
AGENDA14

MÓDULO ADMINISTRADOR – ATUALIZAÇÃO MVC





MERGULHANDO NO TEMA...

A arquitetura de camadas MVC possibilita a integração e a divisão de tarefas entre equipes, facilita o reaproveitamento de códigos e mantém a codificação limpa, diminuindo a complexidade do sistema. Assim, as alterações realizadas em uma das camadas não interfere nas demais, facilitando a alteração nas regras de negócio, a atualização de layouts e a adição de novos recursos.



Imagem 02 - freepik.com

Na arquitetura MVC você sabe onde ficam as regras de negócio e a maioria dos frameworks já vem com uma estrutura de diretórios pronta, por isso, no caso de uma atualização, você consegue localizar facilmente os arquivos.

Vamos acompanhar um processo de atualização, semelhante ao que a equipe da Bia deverá fazer.

Banco de dados

Antes de mais nada os analistas de banco de dados nos passam a tabela de administradores que será utilizada como meio de login para o novo painel a ser desenvolvido. Então nosso projeto vai ser baseado no diagrama a seguir.

Para testes o ideal é criar uma tabela. Veja um script de criação:

```
-- Table `projeto_final`.`administrador`
CREATE TABLE IF NOT EXISTS `projeto_final`.`administrador` (
  `idadministrador` INT(11) NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(45) NULL DEFAULT NULL,
  `cpf` VARCHAR(11) NOT NULL,
  `senha` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idadministrador`))
ENGINE = InnoDB
AUTO_INCREMENT = 2
DEFAULT CHARACTER SET = latin1;
```

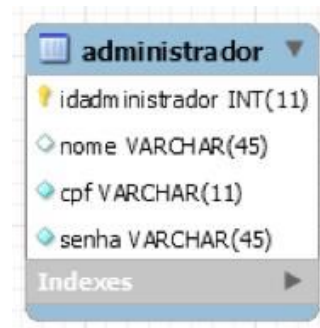


Imagem 03 -
Diagrama
Banco de dados.

Agora é hora de colocar em prática seus conhecimentos em PHP Orientado a Objetos e Arquitetura MVC começando por desenvolver a camada model da atualização.

Veja..

Observação: Não esqueça de inserir direto por meio de SQL um usuário administrador.

Exemplo:

```
INSERT INTO `projeto_final`.`administrador` (`nome`, `cpf`, `senha`)
VALUES ('Bia', '2222222222', 'bia123');
```

Camada Model

Agora vamos modelar a classe Administrador será responsável por gerenciar os dados dos administradores do nosso projeto. Seu nome será “Administrador”, então novamente dentro da pasta Model crie este arquivo php. Não esquecendo de colocar os delimitadores PHP “<?php?” . O desenvolvimento desta classe será baseado no diagrama a seguir.

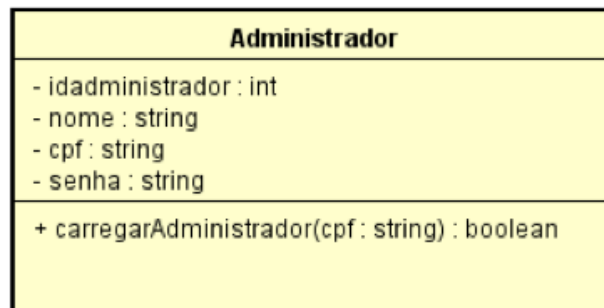


Imagem 04 - Diagrama da Classe Administrador.

Logo após os delimitadores devemos criar a classe e seus atributos:

- idadministrador – código único de cada registro de administrador;
- nome – nome do administrador;
- cpf – cpf do administrador;
- senha – senha para acesso ao nosso painel.

Como resultado deve-se obter o seguinte código.

```
class Administrador
{
    private $id;
    private $nome;
    private $cpf;
    private $senha;
}
```

O próximo passo é criar todos os métodos getters e setters de cada um desses atributos. Eles darão acesso aos atributos privados desta classe. Então, veja a codificação, seguindo os padrões de projeto:

```
//ID
public function setID($id)
{
    $this->id = $id;
}
public function getID()
```

```

{
    return $this->id;
}

//nome
public function setName($nome)
{
    $this->nome = $nome;
}
public function getName()
{
    return $this->nome;
}

//cpf
public function setCPF($cpf)
{
    $this->cpf = $cpf;
}
public function getCPF()
{
    return $this->cpf;
}

// Senha
public function setSenha($senha)
{
    $this->senha = $senha;
}
public function getSenha()
{
    return $this->senha;
}

