
AGENDA 13

PROJETO COMPLETO
– CAMADA
CONTROLLER





MERGULHANDO NO TEMA...

Camada Controller

Para colocarmos em prática, precisamos criar uma pasta para nossa camada Controller. Dentro da mesma pasta que contém as camadas Model e View, na raiz do servidor apache que você utiliza (Imagem 4), é necessário criar uma pasta para a camada Controller. Esta camada gere e define quais dados ou regras devem ser acionadas e para onde serão encaminhados para, posteriormente, serem exibidos ou não pela camada View.



Imagem 4 - Pasta View.

Navegação

Neste momento, criaremos o primeiro arquivo da camada Controller. Esse arquivo, denominado navegação, será o responsável pelo direcionamento dos links para suas respectivas ações e eventos, então, será a classe principal para fluxo das informações.

Agora, dentro da pasta Controller, crie este arquivo PHP denominado “**Navegacao.php**” e não se esqueça de colocar os delimitadores PHP “<?php?>”. Neste primeiro momento, na codificação desse arquivo, será incluída apenas a página login (interface da camada View).

```
<?php
    include_once 'View/login.php';
?>
```

Após salvar, precisamos criar uma página index na raiz do servidor apache, como demonstrado a seguir:

Controller	29/11/2018 10:07
Img	27/11/2018 09:55
Model	22/11/2018 09:54
View	27/11/2018 14:12
index.php	19/11/2018 08:47

Imagem 5 - arquivos Index.

Este arquivo deverá ser codificado da seguinte forma:

```
<!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>Cadastro de Currículos</title>
</head>
<body>
<?php
    include_once 'Controller/Navegacao.php';
?>
</body>
</html>
```

Observe o código `include_once` que contém o arquivo de controle de navegação que, por sua vez, incluirá a interface login, assim, a camada Controller realiza suas atribuições, controlando o fluxo de informações. Por fim, ao executar o site tem-se como resultado a imagem a seguir:



Imagem 6 - arquivo Index

Desta forma, toda action de todos os formulários sempre irão indicar o arquivo **Navegacao.php** para gerenciar o fluxo e direcionamento das páginas do projeto.

Primeiro Acesso

Como a interface Login foi acessada, primeiramente é preciso inserir o caminho para o arquivo **Navegacao.php** na action da camada View. Após realizar a alteração, o código deverá ficar da seguinte forma:

```
<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%;">
```

Ao realizar esse procedimento, no momento em que o usuário clicar em qualquer botão será redirecionado para o Controller de navegação que realizará o direcionamento para o lugar adequado.

Agora precisamos alterar o arquivo **“Navegacao.php”** para que quando o botão **“Primeiro Acesso”** for acionado, o usuário seja direcionado para a interface de primeiro acesso. Então, codifique o arquivo **Navegacao.php** que deve ficar da seguinte maneira:

```
<?php
if(isset($_POST["btnPrimeiroAcesso"]))
{
    include_once '../View/primeiroAcesso.php';
}
else
{
    include_once 'View/login.php';
}
?>
```

De forma simples, criamos um desvio condicional que quando o usuário clicar no botão “btnPrimeiroAcesso”, o controller irá redirecionar nosso usuário para a interface “Primeiro Acesso”. Em caso negativo, o site será redirecionado para a interface login, ou seja, sempre que não for acionado nenhum botão ou link, a interface login irá ser exibida.

Cadastrar

Na página primeiro acesso (Imagem 7), o usuário pode realizar o cadastro do seu login preenchendo todas as informações. Veja:

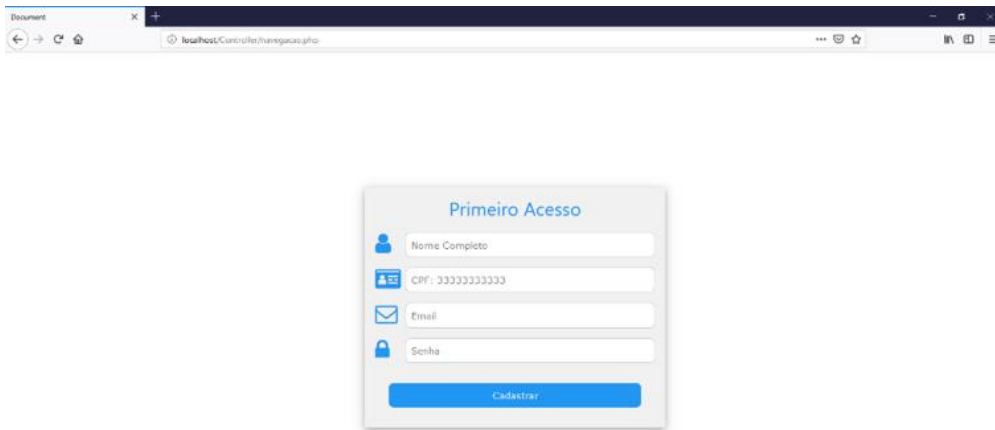


Imagem 7 - Página Primeiro Acesso.

