comum.

Mas porque é tão seguro? Simples, uma chave de 2 bits terá 4 combinações possíveis. Uma chave de 4 bits, terá 16 combinações possíveis e assim por diante. Agora, vejamos a grande diferença: uma chave de apenas 8 bits terá 65.356 combinações possíveis e, em uma chave de 32 bits existem mais de 4 bilhões de combinações, que para serem decifradas levariam mais de 2 meses (levando em conta 1000 tentativas por segundo, por parte do computador).

6.1. MD5

O MD5 (Message-Digest algorithm 5) é um algoritmo de hash de 128 bits unidirecional desenvolvido pela RSA Data Security, Inc., descrito na RFC 1321, usado por softwares com protocolo ponto-a-ponto (P2P), verificação de integridade e logins. Foi desenvolvido para suceder ao MD4 que tinha alguns problemas de segurança.

Por ser um algoritmo unidirecional, um hash MD5 não pode ser transformado novamente na password (ou texto) que lhe deu origem. O método de verificação é, então, feito pela comparação das duas hash (uma da base de dados, e a outra da tentativa de login). O MD5 também é usado para verificar a integridade de um ficheiro através, por exemplo, do programa md5sum, que cria a hash de um ficheiro. Isto pode-se tornar muito útil para downloads de ficheiros grandes, para programas P2P que constroem o ficheiro através de pedaços e estão sujeitos à corrupção de ficheiros.

O MD5 é de domínio público para uso em geral. A partir de uma mensagem de um tamanho qualquer, ele gera um valor hash de 128 bits; com este algoritmo, é computacionalmente impraticável descobrir duas mensagens que gerem o mesmo valor, bem como reproduzir uma mensagem a partir do seu digest. O algoritmo MD5 é utilizado como mecanismo de integridade em vários protocolos de padrão Internet (RFC1352, RFC1446, etc.), bem como pelo CERT e CIAC.

Para usar o md5 no PHP é só usar da seguinte forma por exemplo:

```
<?php
$string = 'O rato reu a ropa do rei de Roma';
$codificada = md5($string);
echo "Resultado da codificação usando md5: " . $codificada;
// 54cf74d1acdb4037ab956c269b63c8ac
?>
```

6.2. Configurando a criptografia no sistema

Para que uma senha digitada seja armazenada no banco de dados com criptografia, precisamos usar a função md5 do php e converter o valor da variável senha antes de armazená-la no banco de dados.

Em nossa configuração atual, apesar de que ao digitarmos a senha no formulário de cadastro a mesma não seja visível, ao ser gravada no banco de dados ela aparece sem criptografia. Veja:

Figura 41 - Formulário de entrada de dados e senha

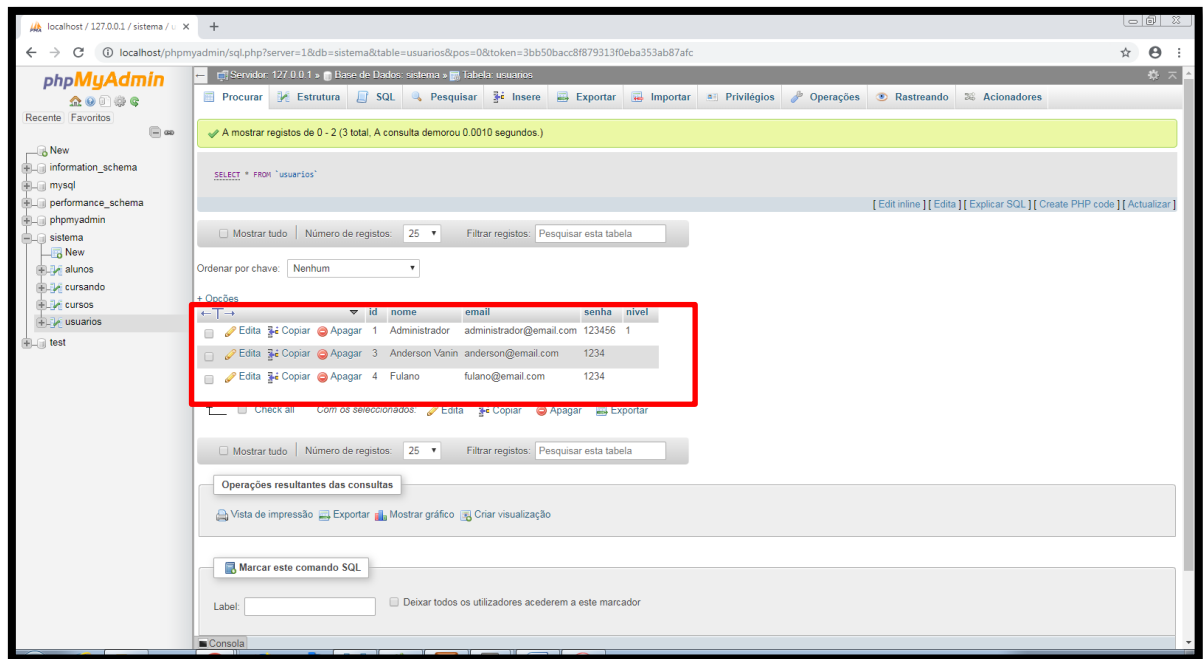


Figura 42 - Dados cadastrados no banco de dados sem criptografia.

Vamos agora fazer uma pequena alteração no arquivo que faz o armazenamento de novos usuários no sistema, arquivo ***cadastra_usuario.php***.

```

1 <?php
2     include 'conecta.php';
3
4     $nome = $_REQUEST['nome'];
5     $email = $_REQUEST['email'];
6     $senha = md5($_REQUEST['senha']);
7
8     $consulta = "INSERT INTO usuarios (nome,email,senha) VALUES ('$nome','$email','$senha')";
9     $conexao->query($consulta);
10
11     header('Location: index.php');
12
13 ?>

```

Figura 43 - Configurando o MD5 no campo senha.

Fazendo um novo cadastro de usuário agora teremos o seguinte dado armazenado no banco de dados:

+ Opções						
		id	nome	email	senha	nivel
<input type="checkbox"/>	<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	1	Administrador	administrador@email.com	123456	1
<input type="checkbox"/>	<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	3	Anderson Vanin	anderson@email.com	1234	
<input type="checkbox"/>	<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	4	Fulano	fulano@email.com	1234	
<input type="checkbox"/>	<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Apagar	5	Um outro usuário	outro@email.com	81dc9bdb52d04dc20036dbd8313ed055	

Figura 44 - Dado inserido com senha criptografada.

Com a senha criptografada, para que o sistema consiga validar essa informação no ato do login é necessário que ao preencher o formulário de login a senha digitada também seja criptografada para que o script de validação dos dados digitados (processa_login.php) possa verificar se as senhas são iguais.

Para isso vamos realizar uma pequena alteração na página que faz o processamento de login (processa_login.php)

```
1  <?php
2      session_start();
3      require('conecta.php');
4
5      $email = !empty($_REQUEST['email'])?$_REQUEST['email']:'';
6      $senha = md5(!empty($_REQUEST['senha'])?$_REQUEST['senha']:'');
7
8      /*=====*/
9
10     if($email&&$senha){
11
12         $consulta = "SELECT * FROM usuarios WHERE email = '$email' AND senha = '$senha'";
13         $resultado = $conexao->query($consulta);
14         $registros = $resultado->num_rows;
```

Figura 45 - Alterando a página processa_login.php