```

O próximo passo é desenvolver um método específico, que será o carregarAdministrador.

```

public function carregarAdministrador($cpf)
{
    require_once 'ConexaoBD.php';

    $con = new ConexaoBD();
    $conn = $con->conectar();
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    $sql = "SELECT * FROM administrador WHERE cpf = ".$cpf ;
    $re = $conn->query($sql);
    $r = $re->fetch_object();
    if($r != null)
    {
        $this->id = $r->idadministrador;
        $this->nome = $r->nome;
        $this->cpf = $r->cpf;
        $this->senha = $r->senha;
    }
}

```

```

        $conn->close();
        return true;
    }
    else
    {
        $conn->close();
        return false;
    }
}

```

Este método segue o mesmo padrão dos que foram desenvolvidos nas agendas anteriores:

- Inclusão da Classe ConexaoBD;
- Instância do Objeto da Classe ConexãoBD;
- Conexão ao Banco de Dados, com verificação do sucesso ou não;
- Confeção da sentença SQL.
- Execução da sentença com verificação do sucesso ou não.

Em caso positivo, deve-se popular os dados do objeto com resultado da consulta ao Banco de Dados, em seguida, deve-se fechar a conexão e por fim retornar a TRUE, como se pode observar no trecho do código a seguir:

```

$this->id = $r->idadministrador;
$this->nome = $r->nome;
$this->cpf = $r->cpf;
$this->senha = $r->senha;
$conn->close();
return true;

```

Como foi solicitado para que seja possível listar todos os usuários, é preciso realizar uma atualização na classe Usuário, já que a lista de usuários faz parte do seu domínio, então, a classe Usuário sofrerá a inclusão de um novo método específico, como demonstrado no diagrama a seguir:

Usuário
- idusuario : int - nome : string - cpf : string - dataNascimento : date - email : string - senha : string
+ inserirBD() : boolean + carregarUsuario(cpf : string) : boolean + atualizarBD() : void + listaCadastrados() : tabelaRegistros

Imagem 05 - Dda Classe Usuario com o novo método específico

```

public function listaCadastrados()
{
    require_once 'ConexaoBD.php';

    $con = new ConexaoBD();
    $conn = $con->conectar();
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    $sql = "SELECT idusuario, nome FROM usuario;" ;
    $re = $conn->query($sql);
    $conn->close();
    return $re;
}

```

Este método segue inicialmente o mesmo padrão:

- Inclusão da Classe ConexãoBD;
- Instância do Objeto da Classe ConexãoBD;
- Conexão ao Banco de Dados, com verificação do sucesso ou não;
- Confeção da sentença SQL.

Porém, este método retorna o resultado da consulta do Banco de Dados que pode ser um ou mais registros. Nesse caso, serão retornados todos os usuários cadastrados na tabela usuário.

Finalizadas as atualizações na classe Model, podemos agora partir para o desenvolvimento da camada View.

Camada View

Para continuar colocando em prática o conhecimento e desenvolver essa atualização, é preciso criar a camada View para apresentar e interagir com o usuário final, possibilitando a visualização do projeto, então, toda a interface faz parte dessa camada: os dados, as informações e os gráficos.

Nesta atualização será necessário desenvolver as seguintes páginas (Interfaces):

1. Login – Página para o administrador inserir seu CPF e senha para acessar o conteúdo do painel.
2. Principal – Página central que gerenciará todas as informações do usuário.
3. Listar Cadastrados – fará uma lista com todos os usuários cadastrados.

Então, mãos à obra! Vamos começar o desenvolvimento da interface:

Interface Login - Nesse momento vamos desenvolver a interface que será responsável por realizar a interação com o administrador, por meio do login. O CPF será “login”, então dentro da pasta View, crie o arquivo PHP denominado ADMLogin. O desenvolvimento desta interface será baseado no mesmo layout do login do usuário, com algumas pequenas mudanças, como pode ser observado a seguir:

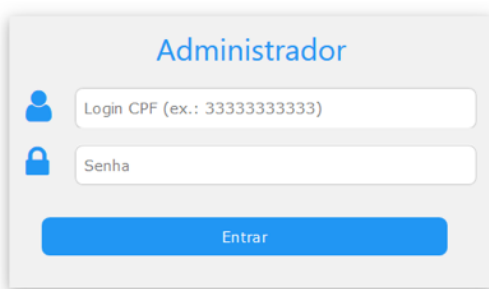


Imagem 06 - InterfaceLogin.