Obs: Não se esqueça: no código da interface deve-se colocar o controller “/Controller/ **Navegacao.php**” no atributo action.

Ao clicar em cadastrar, será necessário ter os dados fornecidos pelo usuário, inseri-los em um objeto usuário da classe modelo e persistir esses dados no Banco de Dados. Mas como fazer isso?

O primeiro passo será realizar a alteração no código **Navegacao.php**, inserindo outro desvio condicional, caso o botão Cadastrar (“btnCadastrar”) for pressionado. Veja o código:

```
if(isset($_POST["btnPrimeiroAcesso"]))
{
    include_once '../View/primeiroAcesso.php';
}
else{
    if(isset($_POST["btnCadastrar"]))
    {

    }
    else
    {
        include_once 'View/login.php';
    }
}
```

Agora partimos para um momento chave!

Precisamos buscar os dados da view e popular um objeto em model, neste momento a função desta camada fica evidente. Para conseguir realizar esse procedimento, criaremos uma classe na camada controle, para que seja possível controlar esse fluxo de informação da camada view para a camada model, e por fim persistir esses dados.

Crie um arquivo denominado “UsuarioController”, lembrando da extensão php. Esta classe terá todos os métodos necessários para controlar os dados pessoais do usuário tais como: nome, CPF etc.

```
<?php

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

class UsuarioController{

    public function inserir($nome, $cpf, $email,$senha) {
        require_once '../Model/Usuario.php';
        $usuario = new Usuario();
        $usuario->setNome($nome);
        $usuario->setCPF($cpf);
        $usuario->setEmail($email);
        $usuario->setSenha($senha);
        //return $usuario->getNome();
        $r = $usuario->inserirBD();
        $_SESSION['Usuario'] = serialize($usuario);
        return $r;
    }
}

?>
```

Vamos entender este código!

Primeiro verificamos se não existe uma sessão criada, caso não exista, criamos uma. Foi criada classe Usuário Controller e, logo em seguida, um método denominado inserir, contendo uma inclusão da classe

Usuário da camada Model, lembrando que essa inclusão é necessária para ser possível a instanciação do objeto desta classe. Então, é criada uma instância da classe Usuário, populando seus atributos por meio dos métodos getters and setters, desenvolvidos previamente na camada Model.

Por fim, chamamos o método inserir BD, que tem como função principal persistir os dados do objeto \$usuario no Banco de Dados, retornando verdadeiro em caso de sucesso ou falso em caso de falha, atribuindo este valor na variável \$r.

Para finalizar, serializamos o objeto. Com essa ação, será possível sua alocação na sessão, tornando possível a transferência de dados entres as páginas. Assim, finalizamos o método com o retorno.

Voltando ao arquivo **Navegacao.php**, logo após os delimitadores php. Vamos verificar se há uma sessão aberta para criá-la ou não, de acordo com a resposta do desvio condicional!

```
<?php

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

}

Ainda neste mesmo arquivo **Navegacao.php**, dentro do desvio condicional, que verifica se o botão “btnCadastrar” foi pressionado, faça a seguinte codificação:

```
require_once '../Controller/UsuarioController.php';
$uController = new UsuarioController();

if($uController->inserir($_POST["txtNome"], $_POST["txtCPF"], $_POST["txtEmail"],
$_POST['txtSenha']))
{
    include_once '../View/cadastroRealizado.php';
}
else
{
    include_once '../View/cadastroNaoRealizado.php';
}
```

Neste código instanciamos um objeto UsuárioController e realizamos a chamada do método inserir programado anteriormente, dentro de um desvio condicional, passando as informações diretamente dos inputs preenchidos pelo usuário para o objeto usuário e tentando realizar a persistência dos dados. Note que neste momento, se o retorno for positivo, é incluída a página “cadastro realizado”. Se o retorno for negativo, a página “cadastro não realizado” será incluída.

Neste momento, precisamos alterar as actions de ambas as páginas da camada View (cadastroRealizado.php e cadastroNãoRealizado.php), atribuindo-lhes o mesmo caminho das demais: “/Controller/Navegacao.php”.

Por fim, criamos outro desvio condicional no arquivo navegação para que os botões btnCadNRealizado e btnCadRealizado sejam redirecionados, respectivamente, para as páginas da camada View primeiroAcesso.php e principal.php, deixando o código do arquivo Navegação dessa forma:

```
if(isset($_POST["btnPrimeiroAcesso"]))
{
    include_once '../View/primeiroAcesso.php';
}
else{
    if(isset($_POST["btnCadastrar"]))
    {
        require_once '../Controller/UsuarioController.php';
        $uController = new UsuarioController();

        if($uController->inserir($_POST["txtNome"], $_POST["txtCPF"], $_POST["txtEmail"],
$_POST['txtSenha']))
        {
            include_once '../View/cadastroRealizado.php';
        }
        else
        {
            include_once '../View/cadastroNaoRealizado.php';
        }
    }
}
```

```

    }
}
else{

    if(isset($_POST["btnCadRealizado"]))
    {
        include_once '../View/principal.php';
    }
    else{
        if(isset($_POST["btnCadNRealizado"]))
        {
            include_once '../View/primeiroAcesso.php';
        }
        else
        {
            include_once 'View/login.php';
        }
    }
}
}
}

```

Ao testar o cadastro e criar um usuário com sucesso, o projeto deverá ficar dessa forma:



Imagem 8 - Fluxo do cadastro do usuário realizado de forma correta.

Dados Pessoais – Exibir dados

Observando a página principal precisamos fazer com que os dados cadastrados pelo usuário estejam disponíveis para visualização na interface da etapa dados pessoais, para isso necessitamos realizar algumas atualizações no arquivo principal.

1. No início da tag Body vamos incluir a classe Usuário e verificar se a sessão está aberta ou não.

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

2. Para cada atributo value dos inputs utilizaremos o objeto serializado na seção para realizar a inserção dos dados. Veja.

a. ID:

```
value="<?php echo unserialize($_SESSION['Usuario'])->getID();?>"
```

b. Nome:

```
value="<?php echo unserialize($_SESSION['Usuario'])->getNome();?>"
```

c. Data De Nascimento:

```
value="<?php echo unserialize($_SESSION['Usuario'])->getDataNascimento();?>"
```

d. CPF:

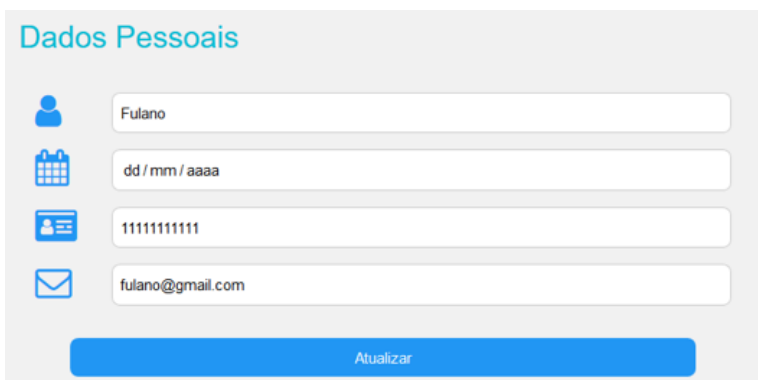
```
value="<?php echo unserialize($_SESSION['Usuario'])->getCPF();?>"
```

e. Email:

```
value="<?php echo unserialize($_SESSION['Usuario'])->getEmail();?>"
```

3. Neste formulário indicar sua action também para: “/Controller/ Navegacao.php”.

Ao carregar a página Dados Pessoais teremos o resultado a seguir:



O formulário, intitulado "Dados Pessoais", contém quatro campos de entrada com ícones representativos: um ícone de pessoa para o nome (contendo "Fulano"), um ícone de calendário para a data de nascimento (contendo "dd / mm / aaaa"), um ícone de documento com uma barra de código de barras para o CPF (contendo "11111111111"), e um ícone de envelope para o e-mail (contendo "fulano@gmail.com"). Abaixo dos campos, há um botão azul com o texto "Atualizar".

Imagem 9 - Fluxo do cadastro do usuário realizado de forma correta

Dados Pessoais – Atualizar dados

O próximo passo será o desenvolvimento do evento para o botão atualizar que, por meio dos dados oriundos dos inputs, atualizará o banco de dados. Para isso, é preciso programar um novo método na classe “usuário controller”, denominado “atualizar”, desenvolvendo o seguinte código:


```

public function atualizar($id, $nome, $cpf, $email, $dataNascimento) {
    require_once '../Model/Usuario.php';
    $usuario = new Usuario();
    $usuario->setId($id);
    $usuario->setNome($nome);
    $usuario->setCPF($cpf);
    $usuario->setEmail($email);
    $usuario->setDataNascimento($dataNascimento);
    $r = $usuario->atualizarBD();
    $_SESSION['Usuario'] = serialize($usuario);
    return $r;
}

```

Seguindo o mesmo padrão do método inserir, será instanciado um objeto da classe Usuário, que terá valores adicionados em seus atributos, por meio dos parâmetros do método atualizar, em seguida, invocamos o método atualizar BD do objeto usuário criado, que retornará “Verdadeiro” caso a atualização tenha sido realizada com sucesso ou “Falso” caso tenha ocorrido alguma falha. Por fim, serializa-se o objeto, atribui-se o mesmo na sessão e retorna “r”.

Finalmente, deve-se desenvolver o arquivo **Navegacao.php**, de acordo com o código a seguir:

```

if(isset($_POST["btnAtualizar"]))
{
    require_once '../Controller/UsuarioController.php';

    $uController = new UsuarioController();

    if($uController->atualizar($_POST["txtID"], $_POST["txtNome"],
$_POST["txtCPF"], $_POST["txtEmail"],
date('Y-m-d', strtotime($_POST['txtData']))))
    {
        include_once '../View/atualizacaoRealizada.php';
    }
    else
    {
        include_once '../View/operacaoNaoRealizada.php';
    }
}

```

Obs: a função date converte do sistema dia/mês/ano para ano/mês/dia.

Com essa codificação, atualizaremos os dados pessoais do usuário no Banco de Dados.

Para as interfaces atualização Realizada e operação não realizada, configure a action do form para: “/Controller/ **Navegacao.php**”.

No arquivo **Navegacao.php** escreva mais um desvio condicional, conforme a codificação a seguir:

```

if(isset($_POST["btnAtualizacaoCadastro"]) ||
isset($_POST["btnOperacaoNRealizada"]))
{
    include_once '../View/principal.php';
}

```

Login

Quando o usuário visitar o projeto pela segunda vez, ele não terá mais a necessidade de criar outro cadastro, então, deve realizar o login com seu CPF e senha cadastrados.

Para conseguir realizar esse procedimento, devemos programar a classe usuário controller com uma função denominada login.

```
public function login($cpf, $senha)
{
    require_once '../Model/Usuario.php';
    $usuario = new Usuario();
    $usuario->carregarUsuario($cpf);
    if($usuario->getSenha() == $senha)
    {
        $_SESSION['Usuario'] = serialize($usuario);
        return true;
    }
    else
    {
        return false;
    }
}
```

O código segue o mesmo padrão dos outros métodos com uma diferença básica: ele tem um desvio condicional verificando se a senha passada como parâmetro é igual à senha que o usuário digitou. Nesse caso, a senha foi obtida por meio do CPF digitado.

No exemplo dessa página de login já foi alterada a action direcionando-a para o arquivo de navegação. Nesse momento, deve-se codificar o botão btnLogin para verificar se o usuário está fazendo o primeiro acesso. Veja essa codificação:

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

    $uController = new UsuarioController();

    if($uController->login($_POST['txtLogin'], $_POST['txtSenha']))
    {
        include_once '../View/principal.php';
    }
    else
    {
        include_once '../View/cadastroNaoRealizado.php';
    }
}
```

Dentro deste código terá outro desvio condicional que, se verdadeiro, será redirecionado para a página principal, caso contrário será redirecionado para cadastro não realizado. Confira a codificação acima.

Formação acadêmica

Na interface formação acadêmica, localizada na página principal, programaremos primeiro a ação para inserir os dados no Banco, a partir do preenchimento dos dados pelo usuário.