Para começar o desenvolvimento, criaremos a estrutura básica de uma página html, sem esquecer de linkar o framework W3Css e biblioteca de ícones. Veja:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="style.css">
```

```

<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
<title>Administrador</title>
</head>
<body>

</body>
</html>

```

Dentro da tag <body> vamos criar um formulário com:

- txtLoginADM – input do tipo text;
- txtSenhaADM – input do tipo password;
- btnLoginADM – button para realizar o login;

O código deve ficar:

```

<form action="/Controller/navegacao.php" method="post" class="w3-container w3-card-4 w3-light-grey w3-
text-blue w3-margin w3-display-middle" style="width: 30%;">
  <input type="hidden" name="nome_form" value="frmLoginADM" />
  <h2 class="w3-center">Administrador</h2>
  <div class="w3-row w3-section">
    <div class="w3-col" style="width:11%"><i class="w3-xxlarge fa fa-user"></i></div>
    <div class="w3-rest">
      <input class="w3-input w3-border w3-round-large" name="txtLoginADM" type="text"
placeholder="Login CPF (ex.: 33333333333)">
    </div>
  </div>
  <div class="w3-row w3-section">
    <div class="w3-col" style="width:11%"><i class="w3-xxlarge fa fa-lock"></i></div>
    <div class="w3-rest">
      <input class="w3-input w3-border w3-round-large" name="txtSenhaADM" type="password"
placeholder="Senha">
    </div>
  </div>
  <div class="w3-row w3-section">
    <div class="" style="">
      <button name="btnLoginADM" class="w3-button w3-block w3-margin w3-blue w3-cell w3-round-
large" style="width: 90%;">Entrar</button>
    </div>
  </div>
</form>

```

Atenção:

- Como já foi desenvolvido, e já sabemos que a action será direcionada para a camada controller, mais especificamente o arquivo navegação, já deixamos codificado.
- Neste exemplo não teremos primeiro acesso, já que foi entregue uma tabela pronta com usuários. Para realizar os testes, insira um administrador via SGBD.

Interface Principal – Agora vamos desenvolver a interface principal da página. Ela será responsável por toda a gerencia administrativa, que no caso ainda é bem simples porque o único recurso solicitado é listar todos os usuários cadastrados.

Com o nome “ADMPrincipal”, novamente dentro da pasta View, crie este arquivo php. O desenvolvimento desta etapa será baseado no layout a seguir.

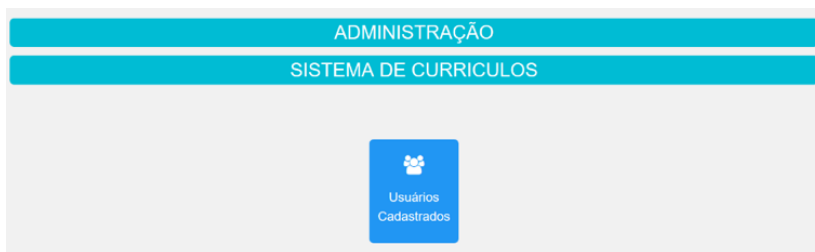


Imagem 07 - interface ADMPrincipal

Vamos lá... Primeiro, criaremos a estrutura básica da página, sem esquecer de linkar o framework W3Css e biblioteca de ícones.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="style.css">
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  <title>Painel de Administração</title>
  <style>

    body,
    h1,
    h2,
    h3,
    h4,
    h5,
    h6 {
      font-family: "Montserrat", sans-serif
    }
  </style>
</head>
<body class="w3-light-grey">

</body>
</html>
```


Não podemos esquecer da verificação e abertura da sessão. Veja:

```
<?php
if(!isset($_SESSION))
{
    session_start();
}
?>
```

Agora, ainda dentro das tags body, criaremos uma div para ser responsável pelo conteúdo do painel. A codificação deve ficar:

```
<div class="w3-padding-large" id="main">
</div>
```

Seguindo o layout, devemos criar um cabeçalho, simples com dois títulos: Administração e Sistema de Currículos. Veja:

```
<header class="w3-container w3-padding-32 w3-center " id="home">
<br>
<h1 class="w3-text-white w3-panel w3-cyan w3-round-large">
    ADMINISTRAÇÃO
</h1>
<h1 class="w3-text-white w3-panel w3-cyan w3-round-large">
    SISTEMA DE CURRÍCULOS
</h1>
</header>
```