O formulário, intitulado "Formação", possui campos para "Data Inicial" e "Data Final", ambos com ícones de calendário e máscara de data "dd/mm/aaaa". Abaixo, há um campo de "Descrição" com o exemplo "Técnico em Desenvolvimento de Sistemas - Centro Paula Souza". Um botão azul com um ícone de pessoa e sinal de mais está centralizado. Na base, uma barra azul contém os botões "Início", "Fim", "Descrição" e "Apagar".

Imagem 10 - Formação acadêmica

Para a realização desse processo, vamos precisar primeiro desenvolver a classe formação acadêmica controller, que terá a função de controlar o fluxo dos dados referente à formação acadêmica. Dentro da pasta da camada controller, crie um arquivo PHP denominado: fomacaoAcadController, sem se esquecer dos delimitadores, verificação e abertura de sessão. Veja a codificação na imagem a seguir:

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

Usando como base o desenvolvimento da classe Usuário controller e pensando em algumas especificidades dessa classe, precisaremos de três métodos:

1. Inserir - Este método não tem nenhum segredo, faz basicamente o mesmo que o programado anteriormente, mas tem a função de instanciar um objeto da classe Formação Acadêmica denominado "FormacaoAcad" e inserir os dados em sua respectiva tabela.

Veja a codificação a seguir:

```
public function inserir($inicio, $fim, $descricao,$idusuario) {
    require_once '../Model/FormacaoAcad.php';
    $formacao = new FormacaoAcad();
    $formacao->setInicio($inicio);
    $formacao->setFim($fim);
    $formacao->setDescricao($descricao);
    $formacao->setIdUsuario($idusuario);
    $r = $formacao->inserirBD();

    return $r;
}
```

2. Remover - Este método é novo, porém o mais simples deles. Esse método realiza um delete de dados na tabela formação acadêmica, por meio de um parâmetro.

```
public function remover($id) {
    require_once '../Model/FormacaoAcad.php';
    $formacao = new FormacaoAcad();
    $r = $formacao->excluirBD($id);
    return $r;
}
```



*Você se lembra do input hidden desenvolvido na Agenda 3?
Então, fique atento porque ele será muito importante nesse momento!*

3. Gerar Lista - quando desenvolvemos a interface da formação acadêmica, definimos uma tabela que contém colunas para: datainicio, datafim, descrição e apagar. Com esse método e uma instância de FormaçãoAcad, teremos disponível a lista de formação acadêmica do usuário logado no sistema de currículo. Veja:

```
public function gerarLista($idusuario)
{
    require_once '../Model/FormacaoAcad.php';
    $formacao = new FormacaoAcad();

    return $results = $formacao->listaFormacoes($idusuario);
}
```

O próximo passo é adicionar o caminho para a execução “/Controller/ **Navegacao.php**” na action do formulário da formação acadêmica. Agora resta apenas codificar o evento do clique do usuário no botão: btnAdd-Formacao.

```
if(isset($_POST["btnAddFormacao"]))
{
    require_once '../Controller/FormacaoAcadController.php';
    include_once '../Model/Usuario.php';

    $fController = new FormacaoAcadController();

    if($fController->inserir(date('Y-m-d', strtotime($_POST['txtInicioFA'])), date('Y-m-d',
    strtotime($_POST["txtFimFA"])), $_POST["txtDescFA"], unserialize($_SESSION['Usuario']->getID()) !=
    false)
    {
        include_once '../View/informacaoInserida.php';
    }
    else
    {
        include_once '../View/operacaoNaoRealizada.php';
    }
}
```

Neste desvio condicional, é preciso incluir duas classes: FormacaoAcadController e Usuário. Isto se dá principalmente por causa da serialização do objeto usuário, que é passado por meio da sessão. Após isso, deve-se seguir o mesmo padrão de instância e de chamada do método (inserir).

Não se esqueça de modificar o action do formulário do arquivo “informacaoInserida” para o caminho de sempre (“/Controller/ **Navegacao.php**”). Nesse caso, altere o desvio condicional de:

```
if(isset($_POST["btnAtualizacaoCadastro"]) || isset($_POST["btnOperacaoNRealizada"]))
{
    include_once '../View/principal.php';
}
```

Para:

```
if(isset($_POST["btnAtualizacaoCadastro"]) || isset($_POST["btnOperacaoNRealizada"]) ||
isset($_POST["btnInfInserir"]))
{
    include_once '../View/principal.php';
}
```

Para a visualização dos dados pelo usuário, é preciso codificar a classe principal em 3 passos:

1. Incluir a classe FormacaoAcadController por meio do código:

```
include_once '../Controller/FormacaoAcadController.php';
```

2. Instanciar, dentro da tabela, um objeto da FormacaoAcadController, que invocará o método gerarLista, e passará como parâmetro o ID do usuário que está armazenado na sessão, que resultando em um objeto contendo um conjunto de formações, caso o retorno tenha sido diferente de nulo.

3. Agora, basta criar uma estrutura de repetição, gerando uma tabela para cada registro encontrado. Veja como deverá ficar o código da tabela:

```
<div class="w3-container">
<table class="w3-table-all w3-centered">
<thead>
<tr class="w3-center w3-blue">
<th>Início</th>
<th>Fim</th>
<th>Descrição</th>
<th>Apagar</th>
</tr>
</thead>

<?php
    $fCon = new FormacaoAcadController();
    $results = $fCon->gerarLista(unserialize($_SESSION['Usuario'])->getID());
    if($results != null)

        while($row = $results->fetch_object()) {
            echo '<tr>';
            echo '<td style="width: 10%;">'.$row->inicio.'</td>';
            echo '<td style="width: 10%;">'.$row->fim.'</td>';
```

```

        echo '<td style="width: 65%;">'.$row->descricao.'</td>';
        echo '<td style="width: 5%;">
        <form action="/Controller/Navegacao.php" method="post">
        <input type="hidden" name="id" value="'.$row->idformacaoAcademica.'">
        <button name="btnExcluirFA" class="w3-button w3-block w3-blue w3-cell w3-round-
        large">

        <i class="fa fa-user-times"></i> </button></td>
        </form>';
        echo '</tr>';
    }
    ?>
</table>

```

Obs.:

Repare que na última coluna criamos um formulário com dois campos para cada registro, o primeiro campo para armazenar o ID da formação, e o outro para inserir o botão de apagar para cada registro de formação.

Depois da programação concluída, o usuário deverá preencher os dados (início, fim, descrição) e pressionar o botão Adicionar para inserir a formação, conforme mostra a imagem 12.



Imagem 11 - Botão de Excluir



Imagem 12 - Botão de Adicionar Formação

Veja, a seguir, o resultado na interface:

A interface de usuário apresenta dois campos de data no topo, cada um com um ícone de calendário e o formato 'dd / mm / aaaa'. Abaixo deles, há um campo de texto para a descrição, precedido por um ícone de menu hambúrguer. O texto dentro do campo é 'Descrição: Ex.: Técnico em Desenvolvimento de Sistemas - Centro Paula Souza'. Centralizado abaixo do campo de descrição está um botão azul com o ícone de uma pessoa e um sinal de mais. Na base da interface, há uma tabela com o seguinte conteúdo:

Início	Fim	Descrição	Apagar
2003-02-01	2005-12-30	Ensino Médio - Centro Paula Souza	

Imagem 13 - Botão de Adicionar Formação

Agora falta pouco!

Para finalizar, é preciso desenvolver o botão “btnExcluirFA” para excluir o registro do banco. Para isso, deve-se inserir no arquivo **Navegacao.php** o seguinte desvio condicional:

```

if(isset($_POST["btnExcluirFA"]))
{
    require_once '../Controller/FormacaoAcadController.php';
    include_once '../Model/Usuario.php';

    $fController = new FormacaoAcadController();

```

```

if($fController->remover($_POST['id']) == true)
{
    include_once '../View/informacaoExcluida.php';
}
else
{
    include_once '../View/operacaoNaoRealizda.php';
}
}

```

Atenção:

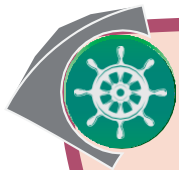
Não se esqueça também de programar um desvio condicional no arquivo **Navegacao.php** para o botão btnInfExcluir ser redirecionado para a interface principal. Veja como fazer isso:

```

if(isset($_POST["btnPrincipal"]) || isset($_POST["btnAtualizacaoCadastro"]) ||
isset($_POST["btnCadRealizado"])
|| isset($_POST["btnInfInserir"]) || isset($_POST["btnInfExcluir"]))
{
    include_once '../View/principal.php';
}

```

Codificando assim de forma simples a exclusão do registro.



VOCÊ NO COMANDO

Com base no que foi estudado até agora...