Por fim, criaremos um formulário com a função de agregar todas as funções e futuras atualizações requeridas pelo setor de Recursos Humanos.

```
<form action="/Controller/navegacao.php" method="post" class="w3-container w3-light-grey w3-text-blue
w3-margin w3-center" style="">
    <input type="hidden" name="nome_form" value="frmLoginADM" />

    <button name="btnListarCadastrados" class="w3-button w3-margin w3-blue w3-cell w3-
round-large" style="">
        <br>
        <i class="fa fa-address-book-o w3-xxlarge"></i><br>
        <p class="w3-xlarge">Usuários<br>
        Cadastrados</p>
    </button>
```

Interface ADMListarCadastrados - Neste momento, vamos desenvolver a interface que será responsável por exibir uma lista com todos os usuários. O desenvolvimento desta interface será baseado no layout a seguir:

Lista de Usuários Cadastrados no Sistema	
Código	Nome
<div>Voltar</div>	

Imagem 08 - interface ADMListarCadastrados.

Primeiro, criaremos a estrutura básica da página, sem esquecer de linkar o framework W3Css e biblioteca de ícones.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="style.css">
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  <title>Usuários Cadastrados</title>
</head>
<body>

</body>
</html>
```

Não podemos esquecer da verificação e abertura da sessão. Veja:

```
if(!isset($_SESSION))
{
    session_start();
}
```

Seguindo o layout, agora devemos criar um cabeçalho, simples com apenas um título: Lista de Usuários Cadastrados no Sistema.

```
<header class="w3-container w3-padding-32 w3-center " >
  <h1 class="w3-text-white w3-panel w3-cyan w3-round-large">
    Lista de Usuários Cadastrados no Sistema
  </h1>
</header>
```

Criaremos uma tabela com o nome e código do usuário a ser listado.

```
<div class="w3-padding-128 w3-content w3-text-grey" >
  <div class="w3-container">
    <table class="w3-table-all w3-centered">
      <thead>
        <tr class="w3-center w3-blue">
          <th>Código</th>
          <th>Nome</th>
        </tr>
      </thead>
    </table>
  </div>
</div>
```

Por fim criaremos um botão para voltar à página principal do administrador. Veja:

```
<div class="w3-padding-128 w3-content w3-text-grey">
```

```

<form action="/Controller/navegacao.php" method="post" class="w3-container w3-light-grey w3-text-
blue w3-margin w3-center" style="width: 30%;">
    <div class="w3-row w3-section">
        <div>
            <button name="btnVoltar" class="w3-button w3-block w3-margin w3-blue w3-cell w3-round-
large" style="width: 90%;">
                Voltar
            </button>
        </div>
    </div>
</form>
</div>

```

Depois de toda a interface desenvolvida, é preciso desenvolver a camada controller para as respectivas ações solicitadas pelo RH.

Camada Controller

Agora vamos programar a camada responsável pelo fluxo de informação que passa pelo site. Essa camada gerará e definirá quais dados ou regras devem ser acionadas e para onde serão encaminhadas para serem exibidas posteriormente.

Login - Neste momento vamos criar um arquivo na camada controller chamado “AdministradorController”, que será o responsável pelo controle de todas as ações que forem necessárias e pelo acesso de dados dos objetos instanciados na classe administrador. Então dentro da pasta Controller, crie este arquivo PHP sem se esquecer de colocar os delimitadores PHP “<?php?>”. Veja:

```

if(!isset($_SESSION))
{
    session_start();
}
class AdministradorController{
    public function login($cpf, $senha)
    {
        require_once '../Model/Administrador.php';
        $administrador = new Administrador();
        $administrador->carregarAdministrador($cpf);
        if($administrador->getSenha() == $senha)
        {
            $_SESSION['Administrador'] = serialize($administrador);
            return true;
        }
        else
        {
            return false;
        }
    }
}
?>

```

O administrador deve realizar o login com seu CPF e senha cadastrados para conseguir acesso ao painel administrativo. No código anterior já definimos um método para realizar o login, seguindo o mesmo padrão do login do usuário normal.

Na página login já alteramos a action direcionando-a para o arquivo navegação, porém nesse arquivo precisamos desenvolver a condição para quando o administrador pressionar o botão Entrar (btnLoginADM), possa realizar o procedimento de verificação para saber se ele já está cadastrado. Então vamos codificar outro desvio condicional. Veja:

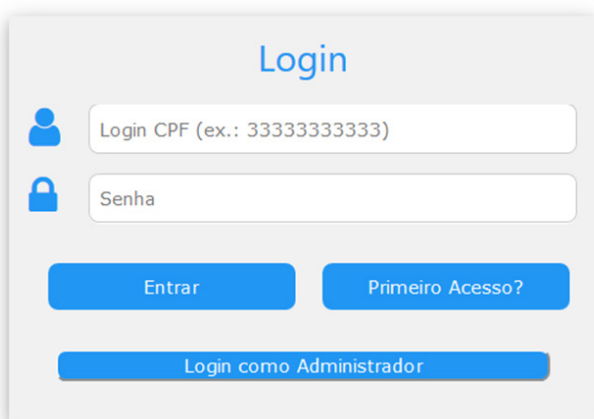
```
if(isset($_POST["btnLoginADM"]))
{
    require_once '../Controller/AdministradorController.php';

    $aController = new AdministradorController();

    if($aController->login($_POST['txtLoginADM'], $_POST['txtSenhaADM']))
    {
        include_once '../View/ADMPrincipal.php';
    }
    else
    {
    }
}
```

Dentro desse código é preciso outro desvio condicional que se retornar a verdadeiro será redirecionado para a página ADMPrincipal, caso contrário não teremos nenhum redirecionamento.

Após salvar, precisamos ainda criar, na interface login de usuário, um botão para redirecionar o administrador para sua tela de login. Para isso devemos criar um novo botão de acordo layout a seguir:



The image shows a login form titled "Login". It has two input fields: "Login CPF (ex.: 33333333333)" and "Senha". Below the fields are three buttons: "Entrar", "Primeiro Acesso?", and "Login como Administrador".

Imagem 08 - Novo layout da página Login.php

O código com essa alteração deve ficar dessa forma:

```
<div class="w3-row w3-section">
    <div class="w3-half" style="">
        <button name="btnLogin" class="w3-button w3-block w3-margin w3-blue w3-cell w3-round-
large" style="width: 90%;">Entrar</button>
    </div>
```

```

<div class="w3-half">
  <button name="btnPrimeiroAcesso" class="w3-button w3-block w3-margin w3-blue w3-cell w3-round-
large" style="width: 90%;">Primeiro Acesso?</button>
</div>
<div class="w3-center" style="">
  <button name="btnADM" class=" w3-block w3-margin w3-blue w3-cell w3-round-large"
style="width: 90%;">Login como Administrador</button>
</div>
</div>

```

Depois de realizar a alteração no layout devemos criar um desvio condicional no arquivo navegação para que quando o botão btnADM for pressionado, o usuário seja redirecionado para a tela de login de administrador. Então basta fazer a seguinte codificação:

```

if(isset($_POST["btnADM"]))
{
    include_once '../View/ADMLLogin.php';
}

```

Se tudo estiver correto, ao clicar no botão btnADM, o usuário será redirecionado à página para login do Administrador. Após realizar o login, o usuário será redirecionado para o painel de administração.

ListarCadastrador - Nessa atualização não será necessário realizar a criação de um novo arquivo, bastando atualizar os arquivos que já foram codificados. O primeiro passo é criar um desvio condicional para quando for pressionado o botão Usuários Cadastrados, na página principal, o usuário seja redirecionado para a página ListarCadastrados. Para isso é preciso fazer a seguinte codificação no arquivo navegação:

```

if(isset($_POST["btnListarCadastrados"]))
{
    include_once '../View/ADMListarCadastrados.php';
}

```

Em seguida, é preciso gerar a tabela com os dados dos usuários cadastrados no arquivo ADMListarCadastrados. Para isso, primeiro devemos codificar as ligações com as classes que serão utilizadas junto a abertura de sessão. Veja:

```

<?php
include_once '../Model/Usuario.php';
include_once '../Controller/UsuarioController.php';
if(!isset($_SESSION))
{
    session_start();
}
?>

```

Depois, basta fazer a seguinte codificação dentro da tabela já definida anteriormente:

```

<?php

$usuario = new UsuarioController();
$results = $usuario->gerarLista();
if($results != null)

while($row = $results->fetch_object()) {
    echo '<tr>';

```

```

        echo '<td style="width: 1%;">'.$row->idusuario.'</td>';
        echo '<td style="width: 50%;">'.$row->nome.'</td>';

        echo '</tr>';
    }
    ?>

```

Ao testar o botão, o resultado deve ser similar ao apresentado na imagem a seguir:

Lista de Usuários Cadastrados no Sistema

Código	Nome
1	Paulo
2	Paulo

Imagem 09 - Resultado de listagem de usuários cadastrados

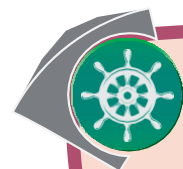
Para encerrar esta etapa, precisamos codificar o clique no botão btnVoltar do arquivo ADMListarCadastrados, para isso basta apenas adicionar mais um desvio condicional (if) no arquivo navegação. Veja:

```

if(isset($_POST["btnVoltar"]))
{
    include_once '../View/ADMPPrincipal.php';
}

```

Finalizando o módulo administrador com todas funcionalidades requeridas.



VOCÊ NO COMANDO

Novos requisitos surgiram para o módulo administrador: o setor de RH solicitou que fosse criada uma listagem de usuários administradores no mesmo formato dos usuários cadastrados. Para isso, crie ou atualize as camadas Model, View e Controller para que seja possível realizar a atualização, devendo apresentar: Código, Nome e CPF.

Utilize o Layout a seguir para a criação da interface:

Lista de Administradores Cadastrados no Sistema

Código	Nome	CPF
1	Paulo	11111111111

Voltar

Imagem 10 - Layout de Administradores.

Dicas:

- Utilize as classes Usuario, UsuarioController como base para o desenvolvimento.
- Não esqueça da action dos formulários.
- Não esqueça de codificar as ações para os eventos de clique do botão voltar.

Confira abaixo se você conseguiu resolver o desafio propostos!

Saiba que a solução apresentada a seguir é apenas uma das possibilidades e variações.

Alterações na Classe Administrador da camada Model: Criação de método listar Cadastrados.

```
public function listaCadastrados()
{
    require_once 'ConexaoBD.php';

    $con = new ConexaoBD();
    $conn = $con->conectar();
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    $sql = "SELECT idadministrador, nome, cpf FROM administrador;" ;
    $re = $conn->query($sql);
    $conn->close();
    return $re;
}
```

Alterações na Classe Administrador Controller da camada Controller (Criação de método GerarLista)

```
public function gerarLista()
{
    require_once '../Model/Administrador.php';

    $u = new Administrador();

    return $results = $u->listaCadastrados();
}
```

Interface ADMListarAdminsitadores da camada View.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
    <title>Administradores Cadastrados</title>
</head>
<body>
<?php
    include_once '../Model/Administrador.php';
    include_once '../Controller/AdministradorController.php';
    if(!isset($_SESSION))
    {
        session_start();
    }
?>

<!--Inicial -->
<header class="w3-container w3-padding-32 w3-center ">
```

```

<h1 class="w3-text-white w3-panel w3-cyan w3-round-large">
  Lista de Administradores Cadastrados no Sistema
</h1>
</header>
<div class="w3-padding-128 w3-content w3-text-grey">
  <div class="w3-container">
    <table class="w3-table-all w3-centered">
      <thead>
        <tr class="w3-center w3-blue">
          <th>Código</th>
          <th>Nome</th>
          <th>CPF</th>

        </tr>
      <thead>
        <?php
          $adm = new AdministradorController();
          $results = $adm->gerarLista();
          if($results != null)

            while($row = $results->fetch_object()) {
              echo '<tr>';
              echo '<td style="width: 1%;">'.$row->idadministrador.'</td>';
              echo '<td style="width: 50%;">'.$row->nome.'</td>';
              echo '<td style="width: 50%;">'.$row->cpf.'</td>';
              echo '</tr>';
            }
          ?>

        </table>

      </div>
    </div>

<div class="w3-padding-128 w3-content w3-text-grey">
  <form action="/Controller/navegacao.php" method="post" class="w3-container w3-light-grey w3-text-
blue w3-margin w3-center" style="width: 30%;">
    <div class="w3-row w3-section">
      <div>
        <button name="btnVoltar" class="w3-button w3-block w3-margin w3-blue w3-cell w3-round-
large" style="width: 90%;">
          Voltar
        </button>
      </div>
    </div>
  </form>
</div>

</body>
</html>

```