1. Crie toda a camada controle para a Interface Experiência Profissional, incluindo a criação da Classe Experiência Profissional Controller com seus métodos (Inserir, Remover e gerar Lista);
2. Altere o Arquivo Principal na interface experiência profissional;
3. Codifique o arquivo navegação para que aconteça todo o fluxo entre as camadas (Model, View e Controller).

Obs: não se esqueça de programar as tabelas na interface.

Dicas:

- Utilize a classe Formação acadêmica Controller como base para o desenvolvimento;
- Não se esqueça de programar a action dos formulários.
- Não se esqueça de codificar as ações para os eventos de clique do usuário.
- Não se esqueça de incluir a classe que você desenvolver na interface principal.

A seguir, confira se você conseguiu resolver o desafio proposto!

Antes de exibir a resposta, é importante saber que a solução apresentada a seguir é apenas uma de algumas possibilidades e variações. Em caso de dúvida, contate o seu professor mediador!

Classe Experiencia Profissional Controller.

```

<?php

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

class ExperienciaProfissionalController{

    public function inserir($inicio, $fim, $empresa, $descricao,$idusuario) {
        require_once '../Model/ExperienciaProfissional.php';
        $expP = new ExperienciaProfissional();
        $expP->setInicio($inicio);
        $expP->setFim($fim);
        $expP->setEmpresa($empresa);
        $expP->setDescricao($descricao);
        $expP->setIdUsuario($idusuario);
        $r = $expP->inserirBD();

        return $r;
    }

    public function remover($id) {
        require_once '../Model/ExperienciaProfissional.php';
        $expP = new ExperienciaProfissional();
        $r = $expP->excluirBD($id);
        return $r;
    }

    public function gerarLista($idusuario)
    {
        require_once '../Model/ExperienciaProfissional.php';
        $expP = new ExperienciaProfissional();

        return $results = $expP->listaExperiencias($idusuario);
    }

}

?>

```

Arquivo Navegação – Inserir

```

if(isset($_POST["btnAddEP"]))
{
    require_once '../Controller/ExperienciaProfissionalController.php';
    include_once '../Model/Usuario.php';

    $epController = new ExperienciaProfissionalController();

    if($epController->inserir(date('Y-m-d', strtotime($_POST['txtInicioEP'])), date('Y-m-d',
    strtotime($_POST["txtFimEP"])), $_POST["txtEmpEP"], $_POST["txtDescEP"],
    unserialize($_SESSION['Usuario'])->getID()) != false)
    {
        include_once '../View/informacaoInserida.php';
    }
}

```



```

    }
    else
    {
        include_once '../View/operacaoNRealizada.php';
    }
}

```

Arquivo Navegação - Excluir

```

if(isset($_POST["btnExcluirEP"]))
{
    require_once '../Controller/ExperienciaProfissionalController.php';
    include_once '../Model/Usuario.php';

    $epController = new ExperienciaProfissionalController();

    if($epController->remove($_POST['idEP']) == true)
    {
        include_once '../View/informacaoExcluida.php';
    }
    else
    {
        include_once '../View/operacaoNRealizada.php';
    }
}

```

Arquivo Principal – Tabela

```


| Início                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Fim | Empresa | Descrição | Apagar |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|---------|-----------|--------|
| <?php                 \$ePro = new ExperienciaProfissionalController();                 \$results = \$ePro->gerarLista(unserialize(\$_SESSION['Usuario'])->getID());                 if(\$results != null)                  while(\$row = \$results->fetch_object()) {                     echo '<tr>';                     echo '<td style="width: 10%;">'.\$row->inicio.'</td>';                     echo '<td style="width: 10%;">'.\$row->fim.'</td>';                     echo '<td style="width: 10%;">'.\$row->empresa.'</td>';                     echo '<td style="width: 65%;">'.\$row->descricao.'</td>';                     echo '<td style="width: 5%;">                         <form action="/Controller/Navegacao.php" method="post"> |     |         |           |        |


```

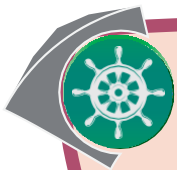
```

round-large">
<input type="hidden" name="idEP" value="'. $row->idexperienciaprofissional.'">
<button name="btnExcluirEP" class="w3-button w3-block w3-blue w3-cell w3-

<i class="fa fa-user-times"></i> </button></td>
</form>';
echo '</tr>';
}
?>

</table>

```



VOCÊ NO COMANDO

2

Com base no que foi estudado até agora...

Bia retorna das férias na próxima segunda, porém o projeto ainda não está finalizado e a equipe Gamma ainda precisa terminar a última etapa do projeto antes do retorno da líder. Para isso é preciso correr contra o relógio.

Nessa corrida contra o tempo, você foi escolhido para auxiliar a equipe Gama a desenvolver a camada controller da etapa outras formações. Então, mãos à obra!

1. Crie toda a camada controle para a Interface Outras Formações que deve incluir a criação Classe OutrasFormações com seus métodos (Inserir, Remover e gerarLista);
2. Alterar o arquivo principal na interface outras Formações.
3. Codificar o arquivo **Navegacao.php** para que aconteça todo o fluxo entre as camadas (Model, View e Controller).

Importante: não se esqueça de programar na interface das tabelas!

Dicas:

- Utilize a classe FormaçãoacadêmicaController ou ExperiênciaProfissional como base para o desenvolvimento.
- Não esqueça da action dos formulários.
- Não esqueça de codificar as ações para os eventos de clique do usuário.
- Não esqueça de incluir a classe que você desenvolver na interface principal.

Classe Outras Formações Controller.

```

<?php

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

class OutrasFormacoesController{

    public function inserir($inicio, $fim, $descricao,$idusuario) {
        require_once '../Model/OutrasFormacoes.php';
        $formacao = new OutrasFormacoes();
        $formacao->setInicio($inicio);
        $formacao->setFim($fim);
        $formacao->setDescricao($descricao);
        $formacao->setIdUsuario($idusuario);
        $r = $formacao->inserirBD();

        return $r;
    }

    public function remover($id) {
        require_once '../Model/OutrasFormacoes.php';
        $formacao = new OutrasFormacoes();
        $r = $formacao->excluirBD($id);
        return $r;
    }

    public function gerarLista($idusuario)
    {
        require_once '../Model/OutrasFormacoes.php';
        $formacao = new OutrasFormacoes();

        return $results = $formacao->listaFormacoes($idusuario);
    }

}

?>

```

Arquivo Navegação – Inserir

```

if(isset($_POST["btnAddOF"]))
{
    require_once '../Controller/OutrasFormacoesController.php';
    include_once '../Model/Usuario.php';

    $fController = new OutrasFormacoesController();

```

```

    if($fController->inserir(date('Y-m-d', strtotime($_POST['txtInicioOF'])), date('Y-m-
d', strtotime($_POST["txtFimOF"])), $_POST["txtDescOF"], unserialize($_SESSION['Usuario'])-
>getID()) != false)
    {
        include_once '../View/informacaoInserida.php';
    }
    else
    {
        include_once '../View/operacaoNaoRealizada.php';
    }
}
}

```

Arquivo Navegação - Excluir

```

if(isset($_POST["btnExcluirOF"]))
{
    require_once '../Controller/OutrasFormacoesController.php';
    include_once '../Model/Usuario.php';

    $fController = new OutrasFormacoesController();

    if($fController->remover($_POST['id']) == true)
    {
        include_once '../View/informacaoExcluida.php';
    }
    else
    {
        include_once '../View/operacaoNaoRealizada.php';
    }
}

```

Arquivo Principal – Tabela

```

<div class="w3-container">
<table class="w3-table-all w3-centered">
<thead>
    <tr class="w3-center w3-blue">

        <th>Início</th>
        <th>Fim</th>
        <th>Descrição</th>
        <th>Apagar</th>
    </tr>
</thead>

<?php
    $fCon = new OutrasFormacoesController();
    $results = $fCon->gerarLista(unserialize($_SESSION['Usuario']->getID()));
    if($results != null)

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

```

```

echo '<td style="width: 10%;">'.$row->inicio.'</td>';
echo '<td style="width: 10%;">'.$row->fim.'</td>';
echo '<td style="width: 65%;">'.$row->descricao.'</td>';
echo '<td style="width: 5%;">
<form action="/Controller/Navegacao.php" method="post">
<input type="hidden" name="id" value="'.$row->idoutrasformacoes.'">
<button name="btnExcluirOF" class="w3-button w3-block w3-blue w3-cell w3-round-large">
<i class="fa fa-user-times"></i> </button></td>
</form>';
echo '</tr>';
}
?>
</table>
</div>

```